Generalized and Efficient Outlier Detection for Spatial, Temporal, and High-Dimensional Data Mining

Erich Schubert

München 2013

Generalized and Efficient Outlier Detection for Spatial, Temporal, and High-Dimensional Data Mining

Erich Schubert

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Erich Schubert
aus München

München, den 29.10.2013

Erstgutachter: Prof. Dr. Hans-Peter Kriegel
Ludwig-Maximilians-Universität München
Zweitgutachter: Prof. Dr. Jörg Sander
University of Alberta, Edmonton, Kanada
Tag der mündlichen Prüfung: 20.12.2013

Zusammenfassung

Knowledge Discovery in Databases (KDD) ist der Prozess, nicht-triviale Muster aus großen Datenbanken zu extrahieren, mit dem Ziel, dass diese bisher unbekannt, potentiell nützlich, statistisch fundiert und verständlich sind. Der Prozess umfasst mehrere Schritte wie die Selektion, Vorverarbeitung, Evaluierung und den Analyseschritt, der als Data-Mining bekannt ist. Eine der zentralen Aufgabenstellungen im Data-Mining ist die Ausreißererkennung, das Identifizieren von Beobachtungen, die ungewöhnlich sind und mit der Mehrzahl der Daten inkonsistent erscheinen. Solche seltene Beobachtungen können verschiedene Ursachen haben: Messfehler, ungewöhnlich starke (aber dennoch genuine) Abweichungen, beschädigte oder auch manipulierte Daten. In den letzten Jahren wurden zahlreiche Verfahren zur Erkennung von Ausreißern vorgeschlagen, die sich oft nur geringfügig zu unterscheiden scheinen, aber in den Publikationen experimental als "klar besser" dargestellt sind. Ein Schwerpunkt dieser Arbeit ist es, die unterschiedlichen Verfahren zusammenzuführen und in einem gemeinsamen Formalismus zu modularisieren. Damit wird einerseits die Analyse der Unterschiede vereinfacht, andererseits aber die Flexibilität der Verfahren erhöht, indem man Module hinzufügen oder ersetzen und damit die Methode an geänderte Anforderungen und Datentypen anpassen kann.

Um die Vorteile der modularisierten Struktur zu zeigen, werden

i) zahlreiche bestehende Algorithmen in dem Schema formalisiert,

ii) neue Module hinzugefügt, um die Robustheit, Effizienz, statistische Aussagekraft und Nutzbarkeit der Bewertungsfunktionen zu verbessern, mit denen die existierenden Methoden kombiniert werden können,

iii) Module modifiziert, um bestehende und neue Algorithmen auf andere, oft komplexere, Datentypen anzuwenden wie geographisch annotierte Daten, Zeitreihen und hochdimensionale Räume,

iv) mehrere Methoden in ein Verfahren kombiniert, um bessere Ergebnisse zu erzielen,

v) die Skalierbarkeit auf große Datenmengen durch approximative oder exakte Indizierung verbessert.

Ausgangspunkt der Arbeit ist der Algorithmus Local Outlier Factor (LOF). Er wird zunächst mit kleinen Erweiterungen modifiziert, um die Robustheit und die Nutzbarkeit der Bewertung zu verbessern. Diese Methoden werden anschließend in einem gemeinsamen Rahmen zur Erkennung lokaler Ausreißer formalisiert, um die entsprechenden Vorteile auch in anderen Algorithmen nutzen zu können. Durch Abstraktion von einem einzelnen Vektorraum zu allgemeinen

Datentypen können auch räumliche und zeitliche Beziehungen analysiert werden. Die Verwendung von Unterraum- und Korrelations-basierten Nachbarschaften ermöglicht dann, einen neue Arten von Ausreißern in beliebig orientierten Projektionen zu erkennen. Verbesserungen bei den Bewertungsfunktionen erlauben es, die Bewertung mit der statistischen Intuition einer Wahrscheinlichkeit zu interpretieren und nicht nur eine Ausreißer-Rangfolge zu erstellen wie zuvor. Verbesserte Modelle generieren auch Erklärungen, warum ein Objekt als Ausreißer bewertet wurde.

Anschließend werden für verschiedene Module Verbesserungen eingeführt, die unter anderem ermöglichen, die Algorithmen auf wesentlich größere Datensätze anzuwenden – in annähernd linearer statt in quadratischer Zeit –, indem man approximative Nachbarschaften bei geringem Verlust an Präzision und Effektivität erlaubt. Des weiteren wird gezeigt, wie mehrere solcher Algorithmen mit unterschiedlichen Intuitionen gleichzeitig benutzt und die Ergebnisse in einer Methode kombiniert werden können, die dadurch unterschiedliche Arten von Ausreißern erkennen kann.

Schließlich werden für reale Datensätze neue Ausreißeralgorithmen konstruiert, die auf das spezifische Problem angepasst sind. Diese neuen Methoden erlauben es, so aufschlussreiche Ergebnisse zu erhalten, die mit den bestehenden Methoden nicht erreicht werden konnten. Da sie aus den Bausteinen der modularen Struktur entwickelt wurden, ist ein direkter Bezug zu den früheren Ansätzen gegeben. Durch Verwendung der Indexstrukturen können die Algorithmen selbst auf großen Datensätzen effizient ausgeführt werden.

Abstract

Knowledge Discovery in Databases (KDD) is the process of extracting non-trivial patterns in large data bases, with the focus of extracting novel, potentially useful, statistically valid and understandable patterns. The process involves multiple phases including selection, preprocessing, evaluation and the analysis step which is known as Data Mining. One of the key techniques of Data Mining is outlier detection, that is the identification of observations that are unusual and seemingly inconsistent with the majority of the data set. Such rare observations can have various reasons: they can be measurement errors, unusually extreme (but valid) measurements, data corruption or even manipulated data. Over the previous years, various outlier detection algorithms have been proposed that often appear to be only slightly different than previous but "clearly outperform" the others in the experiments. A key focus of this thesis is to unify and modularize the various approaches into a common formalism to make the analysis of the actual differences easier, but at the same time increase the flexibility of the approaches by allowing the addition and replacement of modules to adapt the methods to different requirements and data types.

To show the benefits of the modularized structure,

i) several existing algorithms are formalized within the new framework

ii) new modules are added that improve the robustness, efficiency, statistical validity and score usability and that can be combined with existing methods

iii) modules are modified to allow existing and new algorithms to run on other, often more complex data types including spatial, temporal and high-dimensional data spaces

iv) the combination of multiple algorithm instances into an ensemble method is discussed

v) the scalability to large data sets is improved using approximate as well as exact indexing.

The starting point is the Local Outlier Factor (LOF) algorithm, which is extended with slight modifications to increase robustness and the usability of the produced scores. In order to get the same benefits for other methods, these methods are abstracted to a general framework for local outlier detection. By abstracting from a single vector space, other data types that involve spatial and temporal relationships can be analyzed. The use of subspace and correlation neighborhoods allows the algorithms to detect new kinds of outliers in arbitrarily oriented subspaces. Improvements in the score normalization bring back a statistic intuition of probabilities to the outlier scores that previously were only useful for ranking objects, while improved models also offer explanations of why an object was considered to be an outlier.

Subsequently, for different modules found in the framework improved modules are presented that for example allow to run the same algorithms on significantly larger data sets – in approximately linear complexity instead of quadratic complexity – by accepting approximated neighborhoods at little loss in precision and effectiveness. Additionally, multiple algorithms with different intuitions can be run at the same time, and the results combined into an ensemble method that is able to detect outliers of different types.

Finally, new outlier detection methods are constructed; customized for the specific problems of these real data sets. The new methods allow to obtain insightful results that could not be obtained with the existing methods. Since being constructed from the same building blocks, there however exists a strong and explicit connection to the previous approaches, and by using the indexing strategies introduced earlier, the algorithms can be executed efficiently even on large data sets.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Name, Vorname

....

Ort, Datum Unterschrift Doktorand/in

Formular 3.2

Preface

This thesis is the culmination of my research and learning over a period of 5 years (2008–2013) in the domain of outlier detection. The plan, of course, was to be finished much earlier, but there were – and still are – too many interesting questions open for further research and improvements. You only write a dissertation once, so it is hard to let go.

My research in this domain was motivated from my earlier diploma thesis on robust correlation clustering [Kri+08], an approach which was affected by outliers and thus raised my interest in scalable and robust multivariate statistics; but I have also always been interested in efficiency and scalability, which is why I prefer the data mining point of view. Mathematicians will probably find many of the approaches discussed in this thesis to be lacking with respect to their desire to have mathematically proven models; yet, automatic finding such models for large, real data sets is not possible. And while some approaches such as Naïve Bayes may be mathematically nonsense when there are dependencies amongst attributes, they can still be surprisingly effective in practice. Data mining can then be seen as the skill of bringing together just enough statistics to make the results work, but also only so much statistics as to make it scale to the real data at hand; even if this may mean making unrealistic assumptions and simplifications. In this thesis, I will look at outlier detection in data mining, where the majority of methods are driven by the need to compute them effectively, not by the desire to find a mathematical model of the data. I will, however, not just try to propose yet another new method. Instead, the focus is on reviewing and rethinking existing methods, understanding how – and why – they work as well as their limitations. We then can try to bridge the gap between the concrete algorithms and the mathematical models a little, and make them use more statistics, to make them hopefully more robust and also a little bit more respectable from the often harsh point of view of mathematics.

> “We should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data. — Peter Norvig [HNP09]”

The first ideas in the direction of increasing the robustness of outlier detection were published as the method “Local Outlier Probabilities” [Kri+09a]; and while the method performs well in experiments and has received a good amount of citations and independent validation (for example in acoustic emission monitoring [Mej12] and system performance monitoring [Ehl12]), I have never been entirely content with this method as-is. Similarly, “Subspace Outlier Degree” [Kri+09b] and “Interpreting and Unifying Outlier Scores” [Kri+11] have received good attention, but are not the “perfect” approaches that I would like them to be. None of these methods is fundamentally flawed or not sound; they all live up to the scientific standards of early research published in well-respected and peer-reviewed conferences. They do not need to be changed, yet they can be improved with a better theoretical background.

When it then came to summarizing the early research, I could not faithfully discuss the original versions without commenting on their limitations and updating them to the best of my current knowledge. In the end, when trying to explain both the published algorithms and my current, deeper understanding, it all took much more time than expected; but it also improves the research. As such, the majority of the work on this thesis was not just summarizing my earlier research, but revisiting all of it with scrutiny; more often than not redesigning the methods using a deeper theory and improved statistics. This can best be seen in Chapter 5 (Improving Local Outlier Detection), which not only explains the method LoOP, but also raises the question why LoOP works and whether it actually improves robustness and lives up to the goal of a "probabilistic" outlier score.

Other parts of the thesis such as Chapter 8 (Scalability) have been present as unfinished work for some time. They could easily have been left out of the thesis, but for practical use they play an important role and display the breadth of my activity, which not only includes the design of outlier detection algorithms, but also implementation and their comparative evaluation in the data mining toolkit ELKI. By publicly providing open-source implementations of the majority of my work (with the ultimate goal of making everything available and easy-to-use), I hope to both foster the reproducibility of research results, and to make it easier for other researchers to experiment with both my own algorithms, and the algorithms of others that I managed to reproduce. As of writing this, ELKI likely has the largest collection of outlier detection algorithms; and more often than not the fastest and most flexible versions, too.

While I have been in open-source software development before my studies, I consider the publishing not only of theory, but also of source code to be an important part of science. Published source code, with the right to modify and redistribute modified versions on the same terms, is beneficial to the goals of science. While it may lead to my own developments to be superseded sooner, this is not of shame to a scientist (or developer); in particular if the successor may be built upon the own work. If so, we have succeeded at enabling others to perform research.

> “Wissenschaftlich aber überholt zu werden, ist – es sei wiederholt – nicht nur unser aller Schicksal, sondern unser aller Zweck. Wir können nicht arbeiten, ohne zu hoffen, daß andere weiter kommen als wir.
>
> (Being overtaken in our scientific work is not only our common fate … but our common mission. We cannot work without hoping that others will surpass us.)
> — Max Weber [Web85], English translation: [Deu97]”

The majority of this work was in collaboration with my advisor, Hans-Peter Kriegel, and my senior colleagues Arthur Zimek and Peer Kröger, but also others inside and outside of the research group, such as Mike Houle in Tokyo.

Contents Overview

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

AUC	Area under the curve, usually with respect to ROC, see Section 5.4.1
cdf	Cumulative Density Function, see Section 1.2
erf	Error Function, usually Gaussian Error Function, see Equation 1.2
$E[X]$	Expected value of X, usually the arithmetic mean
gain	Relative change towards optimal result, see Equation 7.7
i.i.d.	Independent and identical distributed
k-dist	Distance to the k nearest neighbor, Equation 3.2.
$k\mathrm{NN}(o)$	k nearest neighbors of o
K-S-test	Goodness of fit test by Kolmogorov and Smirnov [Kol33; Smi48]
LMM	L-Moment based estimators [HWW85; Hos91]
$M_p(X)$	Generalized mean of X, see Section 2.1.2, Equation 2.5
MAD	Median average deviation, $\mathrm{MAD}(X) := \mathrm{median}_{x\in X}\{x - \mathrm{median}(X)\}$
MLE	Maximum-Likelihood Estimation, an optimal estimator for certain preconditions
MOM	Method of Moments, estimating distribution parameters from mean, standard deviation, skewness and kurtosis
$\mathcal{O}(\_)$	Asymptotic upper bound for complexity (Landau notation)
pdf	Probability Density Function, see Section 1.2
P-P-plot	Scatterplot plotting observed vs. theoretical quantiles of data
$\mathrm{SNN}_s(o,p)$	Shared nearest neighbors of o and p, see Equation 4.3, Section 4.4
w.r.t.	with respect to
$z(x)$	Standardized value: $z(x) := (x-\mu)/\sigma$, see Equation 1.1

1 Introduction

1.1 Outlier Definitions

“ The intuitive definition of an outlier would be “an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism”. — *Douglas M. Hawkins* [Haw80] ”

“ An outlying observation, or “outlier,” is one that appears to deviate markedly from other members of the sample in which it occurs. — *Frank E. Grubbs* [Gru69] ”

“ An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data — *Vic Barnett and Toby Lewis* [BL94] ”

“ A few unusual observations that do not seem to belong to the pattern of variability produced by the other observations
— *Richard A. Johnson and Dean W. Wichern* [JW92] ”

While these quotes reflect the common understanding of outliers, there is no universally used formal definition. However, there are some recurring aspects of outlierness. Outliers are:

- rare – single observations or small subsets.
- “different” or “inconsistent” …
- …compared to the patterns of the “other” observations
- …probably produced by a different “mechanism”.

The closest tool we have to a theoretical foundation of outliers probably is that of statistical hypothesis testing: given a statistical model of our data set, we can perform a test using the null hypothesis $\mathcal{H}_0$: “the object is generated by the data set model” versus the alternative $\mathcal{H}_1$: “the object is not generated by the model”. Objects for which we have to reject the null hypothesis then are our outliers. In practice, as we will see, we will however rarely be able to construct a complete model of our data set that allows for this kind of reasoning. Yet, part of this thesis will be an attempt to embrace both existing algorithms for efficient outlier detection and statistical reasoning inspired by hypothesis testing and connect these two worlds with each other. We should however, bear in mind that there can also be what is sometimes called the “Type III error”: using the wrong hypothesis – i.e. incorrectly modeling our data.

Outliers are not restricted to data or measurements. The book "Liars & Outliers" [Sch12] looks at the role of trust and rules in society, and how we are forced everyday to decide whether to obey the laws – or break them, such as lying on the taxes, speeding, or even stealing the neighbors car instead of buying one oneself. Some such behavior may be common ("liar"), other behavior is rare ("outliers") and the effect on society is different. In the following, however, we will be focusing on outliers in data that we can more easily analyze in a computer than predicting human behavior.

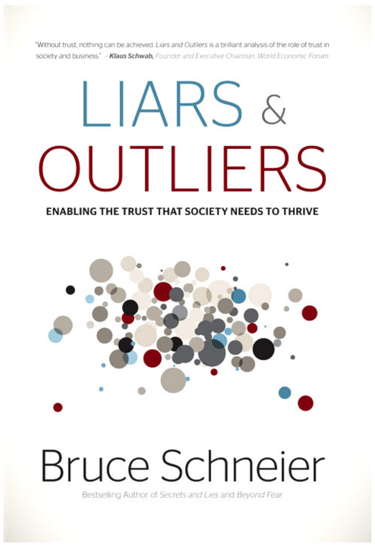

"Without trust, nothing can be achieved. Liars and Outliers is a brilliant analysis of the role of trust in society and business." –Klaus Schwab, Founder and Executive Chairman, World Economic Forum
LIARS & OUTLIERS
ENABLING THE TRUST THAT SOCIETY NEEDS TO THRIVE
Bruce Schneier
Bestselling Author of Secrets and Lies and Beyond Fear

Figure 1.1: Book cover: Liars & Outliers [Sch12]

There can be very different reasons for "outlier" observations in a data set.
Common explanations for the presence of outliers in a data set are

- Measurement errors
- Unusually extreme deviations (but nevertheless real)
- Data input, processing and transmission errors
- Attacks, manipulation and fraud

In the first and third situation, the affected values will usually be removed for data cleaning, in the second case removing the data may sometimes improve the result, but it may not be necessary or desired. In the last situation, the outlying observations are of key interest, while normal data will not be further analyzed. The first three may appear to be quite similar in their manifestation, but they differ in the source of the errors. In the first setting, it is an error of the measurement process, in the second the measurement process may have been perfect, but the observed process has some extreme observations (which is actually common in physical processes), while in the third both may be good, but the data "corruption" occurs at a later stage of the data processing.

Real world data is not as clean and perfect as digital technology pretends. A minute may occasionally have 61 seconds, a day will usually have 24 hours, but in some regions one day a year has 23 and one has 25 hours. In many calculations, a computer will not use the exact value of $1/3$, but a value that is incorrect by an error on the order of $\mathcal{O}(10^{-16})$ – the best approximation of $1/3$ using a IEEE754 double precision floating point number. Physical measurement errors and extreme deviations have been extensively studied in history. The most surprising result with respect to measurement errors probably is the Heisenberg principle of uncertainty, which essentially shows that there is a physical limit to the quality of a measurement that we can perform. Since we cannot get rid of measurement errors, we need processes and statistics that can handle data with errors. Even at the macroscopic level we will always have some measurement errors that we need to be able to deal with. They may arise not only from limited

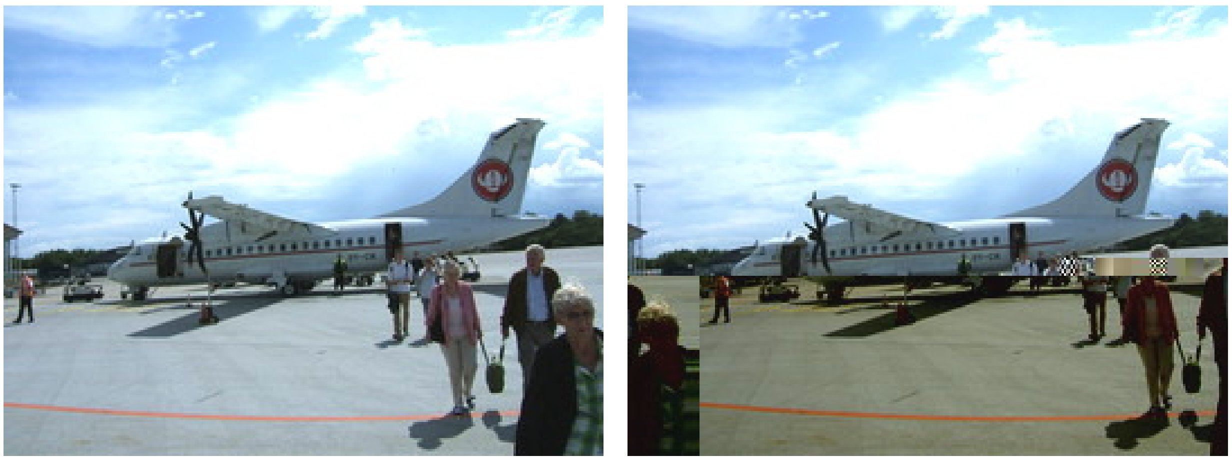


Figure 1.2: Image corruption caused by 1 out of 28466 bytes missing from the file.
This "outlier" can easily be detected by checksums and file format inspection.

precision in the measurement devices, but also in data storage and processing (such as floating point precision errors and rounding errors) or they may be a property of the natural process. Geyser erruption cycles for example do not occur with the precision of an atomic clock, and the full geophysical process is a system with a complexity beyond what we can model easy. No data cleaning process will make these observations perfect, yet it can help our mathematical methods in predicting the next eruption with higher precision; investigating the unusual observations can help understanding the complex geologic process better. Many of such physical processes – both within the measurement processes and the natural processes observed – are modeled using normal distributions. The normal distribution, also known as Gaussian distribution, has been shown both empirically and mathematically (in the form of the central limit theorem) to be a reasonably good approximation for many complex systems. Roughly speaking, the observed error is the combination of probably hundreds of different smaller errors of similar magnitude and distribution and will in total then appear to be normal distributed. With digital technology, in particular data compression, we see a differently natured error. Instead of being normally distributed, the observed errors are often much more extreme. Figure 1.2 visualizes the effect of 1 byte missing from a 28466 byte image file. Instead of producing the expected $.003\%$ error, the visual image corruption affects almost half of the image and creates both a color shift and moves large parts of the image by roughly 10% of the image width. This exemplifies how even small errors in digital processes can produce major errors. However, such errors may or may not be of interest to outlier detection: file integrity can be verified by checksums and transmission errors can be reduced by using error correcting codes. CD discs for example have three layers of error correcting codes, and store 2048 bytes of payload data using 2352 raw bytes on the disc to reduce the likelihood of an *uncorrectable* error. Yet this shows that we need to be able to handle different kinds of errors: errors coming from physical processes that often follow a low-magnitude normal distribution, but also errors coming from any kind of data corruption, that in particular for digital data can be of extreme magnitudes.

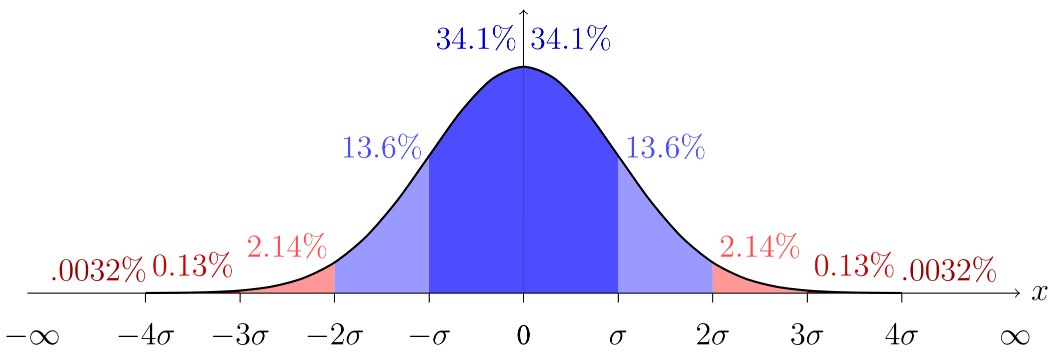


Figure 1.3: Probability densitiy function of the standard normal distribution $\mathcal{N}(\mu = 0; \sigma = 1)$. Percentage numbers indicate the relative area of the curve segment.

1.2 Outliers in Gaussian Distributions

As mentioned in the previous section, Gaussian distributions (also known as normal distributions) occur frequently in physical processes and are mathematically supported by the central limit theorem. Normal distributions are literally omnipresent: while it may as well be only a textbook example, the foot sizes of a larger population are commonly given as an example for normal distributed data.[1]

The normal distribution is a good starting point for understanding outliers in statistics (while the uniform distribution is even simpler, it does not have what one would commonly consider "outliers", since all elements within the value range have the same probability). Figure 1.3 shows the probability density function (PDF) of the standard normal distribution (centered at $\mu = 0$ and scaled to standard deviation $\sigma = 1$). The majority of the objects are expected to be in the center – 68.2% in the dark blue area, 95.4% in the joint blue area. Yet also some rare observations are expected to be further apart: 0.0063% are expected to be either left of the -4σ or right of the 4σ mark. Traditionally in statistics these objects would rarely be seen: 1 in 15787 observations is expected to fall within this range. However in a data set of 1 million objects, the estimated number of such extreme observations already is 63. In statistics, in analyzing a normal distribution, objects outside of a certain range are called outliers. The exact range (or threshold), however, varies.

Formally, the outlier decision can then be formalized as

$$\frac{|x - \mu|}{\sigma} \geq \tau$$

for a threshold τ, where μ is the mean and σ is the standard deviation. These two parameters serve the purpose of rescaling the distribution to the standard normal distribution $\mathcal{N}(0, 1)$. This

[1]Apparently this has indeed been verified by the Canadian Forces in an attempt to improve combat boot fit.

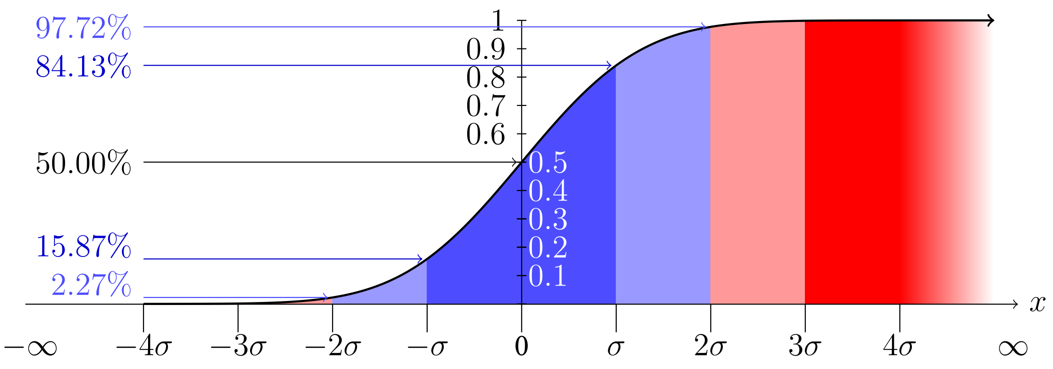


Figure 1.4: Cumulative density function of the standard normal distribution $\mathcal{N}(\mu = 0; \sigma = 1)$. Percentages indicate well quantiles of the cdf at $-2 \ldots 2$.

standardization is popular, and the formula is also known as the z-score:

$$z(x) := \frac{x - \mu}{\sigma} \tag{1.1}$$

which allows us to rewrite the threshold inequality to

$$|z(x)| \geq \tau$$

The additional absolute value function serves the purpose to combining two symmetric one-sided tests into a two-sided test .

Three common versions of choosing the threshold are:

- Three-sigma-rule: $\tau = 3$ (which can be written as $|x - \mu| \geq 3\sigma$)
- Chauvenet's criterion [Cha08]: $\tau = -\mathrm{probit}(\frac{1}{2n})$
- Grubbs' test: $\tau = \frac{n-1}{\sqrt{n}} \sqrt{\frac{t^2_{\alpha/(2n),n-2}}{n-2+t^2_{\alpha/(2n),n-2}}}$

The statistical intuition behind this test can best be understood by looking at Chauvenet's criterion and the function probit used there – which is also known as percent point function or inverse cumulative density function cdfinv – as well as its inverse, the *cumulative density function* cdf (also known as quantile function $\mathrm{quantile}$). The probit function computes the value τ, at which the area below the pdf curve surpasses the given threshold, while $\mathrm{cdf}(x)$ computes the area below the pdf curve left of the given position x. Figure 1.4 visualizes the cdf function of the standard normal distribution. In Figure 1.3, half of the area is left of 0; therefore $\mathrm{cdf}(0) = .50$. Conversely, $\mathrm{probit}(2.27\%) \approx -2$, i.e. approximately 2.27% of the area of the pdf curve are left of $x = -2$ in Figure 1.4

If we now apply the inverse to Chauvenet's criterion

$$\begin{aligned} |z(x)| &\geq -\mathrm{probit}(\alpha) \qquad (= \tau) \\ \mathrm{cdf}\,(-|z(x)|) &\geq \alpha \end{aligned}$$

and use $\text{cdf}(-a) = 1 - \text{cdf}(a)$, we obtain the following two inequalities from this:

$$\text{cdf}\,(z(x)) \geq \alpha \qquad \text{and}$$
$$\text{cdf}\,(z(x)) \leq 1 - \alpha.$$

This is a two sided test, that is failed by the *smallest α objects* and symmetrically by the *largest α objects*, intuitively the 2α most extreme observations. For the purpose of testing, instead of computing the cdf of every single observation, it is of course much easier to compute the threshold τ, at which the desired probability level of α is surpassed. This threshold τ effectively partitions the area into three parts: a lower tail $]-\infty;-\tau]$ with an area of $\text{cdf}(-\tau) = \alpha$, an upper tail $]\tau;+\infty[$ with an area of $1 - \text{cdf}(\tau) = \text{cdf}(-\tau) = \alpha$ and the central area $]-\tau;\tau]$ of size $1-2\alpha$. Objects in the central area are accepted; objects in the tails rejected by the test. The size of the acceptance area – i.e. $1 - 2\alpha$ – can be expressed in terms of the cdf or the Gaussian error function erf:

$$\text{cdf}(\tau) - \text{cdf}(-\tau) \equiv 2 \cdot \text{cdf}(\tau) - 1 \equiv 1 - 2 \cdot \text{cdf}(-\tau) =: \text{erf}\left(\tau/\sqrt{2}\right) \tag{1.2}$$

There are some well-known values of this area, known as the “68–95–99.7 rule”:

τ	$\text{cdf}(\tau)$	$\text{cdf}(\tau) - \text{cdf}(-\tau)$
0	.500000000	.000000000
1	.841344746	.**68**2689492
2	.977249868	.**95**4499736
3	.998650102	.**997**300204

So given a normal distribution, 68% of the objects are expected to be within one standard deviation of the mean, 95% within two standard deviations and 99.7% within three standard deviations. Accordingly, the “three sigma rule” refers to the central 99.7% quantile.

Chauvenet’s criterion can in turn be interpreted in the context of a normal distribution. In general, the cumulative density function can be seen as the transformation from the source distribution to the uniform distribution on the interval $[0, 1]$; and we can intuitively expect the smallest or largest to be at approximately $1/2n$ from the borders 0 and 1.

Grubbs’ test uses a significance level α and is based on the student’s t distribution instead of the normal distribution. This takes into account that the mean and standard deviation have to be estimated from the data as well. With increasing n, the difference between the t distribution and the normal distribution disappears, so that Grubbs’ for $\alpha = 1$ becomes similar to Chauvenet’s criterion. A simplified version of Grubbs’ (or a generalization of Chauvenet’s to different thresholds α) can be formalized as $\tau = -\text{probit}(\frac{\alpha}{2n})$. At $n = 10^6$ instances, the resulting threshold τ deviates by less than 10^{-4} from Grubbs’.

Note that in this situation, all objects actually *are* generated by the same normal distributions. What is considered an “outlier” in this scenario merely are objects in a less dense area of the distribution and can be considered “rare” observations.

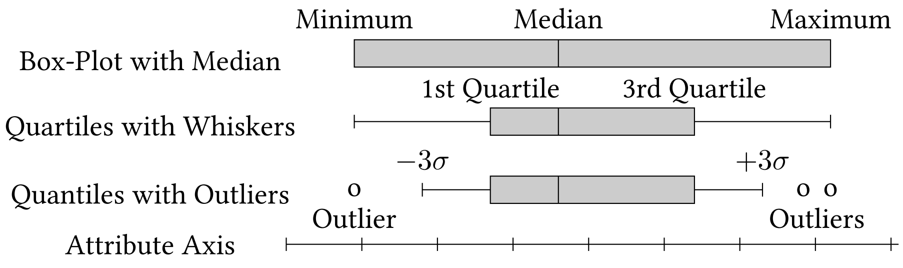


Figure 1.5: Different variants of box- and whisker plots.

1.3 Whisker Plots

The probability density function (PDF) visualized in Figure 1.3 can be approximated for empirical data by using a histogram. However, a PDF or histogram cannot visualize the position of outliers very well, since they do not come in large enough quantities to produce bars – in above figure, the outlier areas outside of 3σ can barely be seen or quantified. The classic intuition for outliers in a normal distribution is however based on quantiles. So instead of plotting the PDF, we can just indicate the quantile positions on the value axis. In Figure 1.3 we did this implicitly: the mean was at 0 and the multiples of σ obviously indicated quantiles according to the "68–95–99.7 rule". A more explicit and popular formalism for this are box plots and whisker plots. These exist in several variations, the simplest being a box from minimum to maximum (i.e. visualizing the 0% and 100% quantiles), the box-and-whiskers plot with the five-number summary (plotting the minimum, lower quartile, median, upper quartile and maximum) while in the more sophisticated versions extreme observations are drawn as outliers, while the whiskers indicate standard deviations. Figure 1.5 shows a few variations of this technique. An interesting property of the advanced versions is that they *summarize* the majority of the data into just 5 numbers, while the outliers are explicitly given. However, for data mining, this technique cannot be simply adapted. First of all, it assumes that we only have a few single outliers. In a large data set, there may be hundreds. Secondly, this is a single-dimensional technique, and in data mining we are particularly interested in multidimensional data, and outliers that only become apparent when we are looking at multiple dimensions at the same time.

1.4 Dealing with Outliers

“ Are they outliers because the data points are bad or because the mathematical form of the equation (i.e., the theory) is wrong? If a data point is suspected of being an outlier, can an experimental reason (i.e., an experimental error) be found to justify eliminating the particular data point? — *Michael L. Johnson* [Joh00] ”

“ We must emphasize that not all outliers are wrong numbers. They may, justifiably, be part of the group and may lead to a better understanding of the phenomena being studied. — *Richard A. Johnson and Dean W. Wichern* [JW92] ”

“ The phrase “appears to be inconsistent” is crucial. It is a matter of subjective judgement on the part of the observer whether or not some observation (or subset of observations) is picked out for scrutiny. — *Vic Barnett and Toby Lewis* [BL94] ”

“ In science, one man’s noise is another man’s signal.
— *Edward W. Ng* [Bla90] based on
Lucretius (1st century BC): “What is food to one, is to others bitter poison.” ”

As indicated by the quotes above, there is a lot of subjectivity in what constitutes an outlier, and how an outlier should be treated. In some situations it may be appropriate to remove the outlier for data cleaning purposes to obtain a better trend estimation. They might stem from outright measurement errors indicating data that actually should be discarded. In another context, the presence of outliers may indicate that our modeling assumptions are incorrect, and we should in fact discard the model and restart with improved assumptions. In yet another context, the individual observations that are outlying are of key interest, such as in fraud detection where the outliers are candidates for detailed inspection.

In the quote above, Edward W. Ng referred not particularly to outlier detection, but to the value of data deemed to be incorrect: ozone readings captured during U.S. space flights in the 1970’s were at that time discarded and considered to be erroneous. Fortunately, however, the data was not deleted but the tapes were kept in an archive. In the 1980’s, when the thinning of the ozone layer was suggested, these measurements could be used to confirm this trend – and show that the data was actually correct, but our assumptions on the data were wrong.

As such, when doing outlier detection, we should try to not think of it as “incorrect” data, but instead of data that is interesting, because it deviates from what we would expect. A good method will not just recommend which data to discard, but which observations to analyze. Ideally, it will even give a hint of how the data deviates from the expectations to help with this analysis.

2 Preliminaries

In this section, some basic mathematical concepts are summarized that will be used later. In order to improve outlier detection results, we will be experimenting with alternate averages than the arithmetic average, use the close relationship of means and distance functions and also use kernel density estimation.

2.1 Averages and Generalized Means

“Then there is the man who drowned crossing a stream with an *average* depth of six inches. — *W.I.E. Gates*”

“Average” is a term with very different meanings. In everyday language, it can be seen as the very opposite of an outlier: anything that is typical, common, normal, middle, central or in some other way usual or representative is called “average”. When dealing with numbers, most people will think of the arithmetic mean where the term “average” is used. For example the SQL idiom `AVG()` refers to the arithmetic mean.

2.1.1 Pythagorean Means

The most well known mean is the arithmetic mean, defined for a set $X := \{x_1, \ldots, x_n\}$:

$$\text{arithmetic-mean}(X) := \frac{1}{|X|} \sum_{i=1}^{n} x_i \tag{2.1}$$

As this is the most popular choice, we will be denoting this simply as $\text{mean}(X)$. However, there are a number of other alternatives in use, where each is more appropriate in one situation or another. The arithmetic mean is the linear additive average, while for example the geometric mean is the corresponding multiplicative average, and the harmonic mean is averaging the reciprocals. These three are known as the Pythagorean means. The fourth that fits nicely to this set is the quadratic mean, also known as root-mean-square. For two values a and b, these means can be geometrically constructed as visualized in Figure 2.1.

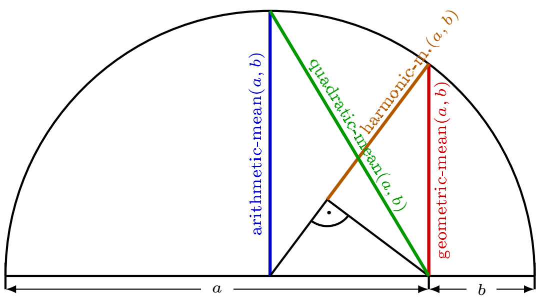


Figure 2.1: Geometric construction of Pythagorean means

Formally, they are defined as:

$$\text{geometric-mean}(X) := \sqrt[n]{\prod_{i=1}^{n} x_i} \tag{2.2}$$

$$\text{harmonic-mean}(X) := |X| / \sum_{i=1}^{n} \frac{1}{x_i} \tag{2.3}$$

$$\text{quadratic-mean}(X) := \sqrt{\frac{1}{|X|} \sum_{i=1}^{n} x_i^2} \tag{2.4}$$

2.1.2 Generalized Means

These means can be generalized to what is known as the generalized mean, power mean, or Hölder mean.

$$\text{generalized-mean}_p(X) :\equiv M_p(X) := \sqrt[p]{\frac{1}{|X|} \sum_{i=1}^{n} x_i^p} = \left(\frac{1}{|X|} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} \tag{2.5}$$

This definition can be extended to consistently also cover $p \in \{0, -\infty, +\infty\}$. The previously discussed Pythagorean means then all show up as special cases of this definition:

$$\begin{aligned}
M_{-\infty}(X) &:= \min\{X\} = \lim_{p \to -\infty} M_p(X) \\
M_{-1}(X) &= \text{harmonic-mean}(X) \\
M_0(X) &:= \text{geometric-mean}(X) = \lim_{p \to 0} M_p(X) \\
M_1(X) &= \text{mean}(X) \\
M_2(X) &= \text{quadratic-mean}(X) \\
M_{+\infty}(X) &:= \max\{X\} = \lim_{p \to +\infty} M_p(X)
\end{aligned}$$

2.1.3 Median, Mode and Trimmed Means

Other more robust definitions of "average" include the median, the mode and trimmed means. The **median** of a set is the most central element. Half of the observations are at least as big, and half of the observations are at least as small as the median. In statistics, the median can be defined by

$$P(x \leq \operatorname{median}(X)) \geq \frac{1}{2} \quad \wedge \quad P(x \geq \operatorname{median}(X)) \geq \frac{1}{2}.$$

The **mode** is the most likely observation. Finally, the **trimmed mean** (or: truncated mean) tries to combine the advantages of medians and means, by computing the mean only on a central share of the fraction, for example only on the second and third quartile (interquartile mean). Winsorization [Has+47] is similar, but replaces extreme values with less extreme data instead of dropping them completely. Assuming that X is sorted and $\alpha \in]0; 0.5[$, we can define it as

$$\text{trimmed-mean}_{p,\alpha}(X) := M_p(x_i, \ldots, x_j) \quad \text{with } i = \alpha|X|, j = (1-\alpha)|X|.$$

2.1.4 Choosing the Appropriate Mean

The choice of the right mean depends on the context and purpose of the analysis. Since there is no best mean, one needs to perform careful analysis of the task to choose the right formula. This can best be seen in this classic riddle:

> A car travels at a speed of 30 mph over a certain distance, and then returns over the same distance at a speed of 20 mph. What is the average speed for the total trip?

The naïve answer, using the arithmetic mean, is $(20 + 30)/2 = 25$ mph. The correct answer however is 24 mph – the harmonic mean. The reason is that the speed is not an additive value, but we instead need to compute the average *time* traveled. In human perception (not taking relativity into account), time is an additive quantity. The relationship of speed to time is $v \sim 1/t$, and substituting this into the arithmetic mean yields the harmonic mean.

Root-mean-square (RMS) – the quadratic mean – is a common best practice in error statistics, as it is closely related to the method of least squares attributed to Carl Friedrich Gauss and Adrien-Marie Legendre. The Gauss–Markov theorem proves that within certain preconditions, the best linear unbiased estimator is the least squares estimator. And last but not least, if we assume that a data set is normal distributed, the quadratic mean of the deviations yields the maximum likelihood estimation of the standard deviation. But there are also many physical effects where the squared values occur naturally: for example the surface of a sphere grows with the squared radius. So if an asteroid continues to travel at the observed speed, the possible area grows with the square of the time since the last observation. Similarly, power of an alternating current will usually have an arithmetic mean of 0. The power delivered however is $p(t) = v(t)^2/r$ where r is the resistance, v the voltage and p the delivered power. In order to estimate the power delivered by an alternating current, the quadratic mean is appropriate:

$P_{\text{avg}} = M_2(v)^2/r$. Mains power supply in many countries is specified as $M_2(v) = 230\,V$ (plus some tolerance). Assuming that the power is provided as a perfect sine curve, the peak voltage then actually is $\pm 230\,V \cdot \sqrt{2} = \pm 325\,V$, since $M_2(\sin) = 1/\sqrt{2}$. The power delivered by this AC power source is comparable to that of to a $230\,V$ DC power source; therefore the RMS value is more important for applications that rely on average power instead of peak voltage.

2.2 Distance Functions, Metrics and Norms

Metrics, distance functions and Norms in mathematics are closely related concepts. Metric and distance functions are usually defined as synonyms and are functions $d : X \times X \to \mathbb{R}$; while a norm has the simpler signature $d : X \to \mathbb{R}$ which can be used to induce a metric using $d(x, y) := d(x - y)$. Intuitively, a norm measures the *length* of a vector, and distance functions induced by norms are those where the distance is the length of the difference vector. The majority of the popular distance functions on vector spaces are induced by a norm, but outside of vector spaces – in particular when $x - y$ is no longer defined – we can no longer have norms, but we can still have distance functions.

Metrics and norms must satisfy similar properties, which we will summarize here:

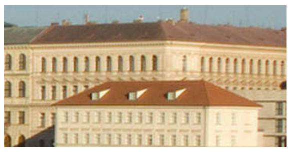
Property	Metric formulation	Norm formulation
Non-Negative	$\forall_{x,y} d(x, y) \geq 0$	$\forall_x d(x) \geq 0$
Identity of Indiscernibles	$\forall_{x,y} d(x, y) = 0 \Leftrightarrow x = y$	$\forall_x d(x) = 0 \Leftrightarrow x = 0$
Symmetry	$\forall_{x,y} d(x, y) = d(y, x)$	$d(x) = d(\lvert x\rvert)$
Triangle Inequality	$\forall_{x,y,z} d(x, y) \leq d(x, z) + d(z, y)$	$\forall_{x,y} d(x + y) \leq d(x) + d(y)$
Homogenity	(usually not required)	$\forall_{\alpha\in\mathbb{R},x} d(\alpha \cdot x) = \lvert\alpha\rvert \cdot d(x)$

Metrics induced by norms have some additional properties that are often useful in practice, and that we tend to assume to hold in general. In particular, these distances are translation invariant ($\forall_{x,y,z} d(x + z, y + z) = d(x, y)$), linear scalable ($\forall_{a\in\mathbb{R},x,y} d(\alpha x, \alpha y) = |\alpha| d(x, y)$) and can be related back to the norm by $\forall_x d(x) = d(x, 0)$.

There exist various relaxations of these properties, of which we will give some of the more popular variations here. Note that these terms are not used consistently throughout literature. In data mining, we usually want to use a relaxed version: in particular Identity of Indiscernibles will generally not hold if we allow duplicate records in the database. So technically, we will usually only have a pseudo-metric on our raw data. However, any pseudo metric induces a metric on the equivalence classes, and this will usually be sufficient for our needs.

However, there exist numerous useful dissimilarity measures in data mining that do not satisfy the triangle inequality, yet these are also commonly called “distance functions”. Metricness is a desirable mathematical property, but will often not hold in practical use. Figure 2.2 shows an example where the human perception of similarity will not satisfy the triangular inequality. The first two images – labeled as images x and y are seemingly unrelated, as the first image shows a boat on water, the second image buildings. However, when presented the context in

(a) First image x


(b) Second image y

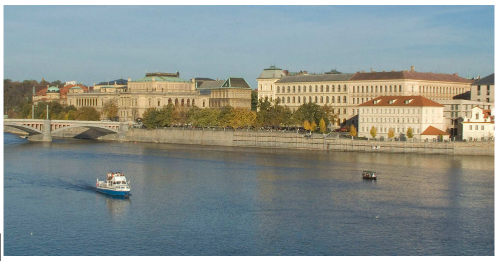
[tb]
(c) Context image z, containing both x and y.

Figure 2.2: When the "middle" image z provides context, the triangle inequality may not hold in perceived distances, resulting in $d(x,z) + d(z,y) \leq d(x,z)$.
Image licensed CC BY-SA 2.0 by Flickr user Ricardo Liberato

Table 2.1: Terminology of distance functions summarized.

	Non-Negative	Identity of Indiscernibles	Symmetry	Triangle Inequality	Homogenity
Dissimilarity function	+				
(Symmetric) Pre-metric	+		+		
Semi-metric, Ultra-metric	+	+	+		
Pseudo-metric	+		+	+	
Metric	+	+	+	+	
Norm, Homogenous metric	+	+	+	+	+

form of an Image z, it becomes apparent that both are part of the same scene and thus actually closely related to each other. This can be interepreted as a violation of the triangle inequality, where image z provides a "shortcut" connection between images x and y. Similarly, noise robust distances will typically violate the triangle inequality [VKG02]. Therefore, we will use the following terminology (summarized in Table 2.1):

Distance Function: A distance function is a function $d : X \times X \to \mathbb{R}$ that is non-negative and symmetric as given above (i.e. a pre-metric).

(Pseudo-) Metric: A metric is a distance function $d : X \times X \to \mathbb{R}$ that additionally satisfies the triangle inequality as given above (many mathematical textbooks will refer to this as a pseudo-metric).

Norm: A norm is a function $d : X \to \mathbb{R}$ that satisfies all of the above properties, and thus induces a homogenous metric by $d(x, y) = d(x - y)$, however often with relaxed identity of indiscernibles.

2.2.1 Minkowski L_p-Norms

The L_p family of norms includes some of the most popular metrics, in particular the Euclidean norm (L_2, or “as the crow flies”) and the Manhattan norm (L_1, also known as taxi cab or city block metric). These metrics are commonly given by the equation

$$L_p(x,y) = \left(\sum_i |x_i - y_i|^p \right)^{1/p},$$

such that in particular $p = 2$ and $p = 1$ yield the formulas:

$$L_2(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \qquad \text{(Euclidean distance)}$$

$$L_1(x,y) = \sum_i |x_i - y_i|. \qquad \text{(Manhattan distance)}$$

However, as the name indicates, these metrics are induced by a norm as follows:

$$L_p(x,y) := L_p(x-y)$$

$$L_p(v) := \left(\sum_i |v_i|^p \right)^{1/p},$$

where intuitively the norm $L_p(v)$ measures the *length* of the vector v. The relationship between L_p-norms and the generalized means M_p is now hard to miss, as obviously L_p-norms and M_p means differ only by a dimensionality-dependent constant

$$L_p(v) = \frac{1}{d^{1/p}} M_p(|v_1|, \ldots, |v_d|).$$

In particular, Euclidean distance is closely related to the quadratic mean (root mean square), while Manhattan distance relates to the arithmetic mean.

Note that L_p is only a norm (respective metric) for $p \geq 1$. For $p < 1$, the function will no longer satisfy the triangle inequality; they do however still yield a distance function as defined above (i.e. a dissimilarity function in the common mathematical sense) [AHK01; FWV07].

2.2.2 Distance Functions Induced by Means

Similar to the way a norm induces a metric, any non-negative symmetric function f (i.e. with $\forall_x f(x) > 0 \wedge f(x) = f(-x)$) can be used to induce a distance function (dissimilarity function) by using $f(x, y) := f(x-y)$. If f additionally satisfies subadditivity $\forall_{x,y} f(x+y) \leq f(x)+f(y)$, then the induced distance function will be a metric (pseudo-metric).

This allows the canonical construction of additional distance functions. Means are a particularly interesting source of such functions, for example we can construct a "trimmed Euclidean distance" by using $f = \text{trimmed-mean}_{2,\alpha}$ or a "median distance". We will not be using these in the context of this thesis, but the strong tie between L_p-norms and generalized means will play a role.

2.2.3 Weighted Distance Functions

Distance functions such as the Minkowski family are based on the implicit assumption that all dimensions have the same scale and magnitude. When the coordinates describe a physical position, or are measurements on the same scale, this assumption is a sensible choice. However, in many situations, the dimensions may differ fundamentally. For these reasons it is a best practice to rescale the data in preprocessing, for example to unit range or unit variance. Rescaling the data with a factor of ω_i on each dimension i is equivalent to weighting the dimensions with ω_i^p in Minkowski norms:

$$L_{p,\Omega}(x, y) = \left(\sum_i (\omega_i |x_i - y_i|)^p\right)^{1/p} = \left(\sum_i \omega_i^p \cdot |x_i - y_i|^p\right)^{1/p}.$$

However, when an axis is not linearly scaled, more complex transformations in preprocessing are needed. An interesting and often effective – albeit heuristic – approach are locally weighted distances such as Bray-Curtis, Kulczynski-1, Canberra and Clark distances:

$$\begin{aligned}
\text{Bray-Curtis}(x, y) &= \frac{\sum_i |x_i - y_i|}{\sum_i x_i + y_i} \\
\text{Kulczynski-1}(x, y) &= \frac{\sum_i |x_i - y_i|}{\sum_i \min\{x_i, y_i\}} \\
\text{Canberra}(x, y) &= \sum_i \frac{|x_i - y_i|}{|x_i| + |y_i|} \\
\text{Clark}(x, y) &= \sqrt{\frac{1}{n} \sum_i \left(\frac{|x_i - y_i|}{|x_i| + |y_i|}\right)^2}
\end{aligned}$$

All three of these distances can be interpreted as weighted variations of the Manhattan metric. Bray-Curtis (which is only sensible for non-negative data) rescales Manhattan distance by the inverse length of the input vectors. On normalized histograms with $\sum_i x_i = 1$, Bray-Curtis (closely related to the Sørensen–Dice coefficient [Sør48; Dic45] and Hellinger distance [Hel09]) and Manhattan distance are equivalent except for a constant factor. Kulczynski similarity 1 can also be interpreted as distance relative to the magnitude of the input vectors. It has links to set theory, where $\sum \min x_i, y_i \equiv |X \cap Y|$. The distance can thus be seen as normalized with respect to the length of the intersection vector. Canberra distance weights each dimension with the inverse of the absolute length of the input data. This yields the interesting property that $d(x, y) = d(2x, 2y)$, making the distance function less dependent on the local scale of the data. Clark distance relates to Euclidean distance essentially the same way Canberra relates to Manhattan distance: it is the M_2 mean of the same one-dimensional differences that Canberra is the sum of.

Many more distance measures can be found in the book "Dictionary of Distances" [DD06].

2.3 Kernel Density Estimation

Kernel density estimation (KDE) is a statistical technique for estimating the density of a data set, given 1-dimensional data samples $X = \{x_1, \ldots x_n\}$. It is commonly formalized as

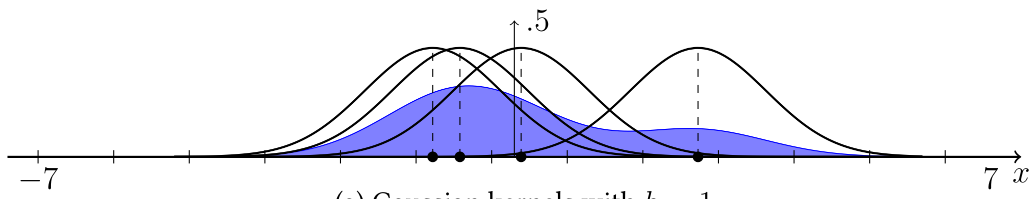
$$f(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i) = \text{mean}\left\{K(x - x_i) \mid x_i \in X\right\},$$

where K is a kernel function with the property that $\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1$ and $\forall_x K(x) \geq 0$.

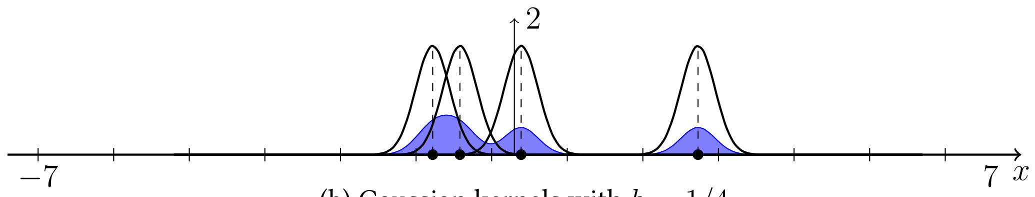
In many situations, the kernel will be rescaled by a bandwidth h using $K_h(x) = \frac{1}{h}K(x/h)$, and we may want to allow kernels with different bandwidth h_i for each object x_i to get an adaptive kernel density estimation [TS92]:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i} K\left(\frac{x - x_i}{h_i}\right) = \text{mean}\left\{\frac{1}{h_i} K\left(\frac{x - x_i}{h_i}\right) \mid x_i \in X\right\}$$

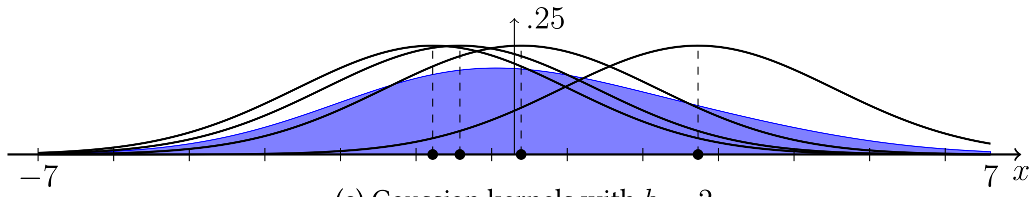
The bandwidth controls the smoothing: a larger bandwidth will produce a smoother density estimation; a smaller bandwidth will have more detail. There are some heuristics for choosing the bandwidth, such as Silverman's rule [Sil86], but it remains a data set dependent parameter, and "choosing a good value for the bandwidth, h, is the most difficult task" [SS05]. Figure 2.3 visualizes the basic idea of kernel density estimation using Gaussian kernels of different (but constant) width and the influence on the result. A copy of the kernel function is centered at each observation (black lines). The total density is then obtained by taking the average of the individual densities (blue area). The problems of choosing the optimal kernel bandwidth h can be seen here: Figure 2.3c is oversmoothed, the bandwidth has been chosen too large. But neither Figure 2.3a nor Figure 2.3b is obviously superior to the other.



(a) Gaussian kernels with $h = 1$

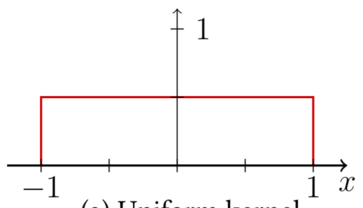


(b) Gaussian kernels with $h = 1/4$

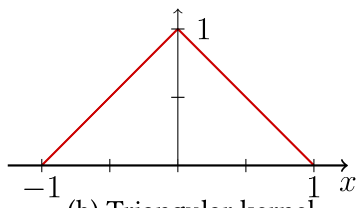


(c) Gaussian kernels with $h = 2$

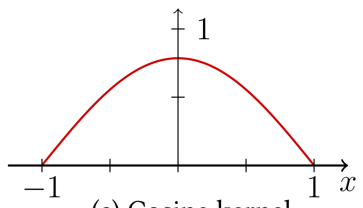
Figure 2.3: Kernel density estimation with Gaussian kernel: the combined density estimate (blue area) is obtained by taking the average of the individual kernel functions (black lines). Different kernel bandwidths h result in different smoothing.



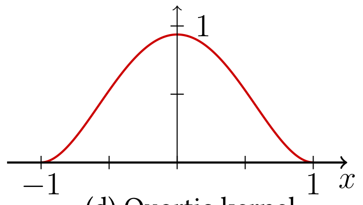
(a) Uniform kernel



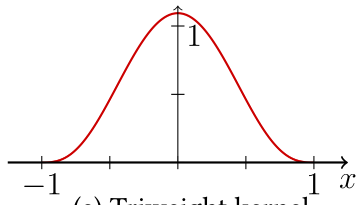
(b) Triangular kernel



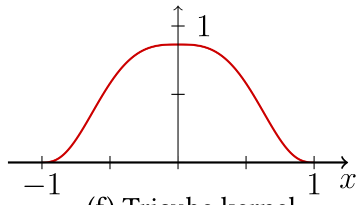
(c) Cosine kernel



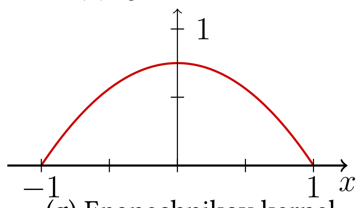
(d) Quartic kernel



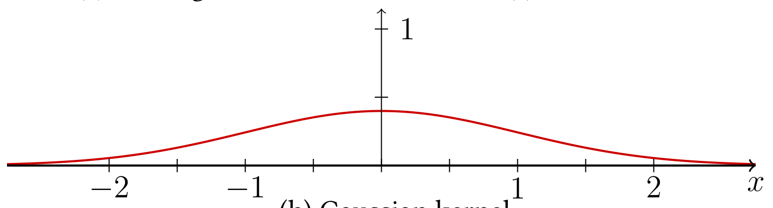
(e) Triweight kernel



(f) Tricube kernel



(g) Epanechnikov kernel



(h) Gaussian kernel

Figure 2.4: Different popular kernel functions. Kernels (a) to (g) have finite support, while kernel (h) has infinite support. Each kernel is normalized to $\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1$.

2.3.1 Popular Kernel Functions

There are two major classes of kernels: kernels with infinite support (such as the Gaussian kernel) and kernels with finite support (such as Epanechnikov kernel). From a mathematical perspective, kernels with infinite support are often cleaner, from a computational point of view a finite support allows for faster computation, as we can skip elements outside a distance of 1. Figure 2.4 visualizes the most popular kernel functions.

2.3.2 Multidimensional Kernel Density Estimation

In density estimation, kernels are usually symmetric, in contrast to for example edge detection kernels in image processing. For multivariate data, the common restriction is to use a radially symmetric kernel, which can be obtained by using the one-dimensional kernel functions and substituting the deviation with the Euclidean distance, or in the more general case, using Mahalanobis distance.

Given a positive definite symmetric bandwidth matrix H, multivariate kernel density is performed using the equation

$$f(x) = \frac{1}{|H^{1/2}|n} \sum_{i=1}^{n} H^{-1/2} K(x - x_i).$$

In the simplest case, when the bandwidth in each dimension is h, i.e. $H = h^2 \cdot I$, then this equation simplifies to

$$f(x) = \frac{1}{\sqrt{d}h^d n} \sum_{i=1}^{n} K(\frac{x - x_i}{h}).$$

The bandwidth suggested by Silverman's rule [Sil86] is given by

$$h_{ii} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{-\frac{1}{d+4}} \sigma_i,$$

where n is the data set size, d the dimensionality and σ_i the standard deviation in dimension i. Outside of the diagonal, $h_{ij} = 0$ for $i \neq j$.

Scott and Sain [SS05] give a good overview of techniques to improve the performance of kernel density estimation, for estimating full bandwidth matrices, and for the challenges associated with kernel density estimation. In particular for high-dimensional data, the typical challenges arise: the bandwidth matrix has d^2 degrees of freedom, so a large data sample is needed to reliably determine the bandwidth matrix. Scott and Sain [SS05] state they consider direct density estimation feasible "in as many as six dimensions". Beyond this, the results match observations we will discuss in Chapter 4 in a broader context: while the quality of the actual estimated density degrades in high dimensionality, the ranking obtained from the density estimates can sometimes still be useful for discrimination.

3 Related Work

This section is in large parts based on related work summaries used in:

> A. Zimek, E. Schubert, and H.-P. Kriegel. "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: `10.1002/sam.11161`
>
> A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 12th International Conference on Data Mining (ICDM), Brussels, Belgium. 2012. DOI: `10.1109/ICDM.2012.9`
>
> A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Gold Coast, Australia. 2013
>
> E. Schubert, A. Zimek, and H.-P. Kriegel. "Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection". In: *Data Mining and Knowledge Discovery* (2012). DOI: `10.1007/s10618-012-0300-z`

3.1 Distance-Based Outlier Detection in Databases

The work on distance-based outliers (DB-Outlier) [KN97b; KN98; KNT00] is probably the first database oriented outlier definition. It uses two parameters, a radius ε and a fraction π (plus the implicit parameter of a distance function d). By the DB-Outlier definitions, an object p of the database D is a DB-outlier if and only if

$$|\{o \in D \mid d(o,p) < \varepsilon\}| \big/ |D| \leq \pi.$$

This definition yields a binary decision of outlierness. However, the definition can be schönfinkeled (curried) in two ways to obtain an outlier score: either the threshold π, or the radius ε at which this inequality becomes true, can be used as outlier score. Both versions can be seen as generalizations of DB-Outlier that need one parameter less, and that produce scores instead of a binary decision. The first version can also trivially be rewritten as a density estimation using the uniform kernel:

$$\varepsilon\text{-density}(p) := |\{o \in D \mid d(o,p) < \varepsilon\}| \big/ |D| = \text{mean}_{o \in D}\left\{K_{\text{Uniform}}\left(d(o,p)/\varepsilon\right)\right\}. \tag{3.1}$$

On the other hand, we can compute the radius at which there are at least k neighbors:

$$k\text{-dist}(p) :=\text{argmin}_\varepsilon |\{o \in D \mid d(o,p) < \varepsilon\}| \geq k \tag{3.2}$$

Neither is identical to kernel density estimation, but both are closely related. The reciprocal value $1/k\text{-dist}(p)$ is a naïve estimate. This distance is also used in one of the common definitions of the k nearest neighbors set (which may contain more than k elements if they have the same distance):

$$k\text{NN}_d(p) := \{o \in D \mid d(o,p) \leq k\text{-dist}(p)\} \tag{3.3}$$

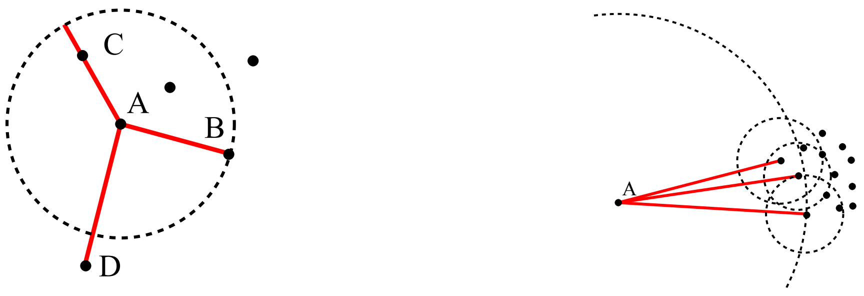
In k nearest neighbor outlier detection (kNN-Outlier, [RRS00]), this k-dist is used as outlier score. It is also seen as an intermediate result in LOF [Bre+00], while kNN-Weight [AP02] modifies this concept slightly to instead use the sum of these distances as outlier score:[1]

$$k\text{NN-Weight}(p) := \sum_{o \in k\text{NN}'_d(p)} d(o,p) \tag{3.4}$$

Again, we can interpret $k/k\text{-weight}(p)$ as a density estimation. Since this estimation takes not only the kth, but all of the kNN into account, we can assume this value to be more stable.

Given the omnipresence of density estimation in outlier detection, it is obvious to try to use more advanced density estimators from statistics, in particular kernel density estimation (KDE, see Section 2.3). However, this yields additional challenges: kernel density estimation requires choosing additional parameters such as an appropriate kernel function (e.g. Gaussian or Epanechnikov) and the kernel bandwidth matrixes [SS05]. For simplicity, often only a diagonal matrix or a single bandwidth (a radially symmetric kernel) is used to simplify this process that has been called "the most difficult task" [SS05] in KDE. Kernels with infinite support such as the Gaussian kernel will increase the computation for exact computation to a quadratic runtime, unless the kernel is truncated at some threshold. There exist kernel density estimation methods that use the k-dist to choose the kernel bandwidth, often referred to as "sample point" and "balloon" estimators [TS92]. Multivariate density estimation is a fundamental statistical principle, and has as such received more attention in the domain of statistics than in the data mining community; including locally adaptive kernel density estimation such as the balloon estimator discussed before. For an overview of the statistics of density estimation, refer to e.g. [Sim96; WJ94]. In data mining, kernel density estimation has recently [LLP07] started to receive interest to be used for local outlier detection, and will be detailed in the next Section 3.2, Section 5.1.3 and Section 9.1.

[1]Actually for this formula to behave as intended, we need to use a modified neighborhood with $|k\text{NN}'(p)| = k$ for all p, which will be discussed and formalized in Chapter 6.



(a) Reachability distance: C and B are in the core of A and thus have the same reachability, while D is further away.

(b) LOF idea: by comparing the density of one point to the densities of neighbor points, local outliers can be detected.

Figure 3.1: Two core concepts of LOF: reachability distance and local reachability density.[2]

3.2 Local Outlier Detection

The first method to discuss "locality" in outlier detection was LOF [Bre+00]. This method is motivated by the observation that different regions of a data set may exhibit a different level of density – and thus using a naïve global density estimation may not be adequate. Instead, we need a score that adapts to the "local" properties and distribution of the data set.

LOF can be seen as a multi-step outlier detection method. While in kNN-Outlier(Equation 3.2, [RRS00]) you can compute the outlier score of an object directly from the distances, LOF will use also the neighbors of the neighbors. For performance reasons (to avoid recomputations), it is desirable to compute the steps one after another and to cache the results as discussed in Chapter 6.

The first step of LOF is again a density estimation. This estimation is based on ideas of clustering, in particular the methods DBSCAN [Est+96] and OPTICS [Ank+99] and the notion of a "core" in these methods. The size of the core of an object o is its $k\text{-dist}(o)$. If the object p is within the core of o, the direct distance is replaced with the $k\text{-dist}(o)$. Figure 3.1a visualizes the reachability distance for an object A. Objects B and C are core reachable, while D is outside of the core. This modified distance is called "reachability distance":

$$\text{reach-dist}_k(p \leftarrow o) = \max\{k\text{-dist}(o), d(o,p)\} \tag{3.5}$$

The effects of this distance will be analyzed in Chapter 5 in more detail. Since reachability is asymmetric, we prefer a notion that emphasizes the asymmetry by using an arrow $\leftarrow$: it is the directed reachability of p *from* o.

Based on this distance, LOF defines the "local reachability density" as the inverse average reachability distance of the object from its neighbors.

$$\text{lrd}_k(p) = 1 / \left(\text{mean}_{o \in k\text{NN}(p)} \text{reach-dist}_k(p \leftarrow o)\right) \tag{3.6}$$

[2]These images are original work, but were also contributed to the Wikipedia LOF article.

Finally, the density of an object is compared to the average density of its neighbors to obtain a "local" score. This idea, sketched in Figure 3.1b, is the main idea of LOF and inspired various other outlier detection algorithms.

$$\mathrm{LOF}_k(o) = \mathrm{mean}_{o \in k\mathrm{NN}(p)} \frac{\mathrm{lrd}_\mathrm{k}(o)}{\mathrm{lrd}_\mathrm{k}(p)} \equiv \frac{\mathrm{mean}_{o \in k\mathrm{NN}(p)} \mathrm{lrd}_\mathrm{k}(o)}{\mathrm{lrd}_\mathrm{k}(p)} \tag{3.7}$$

For inliers, a value around 1 (or below) is expected, while outliers will have higher values, based on the intuition that outliers are less dense than their neighbors.

3.3 Basic LOF Variations

Over time, the LOF method inspired a number of variations: Chiu and Fu [CF03] define the variants LOF' which removes the reachability distance from LOF (this will be discussed in more detail in Section 5.1.1), and LOF", which allows a different value of k to be specified for the comparison step. Local Outlier Correlation Integral (LOCI) [Pap+03] performs similar density estimates and comparisons at multiple different radii instead of using a single value of k. Influenced Outlierness Factor (INFLO) [Jin+06] varies LOF by looking at the intersection of the $k\mathrm{NN}$ and the reverse k nearest neighbors (for a formalization, see Chapter 6). Pei et al. [PZG06] use approximated k nearest neighbors based on reference points. Local Density Factor (LDF) [LLP07] computes LOF scores based on kernel density estimation. Local Distance-Based Outlier Factor (LDOF) [ZHJ09] compares the $k\mathrm{NN}$ distances to the pairwise distances of the neighbors instead. Local Outlier Probabilities (LoOP) [Kri+09a] tries to improve the robustness of LOF with statistical measures (this will be discussed in more detail in Section 5.1.1) as well as normalizing the scores to the range of $[0, 1]$ (which will be discussed in Section 5.3.2).

3.4 Local Outlier Detection in High-Dimensional Data

Probably the first (although not local) approach for outlier detection in high-dimensional data was the subspace oriented method by Aggarwal and Yu [AY01]. It is a grid-based approach that uses an evolutionary search strategy to find unusually sparse subspaces. The data is first divided into ϕ equi-depth ranges in each dimension so that each partition contains approximately $f = 1/\phi$ of the total objects. By intersecting the one-dimensional grids in k dimensions, the expected number of objects in such a hyper-cube then is $N \cdot f^k$ with a standard deviation of $\sqrt{N \cdot f^k \cdot (1 - f^k)}$. Any object in a hypercube that contains significantly fewer objects than expected is considered an outlier by this algorithm.

This method however has various problems, which will be discussed in more detail in Chapter 4. First of all, the expected values themselves become tiny for high-dimensional data very quickly, in particular when ϕ or k are not very small either, making detection with this statistic impossible. Assuming just 10 grid cells and 6 dimensional subspaces, the expected number of objects is 1 out of 1 million, i.e. we need a data set containing several million objects to make this approach feasible. Furthermore, scores for different dimensionality k are not comparable, so this approach only works for finding outliers in a single subspace dimensionality k. The proposed evolutionary search therefore focuses on preserving k, but since it does not have a very good control function (in contrast to clusters, outliers will often not surface gradually), the search for appropriate subspaces is not very effective but more or less just randomized. But probably the worst problem of this method is that due to the equi-depth partitioning strategy (which is required for the estimation of the number of objects in each cell), the method is very likely to put outliers into the same grid cell as a nearby cluster. Last but not least, the presence of clusters with different sizes is not really taken into account in the statistical test, which in fact assumes the data to be uniformly distributed except for the outlier grid cells.

An interesting approach for high-dimensional data is angle-based outlier detection (ABOD) introduced in [KSZ08], because it does not pay explicit attention to the dimensionality. In contrast to many other methods, it does not rely on a density estimation, but looks at the variance of the angular spectrum instead. For performance reasons, it can optionally use only the $k\mathrm{NN}$ as a comparison set, which makes FastABOD a local method. Recent improvements on ABOD include a $\mathcal{O}(n \log n)$ approximation [PP12].

HOS-Miner [Zha+04] tries to approach the subspace search from the other direction: for each object, it tries to find the subspace where it appears to be most unusual. In order to perform an Apriori-like search, they exploit that the knn weight (Equation 3.4) outlier scores increase monotonically with increasing dimensionality. However, the authors neglect the fact that this outlier score is not comparable across different dimensionalities because of the very same monotonicity (as also noted in [NGA11]): in particular, the monotonictiy implies that the maximum score will always be found in the full dimensional space.

OutRank [Ass+07b; Mül+08; Mül+12] avoids the data snooping bias (see Chapter 4) elegantly since the method searches for subspace clusters, instead of looking for outliers. These should be much easier to find in high-dimensional data, as they are not rare objects and can be expected to be recognizable even if the subspace is not yet optimal (of course, subspace clustering does still pose a number of nontrivial challenges). Such clusters can be found for example using DUSC (dimensionality unbiased subspace clustering, [Ass+07a]) or EDSC (efficient density-based subspace clustering, [Ass+08]). The outlierness of an object is then estimated by the number of times the object is contained in such subspace clusters as well as the dimensionality and size of the clusters. This method however relies on the subspace algorithms to produce a highly redundant clustering (subspace clusters are commonly allowed to overlap). Furthermore, outliers can only be found when the data set clusters well – on a data set where no clusters are found, all objects will be considered outliers. But even a good density-based clustering result will often already contain a large number of unclustered objects, so that the methods mostly discover outliers that are the least often contained in clusters. On the other hand, clustering algorithms are usually not designed to exclude outliers, but will often include outliers in a nearby cluster. Furthermore, there is little insight or control over which types of outliers are detected, as they are merely a statistical side product of multiple clusterings.

Subspace Outlier Degree (SOD) [Kri+09b] is an outlier detection method for high-dimensional data. Instead of using the kNN, it uses the more robust concept of shared nearest neighbors (SNN, see Section 4.4), and it chooses a subspace to compute the similarities in. It will be discussed in more detail in Section 5.2.1.

OutRES [MSS10] is a density-based subspace method. For every object, it locates the neighbors in different subspaces and compares their kernel densities. It tries to take the different dimensionalities into account by adjusting kernel size and distances depending on the dimensionality. In order not to have to compute densities in all $2^d - 1$ subspaces, an Apriori style search strategy is employed, combined with a test against uniform distribution.

HiCS (High-contrast subspaces for density based outlier ranking, [KMB12]) can be seen as a meta outlier detection method. It will first identify "high contrast" subspaces; then run an outlier detection method such as LOF in these subspaces. This allows the use of LOF in high-dimensional data it could normally not process. However, the scores are not normalized across different dimensionalities. Depending on the data set, HiCS may find an excessive amount of subspace combinations and the subspace search dominates the total running time.

Correlation Outlier Probabilities (COP) [Zim08; Kri+12] – which will be covered in detail in Section 5.2.2 – try to identify outliers nearby arbitrarily oriented correlation clusters by the deviation from a local trend.

3.5 Efficient Approaches for Local Outlier Detection

ALOCI [Pap+03] is an efficient approximation of LOCI using multiple quadtrees for efficient density estimation. The kNN are no longer used, so one may consider this method to be not as local as before. Instead, quadtree density estimates at different radii are compared. Since these density estimates are best when an object is close to the center of a tree cell, additional alternate quadtrees are produced by cyclic wrapping the data set, which may prevent the detection of outliers on one side of the original space if there is a cluster on the opposite side.

The kNN-weight approach [AP02; AP05] includes an efficient top-n algorithm HilOut using Hilbert space filling curves for L_p-norms only. Due to the curse of dimensionality (see Chapter 4) it however does not scale well to high-dimensional data, as the pruning rules will not work with the loss of contrast caused by the high dimensionality (the top-n element will only have marginally higher scores than the next, so a large part of the data set will have to be analyzed in each pass).

In [WPT11], the authors propose a locality sensitive hashing (LSH) based approach to rank objects by their probability to be in a low-density region. These objects can then be explored first, and will allow pruning dense objects earlier. However, this approach is only benefitial when searching the top-n outliers with respect to a global method such as kNN outlier only. For local outlier detection, the hash tables cannot provide useful pruning information.

Most kNN-based approaches can be accelerated by using an index structure that allows accelerated kNN queries, such as the R*-tree [Bec+90] or the k-d tree [Ben75]. Chapter 8 will discuss both space filling curve, LSH and R*-tree based acceleration of outlier detection.

3.6 Ensemble Methods for Outlier Detection

Feature Bagging for Outlier Detection [LK05] is the canonical adaptation of the original bagging (bootstrap aggregating) technique [Bre96] developed in classification and feature selection. Instead of performing outlier detection in the full dimensional space, a random subset of dimensions (of $[\lfloor d/2 \rfloor; d-1]$ dimensions) is chosen, and the resulting ranks or scores are combined. The authors overlooked that the scores across different dimensionality are not always comparable. Using a fixed subspace dimensionality may turn out to work better in practice.

Feature Bagging for outlier detection [GT06] tries to improve ensembles by fitting the scores to a sigmoid curve or using a mixture model. However, the method essentially tries to maximize the contrast, often turning the scores into a binary 0 or 1 decision, at which point the ensemble degenerates to counting the number of times an object was classified as outlier, or into building the union of all found objects. Furthermore, the score optimization did not show to be numerically stable in practice, and would often produce degenerated results.

The method HeDES [NAG10] improves the earlier feature bagging ensemble approach by also including the results obtained on categorical attributes and by performing a simple score normalization (to zero mean unit variance) before combining the resulting scores.

Later research [Kri+11] on the normalization of outlier scores showed that normalization plays an important role for outlier detection ensembles. The normalization of scores will be discussed in detail in Section 5.3. The role of diversity was further studied in [Sch+12], where random ensembles were pruned while trying to maximize diversity. This will also be discussed in detail in Chapter 7.

3.7 Community Outlier Detection

A different notion of outliers exists in graph structured data. The notion of density does not easily translate to e.g. social networks: it has been shown that the degrees of nodes are usually Zipfian distributed. This means there will be a lot of nodes with few edges (which would be intuitively of “low density”) and only very few nodes with a very high node degree (sometimes referred to as “hubs” or “social stars”). One can further not assume that there is a gradual drop-off: such low-degree nodes will often be directly connected to these hubs. While the hub nodes are interesting for many applications, they are also rather easy to find, because such graphs are usually stored with adjacency lists, that allow trivial access to the node degree.

What then can make a node in a graph anomalous? CODA [Gao+10] is an outlier detection algorithm for communities that analyzes two different graphs at the same time. Objects for which the two graphs do not align well are considered outliers. For example, for a set of conferences, one graph may resemble the topics at the conferences, and the other graph the authors. Usually, the authors and topics align very well. However, there exist some conferences that span multiple communities, both with respect to authors and with respect to topics.

This notion of outliers can be detected with the generalized notion of local outliers as discussed in Chapter 6, and detailed in Section 6.5.

3.8 Geographic Outlier Detection

In particular the recent years, with the widespread availability of cheap GPS in mobile phones, but also due to large scale sensor deployment, we have seen an increasing amount of data that is geo-referenced. For example many photos are now annotated with the spatio-temporal position where the photo was taken.

Yet, the detection of outliers in such data sets reaches back way beyond data mining into the domain of geostatistics [Mor50; Gea54]. Spatial autocorrelation measures such as LISA (Local Indicators for Spatial Association, [Ans95]) are examples of such early approaches of measuring spatial outlierness. A key result of it is the generalization of the global spatial association statistics Moran's I [Mor50] and Geary's C [Gea54] to individual contributions denoted as the local Moran I_i and local Geary C_i; then using a statistical test to identify strong contributions. Additionally the Moran scatterplot was introduced, which plots the locally z-standardized attribute value against the globally z-standardized attribute. In this plot, objects close to the regression line indicate consistency with the trend, whereas objects in the upper left and bottom right areas appear different on local and global scales. This concept of comparing local with global scores can be found in many newer methods in slight variations; others however just use the local scores proposed here directly or with only slight modifications. Spatial outlier detection has since grown as a field of its own interest over several years [SLZ03; LCK03; KLC06; SC04; CS06; LLC10; CLB10].

The key idea is to separate spatial attributes from other attributes, compute the neighborhood w.r.t. the spatial attributes solely but compare the non-spatial attributes only to derive a notion of outlierness. Most of these methods use a local neighborhood based on the spatial attributes just in order to extract a score (the simplest type of model) for the object using the non-spatial attributes only. Many methods can just process a single non-spatial attribute, and there are rather few methods that use a model more complex than a preliminary score or a non-trivial comparison step. However, we will highlight some examples in Chapter 6, when we generalize the concept of outlier detection to such more general data domains. In Chapter 9 we will also see two case studies detecting special kinds of outliers in geographic and geo-spatial data.

4 High-Dimensional Data and the "Curse of Dimensionality"

"Our brains have evolved to get us out of the rain, find where the berries are, and keep us from getting killed. Our brains did not evolve to help us grasp really large numbers or to look at things in a hundred thousand dimensions.
— *Ronald L. Graham* [Hof87]"

"In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from earliest days.
— *Richard Bellman* [Bel61]"

The term "curse of dimensionality" is often used as an umbrella term for all kinds of unexpected effects happening at high dimensionality. While these effects likely have the same reason, they can manifest in very different ways, from a loss of contrast in distance computations to what is called "combinatorial explosion".

Much of our intuition, which is driven by the 3-dimensional world we perceive every day, does no longer hold in complex data spaces. Yet, these spaces are very real when we use mathematics to model not the physical world, but properties such as color and texture of images, or text. Mathematically, a grayscale computer image of 10 by 10 pixels can trivially be represented by a 100-dimensional vector. While this is not be the best representation of the meaning of the image, we must acknowledge that there exists data beyond the 2 and 3 dimensions we use for measuring physical location: in the world of information, there exist plenty of examples for higher-dimensional data, such as images, audio, gene expression data or sensor networks.

In the wider sense of "big data" – commonly defined by "the 3 V's" [Lan01]: volume, velocity and variety – high-dimensional data can be seen as part of the variety challenge of big data. The omnipresence of sensors not only increases the mere volume of data, but also the dimensionality: a large set of low-dimensional sensors can be seen as a high-dimensional multivariate time series. Redundant sensors and increased temporal precision further increase redundancy in the data, which is currently often handled by preprocessing: averaging and downsampling the data to a more manageable size and dimensionality. Yet, one of the key visions of "big data" and "data driven science" is to have tools that can handle this kind of data *natively*. While there has been a lot of progress the last years on processing larger volumes of data, and data coming in at a high velocity, we have just begun to understand the challenges of high-dimensionality.

4.1 Manifestations of the "Curse of Dimensionality"

The reasons for the failing of our intuition are numerous, and this section provides an overview of the effects discussed in greater detail in the publications:

> A. Zimek, E. Schubert, and H.-P. Kriegel. "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: `10.1002/sam.11161`
>
> A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 12th International Conference on Data Mining (ICDM), Brussels, Belgium. 2012. DOI: `10.1109/ICDM.2012.9`
>
> A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Gold Coast, Australia. 2013

Combinatorial Explosion on Grids: In low-dimensional data, many problems can efficiently be handled with grid-based approaches. The data set is divided into a multidimensional grid of cells for analysis – either at a fixed distance, or quantile-based such that they contain the desired amount of objects. In many situations, an algorithm will not need to process all cells, but can either skip cells that contain very few or very many objects, or it may be possible to process all objects in a cell as if they were the same.

If we divide each dimension into just 10 bins, we get 10 cells in one dimension, 100 cells in 2 dimensions and 1000 in 3 dimensions. These numbers of bins are well manageable for computers. However, if we have a 100-dimensional data set, we get 10^{100} cells – the famous number "a googol" that inspired the name of the search engine Google. This number is way beyond what we can grasp. The visible universe's size is estimated to contain 10^{80} protons. So even if we had a database storing information on all the protons of the universe, we would have to expect the vast majority of our grid cells to be empty. However, to be able to draw statistically *significant* conclusions from this, we would need the grid cells to contain much much more objects than just approximately one in each cell. Yet, this number occurs in practice: assuming we had 10 shades of grey, there are 10^{100} different images of resolution 10×10 pixels.

Combinatorial Explosion on Subspaces: Even when we consider not to look at all d dimensions at the same time, we can run into this complexity problem. A 2-dimensional data set has 3 interesting projections (on "x", "y" and the original "x, y" space) – but at higher dimensions we get $2^d - 1$ possible projections. 2^{333} approximately is 10^{100}, so this does not get us much further.

This effect can have surprising consequences. For example, when using the classic statistical threshold of 3σ (see Section 1.2) in high-dimensional data one may run into what is called the "data snooping bias". Analyzing a single object in all dimensions, it will appear to be normal (by the 3σ rule) in each single dimension with a likelihood of 99.73%. However, the likelihood of the object to appear to be normal in every single dimension is 0.9973^d (without even considering other projections than single dimensions). For $d = \{10,\ 100,\ 1000\}$ this drops surprisingly fast: 97.33%, 76.31%, 6.69%. So at 1000 dimensions, we have a 93.71% chance of being able to find at least *one* dimension where the object is a 3σ significant outlier. This supports the informal – and misleading – claim that in high-dimensional data, "every point is an almost equally good outlier" ([AY01], but likely referring to the distance concentration effect below). However, one should make this claim more precise: "In high-dimensional data, every object has a high chance of being a 3σ outlier in at least one of the dimensions".

Concentration of Distances: The concentration of distances is a widely known effect of the "curse of dimensionality" in distance-based data mining. It can be formalized as: [Bey+99]

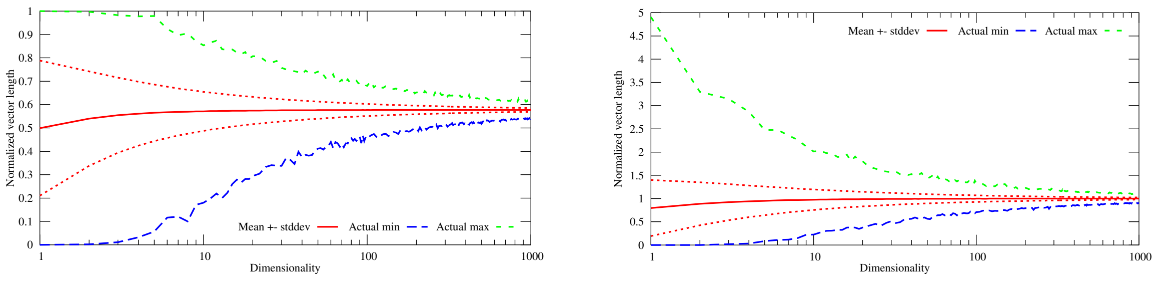
$$\text{If } \lim_{d\to\infty} \operatorname{Var}\left(\frac{\|X_d\|}{E[\|X_d\|]}\right) = 0, \text{ then } \frac{D_{\max} - D_{\min}}{D_{\min}} \to 0, \tag{4.1}$$

where $D_{\max}$ and $D_{\min}$ are the maximal and minimal distances in the data set. This *concentration effect* is also called the *loss of contrast* for distance functions. This effect can be observed in practice, has been studied experimentally, and has even theoretically been proven for a number of distance functions and data distributions [Bey+99; HAK00; AHK01]. Some distance functions (such as fractional L_p-norms[1] [AHK01; FWV07] and cosine dissimilarity [RNI10b]) appear to be slightly less susceptible, yet remain affected.

This loss of contrast makes certain algorithms such as the classic Knorr-Ng outliers [KN97a] and kNN-Outlier[RRS00] essentially meaningless: the radius parameter ε of DB-Outlier will be hard to choose precisely enough, and the k-distances will also no longer differ substantially across objects. This is probably the observation that caused Aggarwal and Yu [AY01] to claim "every point is an almost equally good outlier" for motivating their subspace approach. Yet, it again should be phrased more precisely: "In high-dimensional data, every object is an almost equally good kNN-Outlier."

So if the concentration of distances is *proven*, why would we still bother with high-dimensional data? First of all, the proofs are for $d \to \infty$, but in practice we will always deal with finite-dimensional data. Secondly, there is the notion of "intrinsic dimensionality" (see Section 4.3), based on the observation that data represented with d-dimensional vectors often behaves as if it only had $d' < d$ dimensions, and any d'-dimensional data can be inflated to $k \cdot d'$ dimensions without significantly affecting the results by duplicating dimensions or even simpler: by adding dimensions which are 0 for all objects. Third, the proofs only cover certain distance functions, and in particular second order distance functions such as the shared nearest neighbor distances (see Section 4.4) seem to offer some relief. And finally, and most importantly, the proofs assume

[1]Fractional L_p-norms (i.e. with $0 < p < 1$) are *not* mathematical norms, since they do *not* satisfy the triangle inequality. They however have the same formula as the proper L_p-norms with $p \geq 1$. See also Section 2.2.



(a) Uniform $\mathcal{U}[0;1]$ (b) Gaussian $\mathcal{N}[0;1]$

Figure 4.1: Normalized Euclidean length of vectors with increasing dimensionality. The x axis is logarithmically scaled.

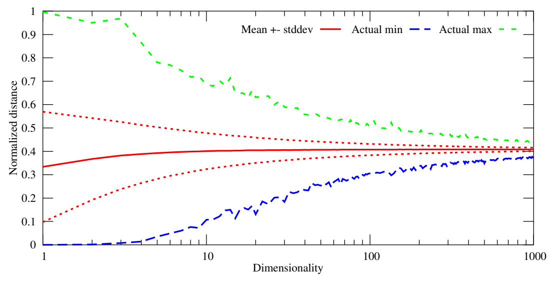
that the data is i.i.d. – independent and identically distributed –, a scenario in which any analysis would be a wasted effort anyway: In any scenario interesting for analysis we must assume that there exist different mechanisms such as clusters and outliers.

Hubness: A rather new and interesting observation is that if we look at the kNN of a large quantity of objects, a few elements occur more often in the kNNs of others, while the majority of objects occurs less often than one would expect [RNI09; RNI10a]. These rare objects – called “Hubs” – are however not classic outliers, but objects that can be seen as “unusually usual”, objects that apparently are more “central” than others. Their exact behavior, nature and meaning is subject to ongoing research, since they might be “Fact or Artifact” [Low+13].

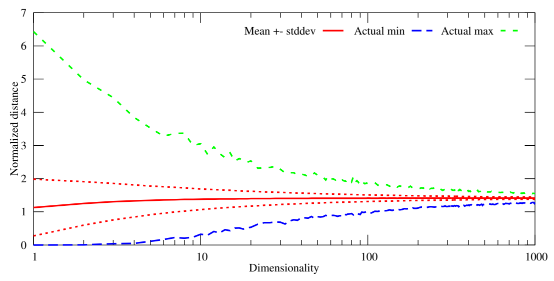
Given these different aspects of the curse, it remains an ongoing research question how to analyze high dimensional data. We will now discuss a number of experiments to study these theoretical effects in practice, how they affect the full-dimensional methods discussed earlier in Section 3.2, as well as approaches to remedy the effects to some extend.

4.2 Empirical Observations on High-Dimensional Data

In this section, we analyze the behavior not only of distances (reproducing the earlier results on distance concentration), but we also are interested to validate the effect on outlier scores. To make the plots easier to read, we do not plot the Euclidean distance, but we normalize it by the expected maximum length in the unit cube – the length of the diagonal, which is $\sqrt{d}$. This is appropriate also for the normal distributed data set, since the expected values grow with this factor as well. In the unit cube for example, the expected distance from the origin converges from 0.5 in one dimension to $\sqrt{d/3} \approx 0.5774\sqrt{d}$. In the multivariate standard normal distribution, the expected length converges to $\sqrt{d}$. We sampled 10000 objects from an uniform $U[0;1]$ distribution as well as a Gaussian $\mathcal{N}[0;1]$ distribution in up to 1000 dimensions. The dimensionality is drawn on the x axis, and we use a logarithmic scale to increase the visibility

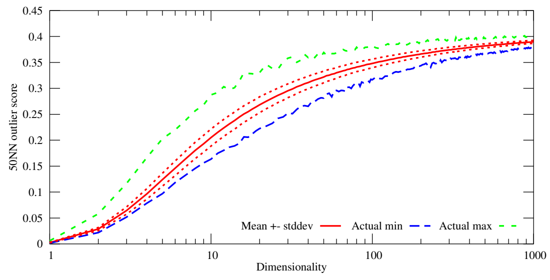


(a) Uniform $\mathcal{U}[0;1]$

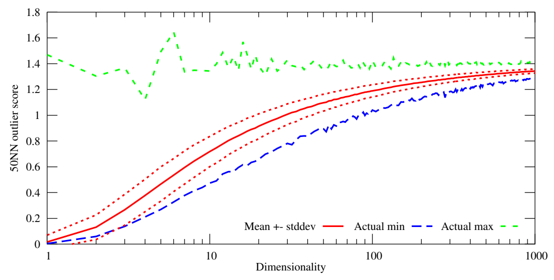


(b) Gaussian $\mathcal{N}[0;1]$

Figure 4.2: Normalized Euclidean pairwise distances with increasing dimensionality.



(a) Uniform $\mathcal{U}[0;1]$

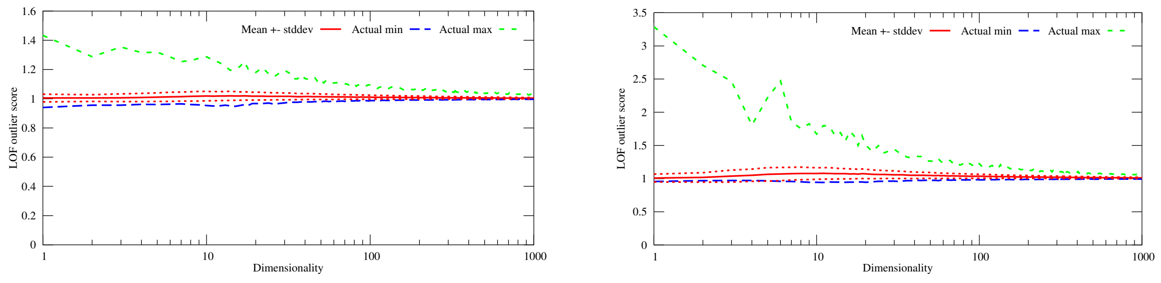


(b) Gaussian $\mathcal{N}[0;1]$

Figure 4.3: Normalized Euclidean 50NN distances with increasing dimensionality.

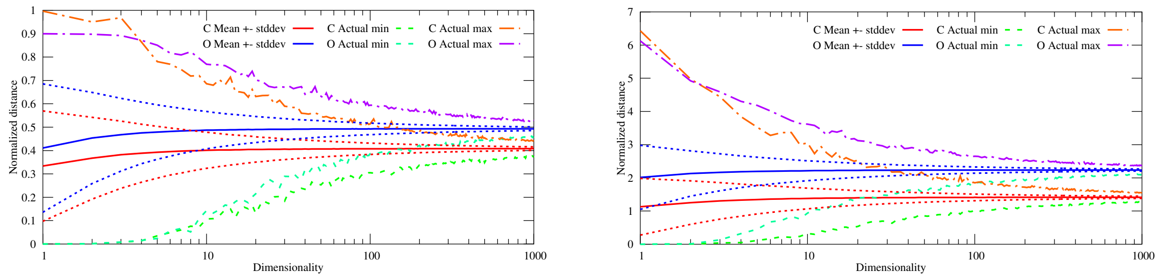
of the observed effects (the difference between 999 and 1000 dimensions can be expected to be marginal, and a logarithmic scale is visibly appropriate).

Independent and Identically Distributed Data (i.i.d.): Figure 4.1 shows the vector lengths after normalization. In low dimensionality, the observed minimum remains close to 0, and the maximum close to 1, while the mean converges to the predicted values (after normalization $\sqrt{1/3}$ respectively 1) with decreasing variance. However, at higher dimensionality, even the observed minimum and maximum distances begin to close in on the mean, with the normalized standard deviation approaching 0. So both of the data sets clearly exhibit the "distance concentration effect". In outlier detection, we however do not judge objects by their distance from 0, but by pairwise distances. In Figure 4.2 we can validate that this concentration still happens. One of the earlier distance-based outlier detection methods is based on the k nearest neighbor distance [RRS00]. Figure 4.3 plots this distance for $k = 50$. Here, the result may appear to be surprisingly different for uniform and Gaussian data at first. However, it is easy to see that uniform data in low dimensions does not show significant outliers, while as explained in the paragraph on the combinatorial explosion of subspaces, we actually can expect the Gaussian scores to converge against the one-dimensional maximum: as discussed there, almost every object will appear to be an outlier in at least one dimension! The reason that the uniform data converges for high dimensionality virtually the same way as the Gaussian distribution is the classic example of the central limit theorem. In fact any finite mean and variance i.i.d. data is ex-



(a) Uniform $\mathcal{U}[0;1]$ (b) Gaussian $\mathcal{N}[0;1]$

Figure 4.4: LOF $k = 50$ scores with increasing dimensionality.



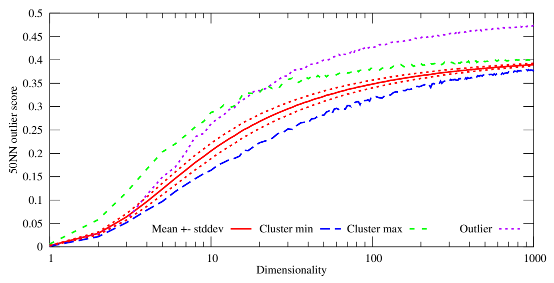
(a) Uniform $\mathcal{U}[0;1]$ (b) Gaussian $\mathcal{N}[0;1]$

Figure 4.5: Normalized Euclidean pairwise distances – with true outlier.

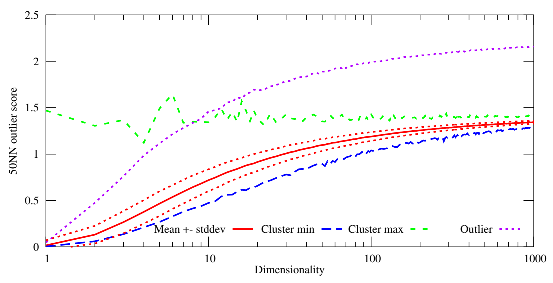
pected to converge this way. Finally, in Figure 4.4 we perform the same analysis for LOF scores (with $k = 50$). Unsurprisingly, due to the decreasing relative variance of the 50NN scores, the LOF scores quickly tend to 1. However, note that so far the data set was generated i.i.d., therefore it does actually not contain true outliers. Even the extreme values in the one-dimensional Gaussian distributions were generated by the same mechanism.

Different Mechanisms: For the next experiments, we add a second mechanism to the data set. By doing so, we remove one of the key assumptions for the proofs of the concentration of distances: our data set is no longer “i.i.d.”! Since we are interested in outlier detection, we use a very simple “outlier” mechanism: a single object placed at a fixed location. For the uniform data set we place the outlier at the constant value of 0.9, for the Gaussian distributed data at $2\sigma = 2$). Obviously, the outlier will be “normal” in every single dimension, and cannot be detected in one-dimensional data.

Figure 4.5 visualizes the pairwise distances again, but now with two different mechanisms. We therefore split the statistics into two groups: one consisting of the distances to the outlier object, and the other group containing the distances between two inlier objects. Since the outlier is constantly away from the center of the data set, it becomes more and more visible. At 1000 dimensions, the largest distance between inliers is substantially smaller than the smallest

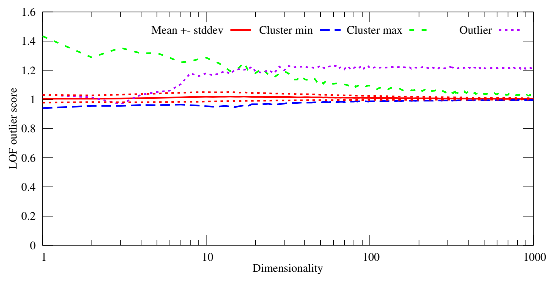


(a) Uniform $\mathcal{U}[0;1]$

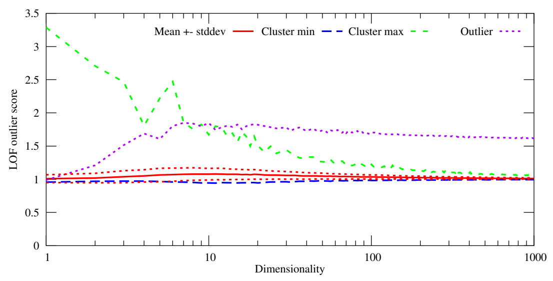


(b) Gaussian $\mathcal{N}[0;1]$

Figure 4.6: Normalized Euclidean 50NN distances – with true outlier.



(a) Uniform $\mathcal{U}[0;1]$



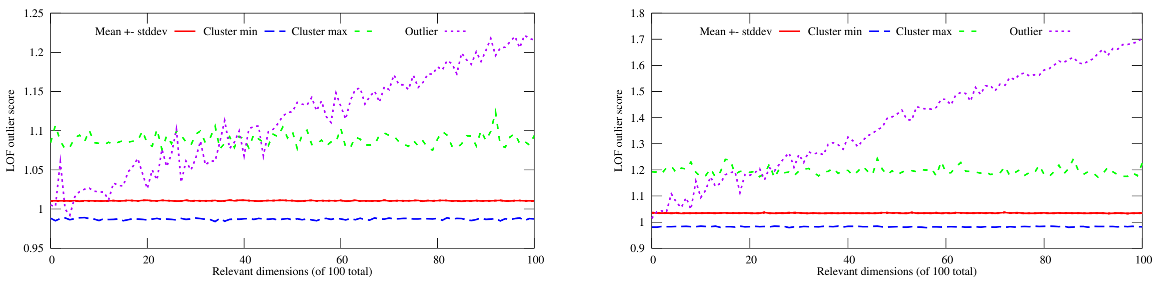
(b) Gaussian $\mathcal{N}[0;1]$

Figure 4.7: LOF $k = 50$ scores with increasing dimensionality – with true outlier.

distance of any inlier to the outlier.[2] Similarly, if we look at the 50NN outlier scores in Figure 4.6 or the LOF scores in Figure 4.7, the outlier quickly becomes well visible, with an outlier score deviating substantially even from the most extreme cluster element score.

Relevant and Irrelevant Attributes: However, in reality, we cannot rely on either of above situations to occur: neither will the data be i.i.d., nor will every single dimension increase the ability to differentiate the outliers from the clusters. And if we have a number of dimensions that are useful, and other dimensions that are i.i.d., then we need to investigate methods of recognizing these cases, in order to remove the useless dimensions and improve our "signal to noise" ratio. This is a primary motivation for investigating subspace methods: we need to identify discriminative dimensions and remove "noisy" dimensions. We do not need to get it all perfect: as long as we can keep most of the "good" dimensions and remove some of the "bad" dimensions, we should see an improvement in the results. Figure 4.8 visualizes the results of the experimental validation: we kept the number of dimensions fixed at $d = 100$, but varied the number of dimensions in which the outlier is i.i.d. to the inlier objects and in which it is set to the fixed position otherwise.

[2]The effect may appear to happen much earlier for Gaussian data, but this is due to the fact that 2σ was a more extreme choice than 0.9 for uniform data. We should have chosen $\approx 1.281551\sigma$ for better comparability.



(a) Uniform $\mathcal{U}[0;1]$ (b) Gaussian $\mathcal{N}[0;1]$

Figure 4.8: LOF $k = 50$ scores with increasing relevant dimensions – with true outlier.

Conclusions: As seen in the first series of experiments, the curse of dimensionality does exist, and the concentration of distances happens and affects results. However, the curse is not that simple: If there is any *systematic* difference between mechanisms present in every single dimension, then the analysis task can actually become *easier* with increasing dimensionality. Whether or not the dimensionality is a problem can roughly be characterized as a "signal to noise" ratio problem: in the second series of examples, we had two mechanisms where the average difference between the two mechanisms grew faster than the in-mechanism deviations. And in fact this is not a surprising conclusion: if adding another dimension increases the separation of the two mechanisms – why should adding additional dimensions introduce problems?

This may also explain why information retrieval for textual data – even when in a vector space model – works. While the technical dimensionality are thousands of words, the majority of the dimensions will likely be 0; and except for some very frequent words (e.g. "the" and "is"; known as "stopwords") many of these dimensions will contribute signal to the analysis. This will, however, need future research.

4.3 Intrinsic Dimensionality

The term "intrinsic dimensionality" refers to the fact that the apparent data dimensionality may be misleading. This is best explained with a very simple example. Assuming we had a two-dimensional data set, where $x_1 = x_2$ for all objects. Technically, the data set is two-dimensional, but obviously it can be reduced to a single dimension without any loss of information. If we now consider distance computations on this data set, e.g. Euclidean distance, the distance will in fact also depend only on one dimension:

$$d(a,b) := \sqrt{|a_1 - b_1|^2 + |a_2 - b_2|^2} = \sqrt{2|a_1 - b_1|^2} = \sqrt{2} \cdot |a_1 - b_1| = \sqrt{2} \cdot d(a_1, b_1) \tag{4.2}$$

Redundancies do not need to be always that simple. It could be any complex relationship such as $x_3 = x_1 + x_2^2$, or there also could be noise in the relationship: $x_2 = x_1 + \mathcal{N}[0; 0.01]$. In the most extreme case, the relationship might not be the same everywhere in the data set. This makes anything but the simplest cases of intrinsic dimensionality hard to grasp analytically.

Roughly defined, the "intrinsic dimensionality" is the number of variables needed to represent the data. However, this definition is not usable in practice. Without putting additional constraints on the mapping function, any data could be considered just one dimensional, as a function of a unique object ID, or by mapping the data to its closest position on a space filling curve.[3] Furthermore, for any practical use, one will want to allow an application dependent amount of error in this representation.

How to practically measure or use the intrinsic dimensionality remains an ongoing question. There exist different definitions and measures to estimate a global or local intrinsic dimensionality, such as the expansion dimension [dCH10] and the generalized expansion dimension [HKN12]. Yet it has been noticed [HS05; dCH10; Hou+12] that many of the problems ascribed to the curse of dimensionality can be handled with appropriate approaches when the intrinsic dimensionality is low. As hinted upon by Equation 4.2, the concentration of distances is expected to happen with the intrinsic dimensionality. The Johnson-Lindenstrauss [JL84] error bounds of random projections [Ach01] can also be expected to depend on the intrinsic dimensionality instead of the technical data set dimensionality. Other problems, such as the combinatorial explosion for grid-based approaches or the bias problems of approaches testing excessive combinations of subspaces however will remain since they do not exploit correlations in the data set that may reduce the effective dimensionality.

4.4 Shared Nearest Neighbors (SNN)

The material discussed in this section is a condensed version of the following publications:

> M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 482–500. DOI: `10.1007/978-3-642-13818-8_34`
>
> T. Bernecker, M. E. Houle, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. "Quality of Similarity Rankings in Time Series". In: *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN.* 2011, pp. 422–440. DOI: `10.1007/978-3-642-22922-0_25`

An interesting alternative to traditional similarity measurement is the definition of second order distance measures based on the rankings induced by a primary similarity measure (such as an L_p-norm, cosine similarity or also a domain specific distance). The simplest and most common of these methods involves the use of *shared nearest neighbor* (SNN) information, in

[3]Assuming floating point precision, any point can actually be represented with a one-dimensional coordinate on a space filling curve – there are no irrational floating point numbers.

which the similarity value for an object pair (x, y) is a function of the number of data objects in the common intersection of fixed sized neighborhoods centered at x and y, as determined by the primary measure. The primary similarity measure can be any function that determines a ranking of the data objects relative to the query; it is not necessary for the data objects to be represented as vectors, but for example rankings obtained by a text search function can also be used as primary similarity measure.

The most basic form of shared nearest neighbor similarity measure is that of the "overlap". Given a data set S consisting of $n = |S|$ objects and $s \in \mathbb{N}^+$, let $k\mathrm{NN}_s(x) \subseteq S$ be the set of s nearest neighbors of $x \in S$ as determined using some specified primary similarity measure. The overlap between objects x and y is defined to be the intersection size

$$\mathrm{SNN}_s(x, y) = |k\mathrm{NN}_s(x) \cap k\mathrm{NN}_s(y)|. \tag{4.3}$$

Based on this definition of overlap, other shared-nearest-neighbor-based similarity measures such as the *cosine measure* have been proposed:

$$\mathrm{simcos}_s(x, y) = \frac{\mathrm{SNN}_s(x, y)}{s}, \tag{4.4}$$

so called since it is equivalent to the cosine of the angle between the zero-one set membership vectors for $k\mathrm{NN}_s(x)$ and $k\mathrm{NN}_s(y)$. This was used in [ESK03; Hou03] as a local density measure for clustering. An alternative similarity definition based on shared nearest neighbors is *set correlation*, given as:

$$\mathrm{simcorr}_s(x, y) = \frac{n}{n - s}\left(\frac{\mathrm{SNN}_s(x, y)}{s} - \frac{s}{n}\right), \tag{4.5}$$

which results when the standard Pearson correlation formula

$$r = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2)(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2)}}$$

is applied using the coordinates of the characteristic vectors of $k\mathrm{NN}_s(x)$ and $k\mathrm{NN}_s(y)$ as variable pairs. Objects of S that appear in both $k\mathrm{NN}_s(x)$ and $k\mathrm{NN}_s(y)$, or neither of $k\mathrm{NN}_s(x)$ and $k\mathrm{NN}_s(y)$, support the correlation of the two neighborhoods (and by extension the similarity of x and y); those objects that appear in one neighborhood but not the other detract from the correlation. Note that the set correlation value tends to the cosine measure when $\frac{s}{n}$ tends to zero. Set correlation was introduced in [Hou08] for the purpose of assessing the quality of cluster candidates, as well as ranking the cluster objects according to their relevance (or centrality) to the cluster.

A common variant of shared nearest neighbor similarity uses $k\mathrm{NN}$ sparsification, where two objects are required to be in each others $k\mathrm{NN}$ in order to be considered for similarity. This can be formalized as:

$$\mathrm{SNN}'_s(x,y) = \begin{cases} |k\mathrm{NN}_s(x) \cap k\mathrm{NN}_s(y)| & \text{iff } x \in k\mathrm{NN}_s(y) \wedge y \in k\mathrm{NN}_s(x) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

There are several common ways to convert a similarity measure into a dissimilarity measure. For the SNN similarity simcos (Equation 4.4) with a given number of neighbors s considered, we propose in [Hou+10] the possible distance measures

$$\begin{aligned} \mathrm{dinv}_s(x,y) &= 1 - \mathrm{simcos}_s(x,y) \\ \mathrm{dacos}_s(x,y) &= \arccos\left(\mathrm{simcos}_s(x,y)\right) \\ \mathrm{dln}_s(x,y) &= -\ln \mathrm{simcos}_s(x,y). \end{aligned}$$

For computing the nearest neighbors in high-dimensional data, SNN measures have been reported to be effective in practice, and supposedly less prone to the curse of dimensionality than conventional distance measures. SNN measures have found use in the design of merge criteria of agglomerative clustering algorithms [ESK03; GRS98; JP73], in approaches for clustering high-dimensional data sets [Hou03; Hou08], and in finding outliers in subspaces of high-dimensional data [Kri+09b]. However, in all of these studies, no systematic investigation has been made into the advantages of SNN measures over conventional distance measures for high-dimensional data.

While dinv, which we will be using throughout our experiments, is simply a linear inversion of the values, dacos penalizes slightly suboptimal similarities more strongly, whereas dln is more tolerant than dinv for a broad range of higher similarity values but approaches infinity for very low similarity values. In general, any function f that is monotonically decreasing on the interval $[0;1]$ with $f(1) = 0$ can be used to transform the SNN similarity measure into a dissimilarity measure. The functions only differ in their contrast at different ranges. All of these functions are symmetric (since sincos is symmetric) and maintain the same ranking. However, it should be noted that of the three, only dacos satisfies the triangle inequality. While most retrieval results (based simply on rankings) remain unaffected by different formulations of these secondary distances, the effects on indexing and clustering may vary from formulation to formulation. For example, the separation of clusters in terms of absolute distances depends on the concrete choice of the distance measure and on the secondary distance measure.

4.5 Empirical Observations on SNN Similarity

In this section we want to study the behavior of shared nearest neighbor similarity, to understand the benefits of using it, in particular with respect to high-dimensional data.

This section is an excerpt (focused on the aspects relevant for outlier detection) of the study published as:

M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 482–500. DOI: 10.1007/978-3-642-13818-8_34

4.5.1 Data Sets

To study the effects of the curse of dimensionality, we require a series of data sets that scale in dimensionality without introducing bias. After controlling for dimensionality, each of the sets in the series must be constructed such that they share common characteristics to the greatest degree possible. This is difficult to achieve with real world data, since the different attributes often vary in their scales and expressivity. If generating low-dimensional examples from a high-dimensional data set, it is not always clear how to select the projective dimensions fairly. In addition, well-defined ground truth sets necessary for assessing the expressiveness of query results are typically unavailable for large real data sets. The use of synthetic data allows us to study individual effects separately, while real data sets usually prevent the isolation of different influences. For these reasons we construct several series of artificial data sets using pseudo-random generators with largely fixed parameters, avoiding those parameter choices leading to data sets with groupings that are either too difficult or too easy to discriminate. Unless stated otherwise, the synthetic data sets were constructed with the following characteristics: $n = 10,000$ points grouped into $c = 100$ clusters in up to $D_{\max} = 640$ dimensions. Cluster sizes are randomized with a mean of $\frac{n}{c} = 100$ and standard deviation $\frac{n}{10 \cdot c} = 10$, with the size of the last generated cluster adjusted so that the total number of points is n. If generating data sets for a series, those sets with dimensionality $d < D_{\max}$ were generated so that their attributes coincided with the first d attributes of all other data sets having dimensionality $\geq d$.

For each object, attribute values were generated depending on whether the attribute is to be considered 'relevant' or 'irrelevant' for the formation of the cluster to which the object belongs. If the i-th attribute is deemed relevant to the j-th cluster, the value of this attribute for all members of c are normally distributed with a standard deviation in the range $\sigma_{j,i} \in [0.05; 0.8]$, and a mean in the range $\mu_{j,i} \in [\frac{\sigma_{j,i}}{2}; 1 - \frac{\sigma_{j,i}}{2}]$. These ranges were chosen to avoid overly compact or overly wide distributions, as well as boundary effects, while still providing a wide variety of distributions and overlaps. No additional clipping or normalization was applied. Any attributes irrelevant to the cluster were assigned noise values uniformly distributed in the interval $[0; 1]$.

For the experimentation, 6 synthetic data series were created, each consisting of 7 sets of differing dimensionality $d = 10, 20, 40, 80, 160, 320, 640$:

- *All-Relevant*: in this series, all attributes were generated so as to be relevant for all clusters.
- *10-Relevant*: in this series, the first 10 attributes are relevant for all clusters, the remaining attributes are irrelevant.

- *Cyc-Relevant*: in this series, the i-th attribute is relevant for the j-th cluster when $i \bmod c = j$; otherwise, the attribute is irrelevant. This series has $n = 1,000$ and $c = 10$.
- *Half-Relevant*: in this series, for each cluster, an attribute was chosen to be relevant with probability $\frac{1}{2}$, and irrelevant otherwise. The selection of attributes was consistent within a cluster, and performed independently of the selection for other clusters.
- *All-Dependent*: this series is derived from *All-Relevant* introducing correlations among attributes.
- *10-Dependent*: this series is derived from *10-Relevant* introducing correlations among attributes.

For the correlated data sets *All-Dependent* and *10-Dependent*, the i-th attribute value X_i was generated by computing $X_i = Y_i$ for $1 \leq i \leq 10$, and $X_i = \frac{1}{2}(X_{i-10} + Y_i)$ for $i > 10$, where Y_i is the attribute of the corresponding uncorrelated data set *All-Relevant* or *10-Relevant*. This way of introducing correlations is inspired by Example 3 in [Bey+99].

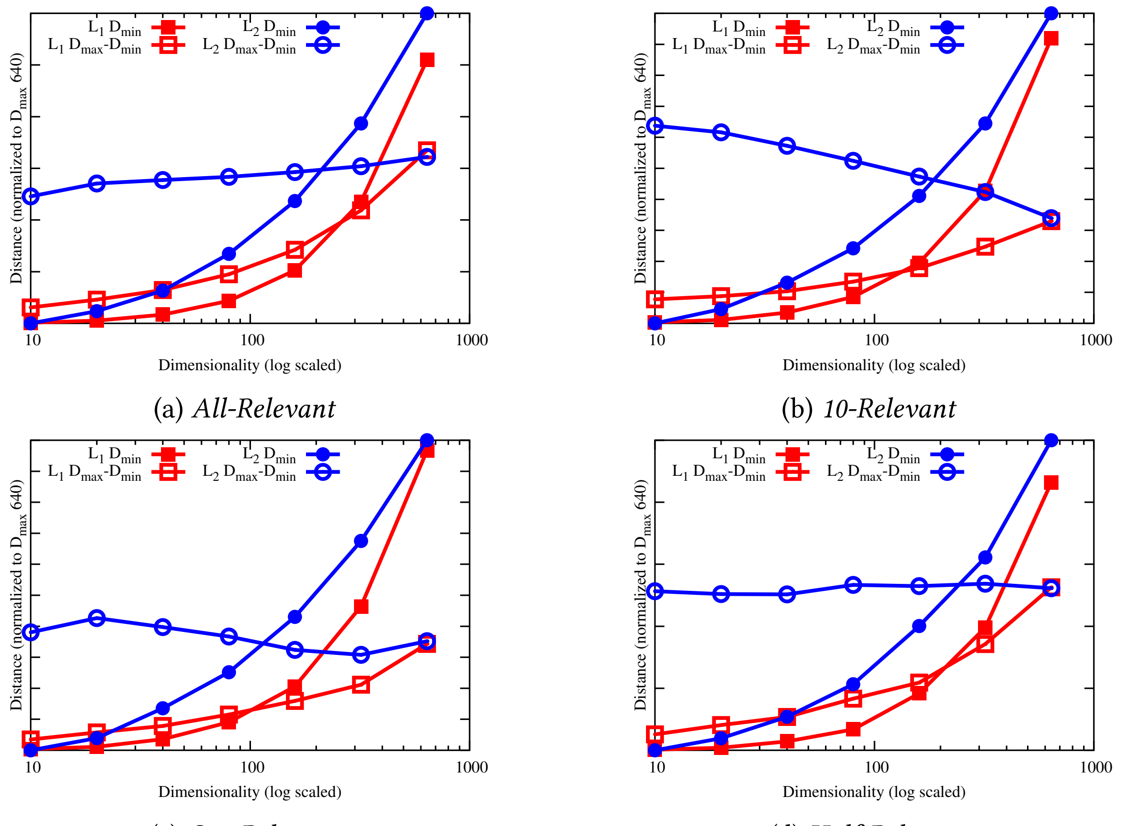
These 6 series provide us with the means to study different aspects of the curse of dimensionality. Data series *All-Relevant* is the basic setting referred to as the "concentration of distances". However, the sets differ from those considered in many earlier studies [Bey+99; HAK00; AHK01], and conforms with Bennet et al. [BFG99] in that the data objects are partitioned into clusters (as are all our data sets). Data sets *10-Relevant* and *Cyc-Relevant* relate exclusively to the "signal to noise" ratio in different settings. The clusters are further distinguished in the data set *Cyc-Relevant*, where every attribute is relevant for exactly one cluster. In the series *Half-Relevant*, we give up control of the number and choice of relevant attributes. Half the attributes are expected to be relevant to a given cluster, but the selection of relevant attributes varies (independently) from cluster to cluster.

Our synthetic data sets do not satisfy the i.i.d. (independent and identically distributed) assumptions used in the proofs of Beyer et al. [Bey+99], since the sets are composed of multiple clusters that overlap in some dimensions and are well-distinguished in others. However, the analysis of François et al. [FWV07] applies when dimensional values are comparable in their extent and exhibit the same properties as for normalized data.

As intended, our synthetic data sets show the typical behavior ascribed to the curse of dimensionality, more precisely they exhibit the concentration of distances (see Equation 4.1). Figure 4.9 plots the numerator and denominator of the contrast formula $\frac{D_{\max}-D_{\min}}{D_{\min}}$ for selected data sets, to demonstrate that $D_{\min}$ (the solid symbols) indeed grows much faster than the difference $D_{\max} - D_{\min}$ (the hollow symbols). The plots indicate that $D_{\min}$ grows exponentially faster than $D_{\max} - D_{\min}$.

For the experiments with time series data, we chose a number of classic benchmark examples such as *Cylinder Bell Funnel (CBF)*, where a low magnitude noise is overlaid with a high magnitude signal that either is constant (cylinder), increasing to a cut off (bell) or decreasing after an on-set (funnel). We will, however, only show summarized results in this section.

In addition to the synthetic data sets described above, we also considered real-world data sets for our study. Real-world data sets suitable for this study are difficult to obtain, since they



(a) *All-Relevant* (b) *10-Relevant*

(c) *Cyc-Relevant* (d) *Half-Relevant*

Figure 4.9: Curse of dimensionality: $D_{\max} - D_{\min}$ compared to $D_{\min}$

should have a reasonable size, number of classes, dimensionality, comparable dimensions and of course a solid ground truth. Results for real data can be difficult to interpret due to a lack of knowledge of the underlying data distributions, even when ground-truth class knowledge is available. Nevertheless, we report experimental results for 3 real data sets. The first real data set we used is the *Multiple Features* data set [AN07]. It consists of 2000 instances from 10 classes (corresponding to the digits 0 to 9). There are two variants, one with 649 dimensions (coming from multiple feature extraction algorithms and giving the data set its name), and another with 240 dimensions (the pixel averages features, which is the largest subset of directly comparable features). The second set considered is the *Optical Recognition of Handwritten Digits* data set [AN07]. It consists of 5620 instances from 10 classes (also corresponding to the digits 0 to 9) in 64 dimensions, in the form of an 8×8 grid of integer values in the range of 0 to 16 obtained by downsampling from a larger 32×32 grid. The third real data set comes from the Amsterdam Library of Object Images (ALOI) image database [GBS05], each image being described by 641 dense features based on color and texture histograms (for a detailed description of how the vectors were produced, see [Bou+01]). The full *ALOI-IKONA* data set consists of $110,250$ images of $1,000$ objects taken from different orientations and in different lighting conditions, each object being treated as a class. We used only the first $22,050$ instances of the data set, covering the first 205 objects, with an average class size of approximately 107 objects.

4.5.2 Distance Measures

As primary distance measures we considered for our experimental evaluation a range of different L_p-distances, in particular the Manhattan (L_1) and Euclidean (L_2) distances, and the $p = 0.6$ and $p = 0.8$ fractional L_p-distances. In addition, we used also the cosine distance, here referred to as invcos, as it is computed as the arc of the cosine similarity. All these distance measures can be used as the primary distance for the computation of a secondary similarity simcos_s, as defined in Equation 4.4. For our experiments, we use the distance measure $1 - \mathrm{simcos}_s$, and compare the performance of this secondary distance measure with the corresponding primary distance measure to assess whether the accuracy is improved. There are other possibilities for constructing distance measures from similarity measures. The particular choice of method, however, does not affect the ranking of query results, although it may influence the contrast.

For the experiments involving time-series data, the traditional simple distance functions (such as cosine similarity and L_p-norms) did not perform very well. However, to cope with spatio-temporal data, several specialized distance measures have been developed that attempt to determine the best matching of events along the time axis. Since SNN puts next to no requirements on the primary distance function, it can also be used with these specialized distance functions. One of the most prominent of these is the *Dynamic Time Warping (DTW)* distance [BC94], used extensively in speech recognition. *DTW* supports asynchronous matching — matches with shifts along the time dimension — by extending each sequence with repeated elements, and applying L_p-norms to the extended time series. The advantages of *DTW* are invariance to (local) phase-delays, the acceleration or deceleration of signals along the time dimension, and the ability to support matches between series of differing lengths. Like the L_p-norms, *DTW* requires a complete matching of both time series, in that each value from one time series must be matched with at least one value from of the other time series. For this reason, *DTW* is sensitive to noise and outliers within the time series.[4]

In general, distance measures that are robust to extremely noisy data typically violate the triangle inequality [VKG02], and thus are inapplicable for most indexing methods. Well-known distance measures for sequence data that fall into this category are the *Longest Common Subsequence (LCSS)* distance [VKG02], the *Edit Distance on Real sequence (EDR)* [CÖO05] and the *Edit distance with Real Penalty (ERP)* [CN04b]. In contrast with *DTW*, *LCSS* and *EDR*, the measure *ERP* has the advantage of satisfying the triangle inequality and is therefore a distance metric. The aforementioned measures are adaptations of the edit distance, a commonly-used distance

[4]Note that we are not looking for these in-series outliers, but we are interested in complete series that are as a whole anomalous with respect to the full data set. The in-series outliers are a different kind of outlier, that can better be detected with traditional one-dimensional statistical methods.

measure for matching strings that can accommodate gaps in the matching. Unlike the L_p-norms and *DTW*, these measures are able to ignore noise and outliers. As such, edit distance variants are better at coping with different sampling rates, different time rates, and different series lengths; they can also be computed more efficiently.

The high computational cost associated with distance measures for time-series data has led to the development of many methods for dimensionality reduction, in which distance measures are applied to subsets of features extracted from objects in the series [AFS93; YJF00; ANR74; CF99; CN04a]. Distance measures based on dimensionality reduction can be regarded as specialized similarity measures, in that they process the full set of spatio-temporal features to ultimately produce similarity values for the original objects.

4.5.3 Evaluation Criteria

The purpose of a distance function is to facilitate the separation of data objects similar to the query from those objects which are not similar. In our context of different generating mechanisms, similar objects correspond to points that have been generated according to the same mechanism as the query point, whereas irrelevant points have been generated by different mechanisms. The discriminative ability of a given distance function can best be evaluated by computing a nearest neighbor ranking of all data points with respect to a given query point. Ideally, at the top positions of the ranking, we would find all objects drawn from the same natural cluster as the query object, followed by the objects from outside the cluster. Different dissimilarity functions have different ranges of values. For example, the maximum L_p-distance in the unit cube is $\sqrt[p]{d}$; dacos_s has a value range of $[0; \frac{\pi}{2}]$, and the value of $1 - \mathrm{simcos}_s$ lies conveniently in the range $[0; 1]$. To evaluate the discriminative ability of dissimilarity functions without referring to the actual values, we compute Receiver Operating Characteristic (ROC) curves that compare the true positive rate with the false positive rate (see Section 5.4 for a detailed discussion of ROC curves). For each query, the objects are ranked according to their similarity to the query point. We can compute the matching ROC curve and the corresponding area under the curve (AUC) for each ranking result. An AUC of 1.0 indicates perfect discrimination – all relevant objects are ranked ahead of all other objects. An AUC of 0.5 indicates a total lack of discriminative ability, since this value is what would be expected with a uniform random permutation of the query result set. An AUC significantly less than 0.5 indicates a reversed ordering. The ROC curve and its AUC value provide a summary for a single ordering of points – that is, for a single query object. By generating a ROC curve and AUC value for each data object, the mean AUC value and standard deviation could then be used to rate the quality for a particular distance function. However, we expect points near the center of a cluster (the mean of the generating distribution) to discriminate well for many distance functions. On the other hand, for points near the border of a cluster or in the overlap of clusters, values of the dissimilarity measure will most likely perform less well. Therefore, at data generation time, we assign to each point a centrality rank, based on both its deviation from the mean and the size of the cluster, so as to normalize across clusters of differing sizes. The point generated for cluster M that is closest to the mean of M is assigned a centrality of 1, and the point of M that is farthest from the mean of M is assigned a centrality of 0. To obtain readable graphs, the ROC AUC values then are aggregated into bins based on their centrality values. This allows us to plot the degradation of the distance function with respect to the centrality of a point within its distribution. For the plots shown in this Chapter, we will be using three bins for the central 20%, middle 60% and outer 20%.

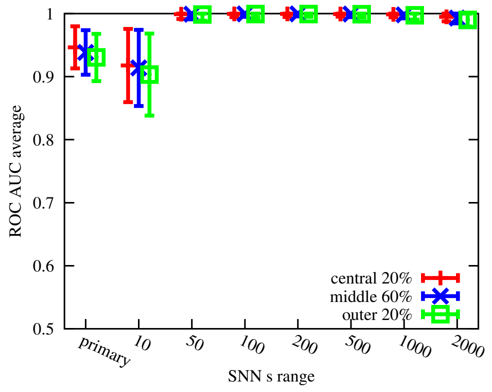
4.5.4 Experimental Results

Experimental Results on Synthetic Data: At first glance, the fact that our synthetic data sets exhibit the typical symptoms of the curse of dimensionality would seem to indicate that these data series are not amenable to indexing or mining. However, such a conclusion would be unnecessarily pessimistic. Especially for data sets with many relevant attributes (such as the *All-Relevant* series), any given number of clusters should become distinguishable when the number of relevant attributes becomes sufficiently large. This intuition is justified by examples such as the combination of kernels and support vector machines (SVM): the number of dimensions is increased in order to be able to separate classes linearly by hyperplanes [BGV92]. In fact, what is stated as a condition for the pairwise stability of clusters in [BFG99], we would expect to hold for any two clusters where the number of discriminative attributes dominates. This is not an essentially original contribution of our study but confirms prior results.

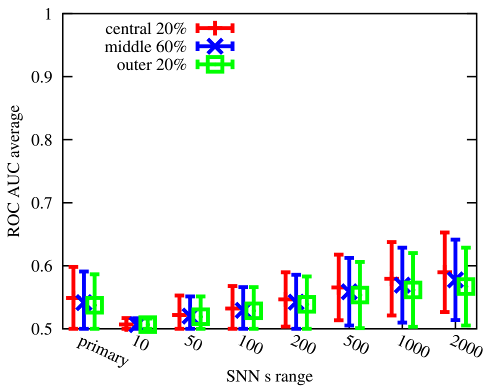
One point that must be stressed is that while the curse of dimensionality tells us not to rely on the absolute values of distances, it is still viable to use distance values to derive a ranking of data objects. An ε-range query is dependent upon the choice of an appropriate value of ε, and thus suffers from the lack of contrast, whereas a k-nearest neighbor query will retrieve the top k neighbors independently of their absolute distance values. Hence, the computation of k-nearest neighbor queries and rankings has the potential to be viable in higher dimensions, whereas that of ε-range queries likely does not. Furthermore, although the curse of dimensionality contrast formula holds for all our data sets, the ranking results are not tied solely to the data dimensionality, and can in certain situations improve significantly with increasing dimensionality, as reported in [Bey+99]. The conclusion we draw is supported by the research literature as well as by our experiments on our synthetic data sets:

Conclusion 4.5.1 (Relevant vs. Irrelevant Attributes)**:** The quality of the ranking – and thus the separability of the different generating mechanisms – may not necessarily depend on the data dimensionality, but instead on the number of relevant attributes in the data set.

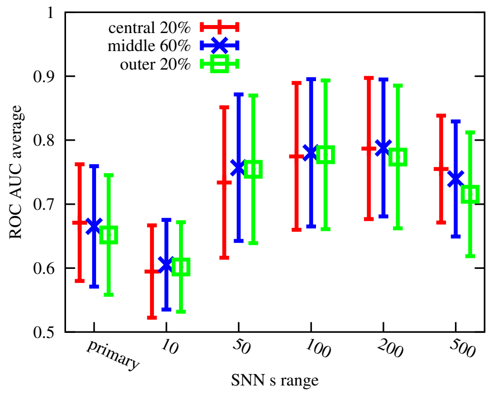
More specifically, there are two contrary effects of an increase in dimensionality when the number of relevant attributes is high: the relative contrast between points tends to decrease, but the separation among different generating mechanisms can increase. On the other hand, if the data dimensionality is high and the number of relevant dimensions is rather low, the curse of dimensionality fully applies, and hampers any analysis task. In retrospect, this is an important



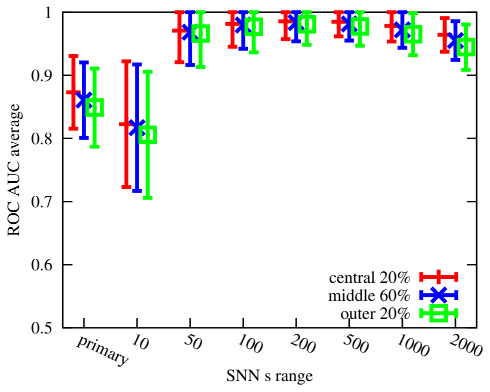
(a) *All-Relevant*



(b) *10-Relevant*



(c) *Cyc-Relevant*

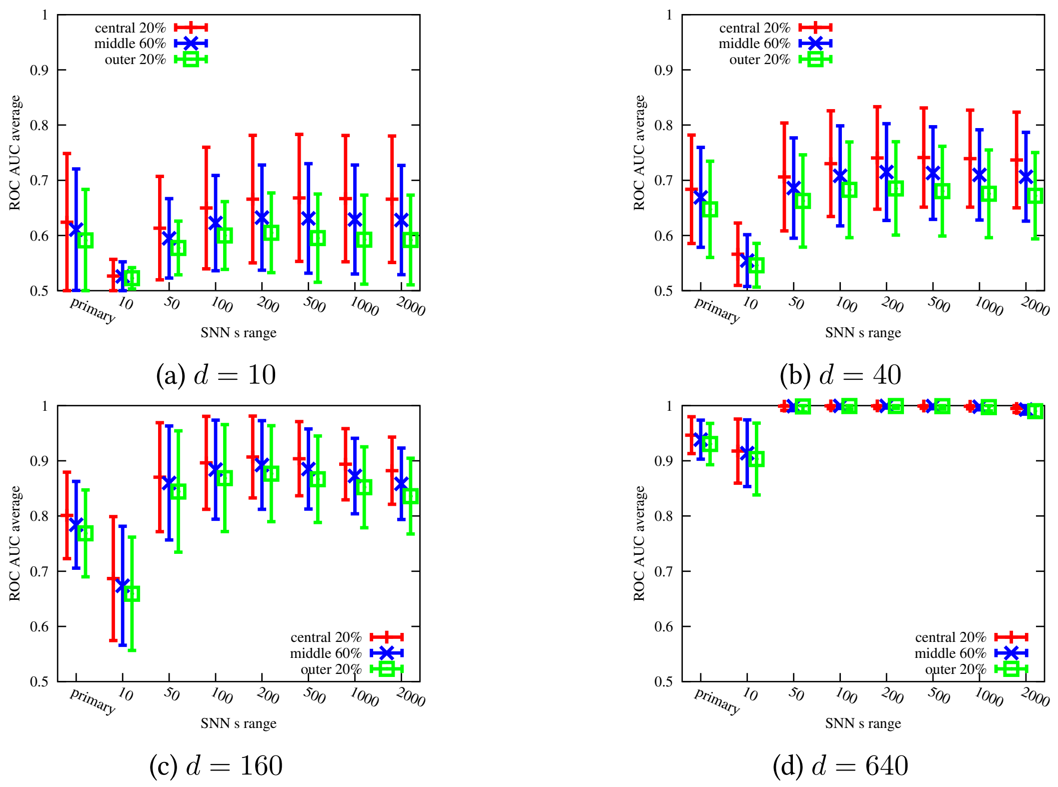


(d) *Half-Relevant*

Figure 4.10: Ranking quality with different SNN distances based on L_2 at 640 dimensions.

yet unsurprising conclusion to draw. Nevertheless, it has not gained much recognition in the research literature to date.

As a further original contribution of this study, we evaluate the behavior of SNN as a secondary similarity measure. Motivated by the findings sketched above, an improved performance can be expected for a rank-based similarity measure such as SNN, whenever the ranking provided by the primary similarity measure is meaningful. Figure 4.10 compares results for the secondary distance measure with different SNN reference sizes s, based on Euclidean distance as the primary distance measure, for dimension $d = 640$. The performance of the corresponding primary distance is given on the left side of each diagram as a reference. Results for lower dimensionalities are comparable, and are shown in the following figures. For easily separable data sets such as *All-Relevant*, most choices of s yield excellent results. On *Half-Relevant* and *Cyc-Relevant*, the best results are achieved for choices of s of the same order as the cluster size (100). This can also be seen for *All-Relevant* on lower dimensionality, where the contrast between the results is better. On the barely separable *10-Relevant* data set, even larger values of s seem to be needed, although the average ROC AUC score is not significant, being below 0.6. Figure 4.11 shows the same plots for different dimensionalities of the *All-Relevant* data set. It can be seen that



(a) $d = 10$ (b) $d = 40$

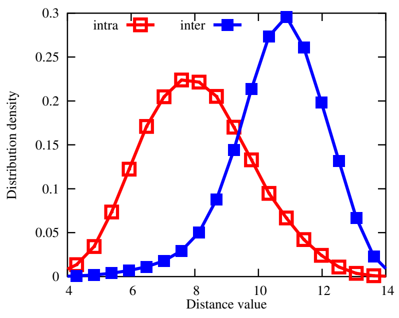
(c) $d = 160$ (d) $d = 640$

Figure 4.11: Ranking quality for the *All-Relevant* set with different SNN distances based on L_2.

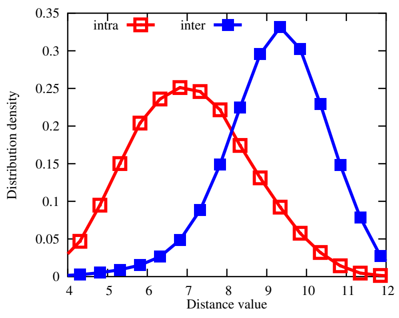
by using an SNN distance, a considerable improvement can be achieved given that the data set is sufficiently separable, and that the parameter s is chosen roughly in the range of the cluster size. In particular, the secondary distance performs very well at high dimensionalities, and is reasonably robust with respect to the choice of s. The observations on the correlated data sets (given in the supplementary material) are quite similar. To summarize, we can draw the following conclusion from our experiments:

Conclusion 4.5.2 (Ranking Quality Improvement)**:** Our experiments suggest that the use of an SNN similarity measure can significantly boost the quality of a ranking compared to the use of the primary distance measure alone, provided that the primary distance already provides some degree of distinguishability of clusters.

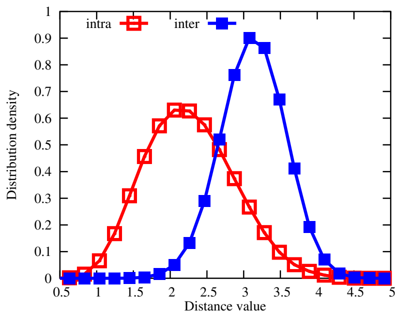
The experimental results confirm that although the discrimination of primary distances worsens with increasing data dimensionality, the natural data groupings may still be separable and, if so, the neighborhoods of query points would contain many points from the same grouping. Clearly, for two points from a common data grouping, when increasing the value of s, the probability that their neighborhoods have significant overlap increases as well. On the other hand, if s is substantially larger than the size of the grouping, many objects from different groups are contained in the neighborhoods of the two points, and the performance of secondary distance measures become less predictable.



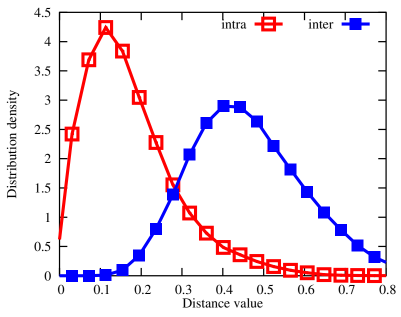
(a) *MultipleFeatures*



(b) *MultiFeat pixel*

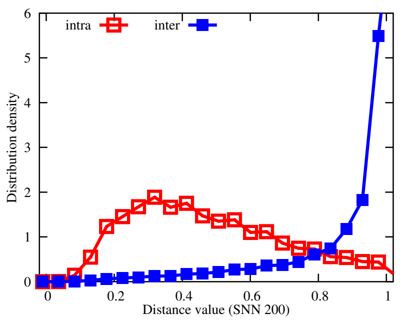


(c) *Optdigits*

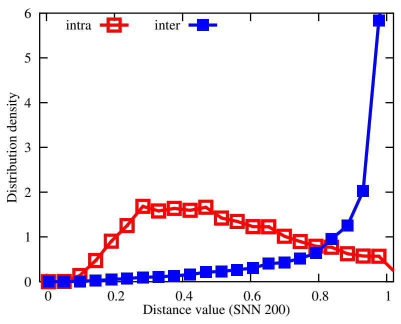


(d) *ALOI-IKONA*

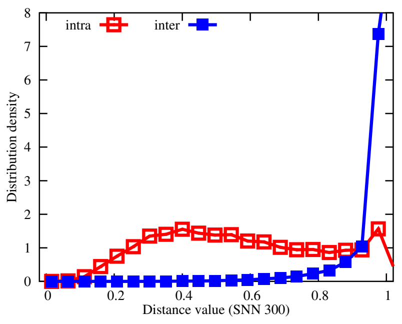
Figure 4.12: Distributions of intra-class and inter-class distances (Euclidean distance).



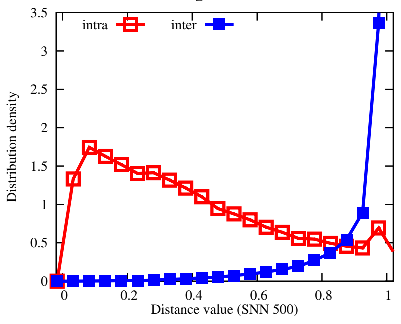
(a) *MultiFeat* $s = 200$



(b) *MultiFeat pixel* $s = 200$



(c) *Optdigits* $s = 300$



(d) *ALOI-IKONA* $s = 500$

Figure 4.13: Distributions of intra- and inter-class (SNN-Euclidean distance).

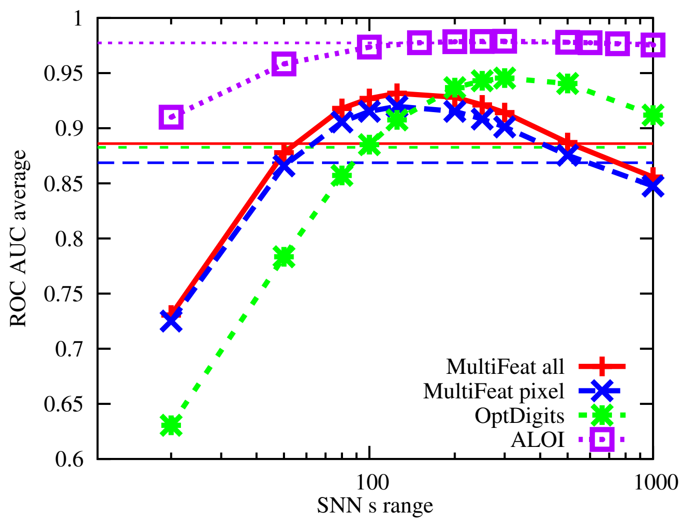


Figure 4.14: Ranking quality with different SNN distances based on Euclidean distance (straight lines: ranking quality with primary distance).

Experimental Results on Real Data: Experiments on artificial data allow more control over parameters such as the data dimension, and are more amenable to studying effects on the performance of distance measures in isolation. Real-world data, on the other hand, is considerably more difficult to control in this way. Nevertheless, in this section we offer experimental results for real-world data sets in order to validate and confirm some of the effects observed for artificial data. As seen in Figure 4.12, on all the real data sets considered, the distance distributions are approximately Gaussian (which is to be expected in high dimensionalities for the L_p-norms, due to the central limit theorem). It is also apparent that these data sets will be reasonably separable, since the overlap of the distance distributions is not very large. The results for other primary distances are comparable. Figure 4.13 shows the histogram results when using an SNN-based distance. The data set groupings have become very well separable. The effects of s on the results for real-data are as one would expect from the experiments on artificial data: Figure 4.14 displays the results for various sizes of s. Choosing s to match the class size gives reasonable results; however, the best performances are achieved with even larger values of s. Only when s approaches the full data set size does performance drop. The benefits of using SNN on the *ALOI-IKONA* data set are minimal, as the groupings of that set are already very well separable for primary distances.

Experimental Results on Time Series Data: We performed similar experiments on time series data, as a prototype of high-dimensional data. Note that many time series usually have a much lower intrinsic dimensionality, since the raw data dimensionality is determined by the measurement process.

The results are very similar to what we have been seeing on the previous data sets: given a primary distance that offers a reasonably good ranking, the shared nearest neighbor distances

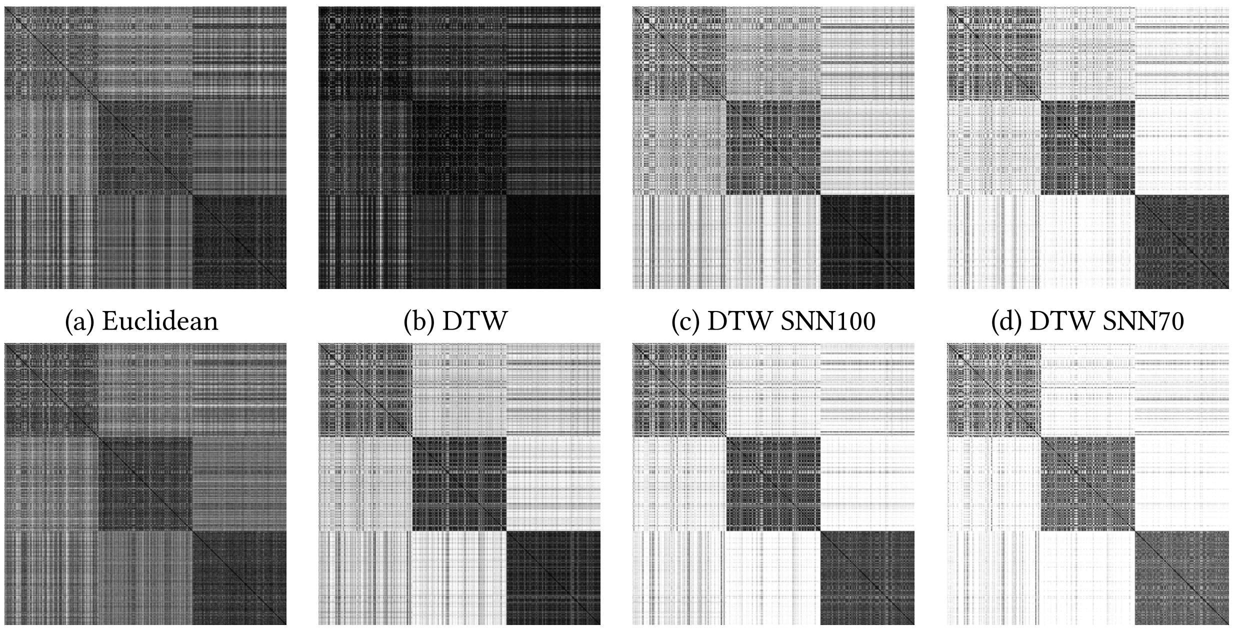


Figure 4.15: Similarity matrices for *CBF*: All items are sorted according to their class label and pairwise distances are plotted. Similarity values are shown in grey scale, where black indicates high similarity and white indicates low similarity.

further improve the ranking, while offering a similarity function constrained to the easier to use value range of $[0; 1]$ with improved contrast. Figure 4.15 is an exemplary results from this experiment, visualizing the improved contrast on the artificial *Cylinder-Bell-Funnel (CBF)* data set, consisting of time series with low magnitude noise and three classes of high magnitude signals. The primary distance functions in Figures 4.15a, 4.15b and 4.15e have much worse visual contrast than the SNN-based versions. Similar to the results before, setting the neighborhood size to approximately the class size offered best results. In this plot, each class consisted of 100 samples. When decreasing the neighborhood size to 50, the result has become clearly worse in contrast, and the retrieval performance degraded similarly. Also, the signal-to-noise ratio was found to impact overall performance as expected from the curse of dimensionality.

For the full results of the study, see the original publications [Hou+10; Ber+11] and the additional plots published on the web page `http://www.dbs.ifi.lmu.de/cms/Research/SNN`.

4.6 Visualization of High-Dimensional Data

The visualization method introduced in this section was published as:

> E. Achtert, H.-P. Kriegel, E. Schubert, and A. Zimek. "Interactive Data Mining with 3D-Parallel-Coordinate-Trees". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), New York City, NY.* 2013, pp. 1009–1012. DOI: `10.1145/2463676.2463696`

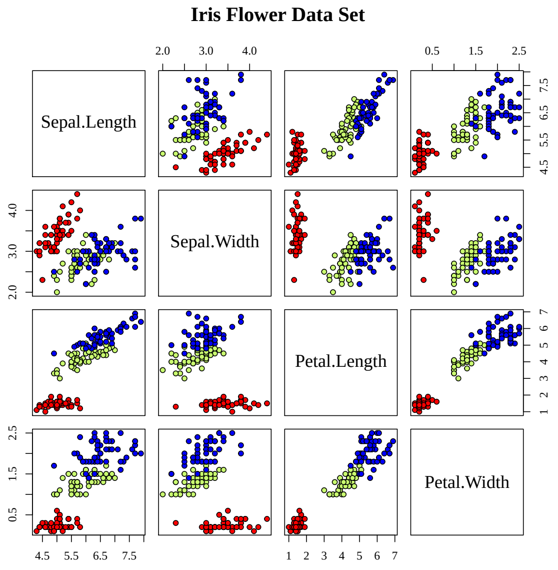
Automated data mining methods for high-dimensional data, such as subspace and projected clustering [Agg+99; AY00; Böh+04b] or outlier detection [RRS00; AY01; Kri+09b], have found much attention in database research. Yet all methods in these fields are still immature and all have deficiencies and shortcomings (see the discussion in surveys on subspace clustering [KKZ09; Sim+13; KKZ12] or outlier detection [ZSK12a]). Visual, interactive analysis and supporting tools for the human eye are therefore an interesting alternative but are susceptible to the "curse of dimensionality" themselves.

Even without considering interactive features, visualizing high-dimensional data is a non-trivial challenge. Traditional scatter plots work fine for 2D and 3D projections, but for high-dimensional data, one has to resort to selecting a subset of features. Technically, a 3D scatter plot also is a 2D visualization. In order to get a proper 3D impression, animation or stereo imaging is needed. In Figure 4.16a, each pair of dimensions is visualized with a scatter plot. Figure 4.16b visualizes 3 dimensions using a scatter plot.

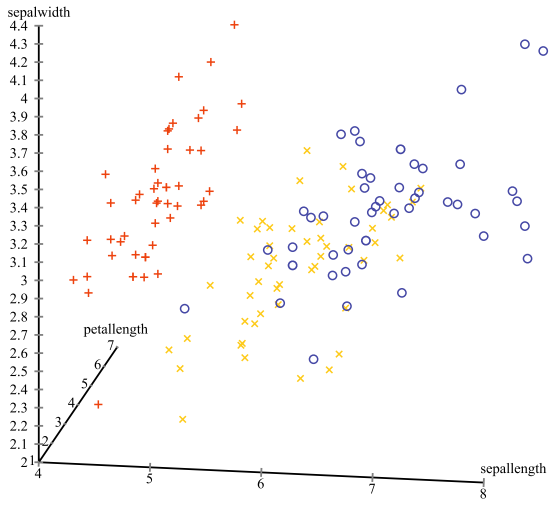
Parallel coordinates were popularized for data mining by Alfred Inselberg [ID90; Ins09]. By representing each instance as a line path, we can actually visualize more than 2 dimensions on a 2-dimensional plane. For this, axes are placed in parallel (or alternatively, in a star pattern as in [FCI05]), and each object is represented by a line connecting the coordinates on each axis. Figure 4.16c is the same data set as above, with the four dimensions parallel to each other. Each colored line is one observation of the data set. Some patterns become very well visible in this projection. For example one of the classes is clearly separable in attributes 3 and 4, and there seems to be an inverse relationship between axes 1-2 as well as 2-3: one of the three Iris species has shorter, but at the same time wider sepal leaves. Of course in this particular, low-dimensional data set, these observation can also be made on the 2D scatter plots in Figure 4.16a.

4.6.1 Related Work

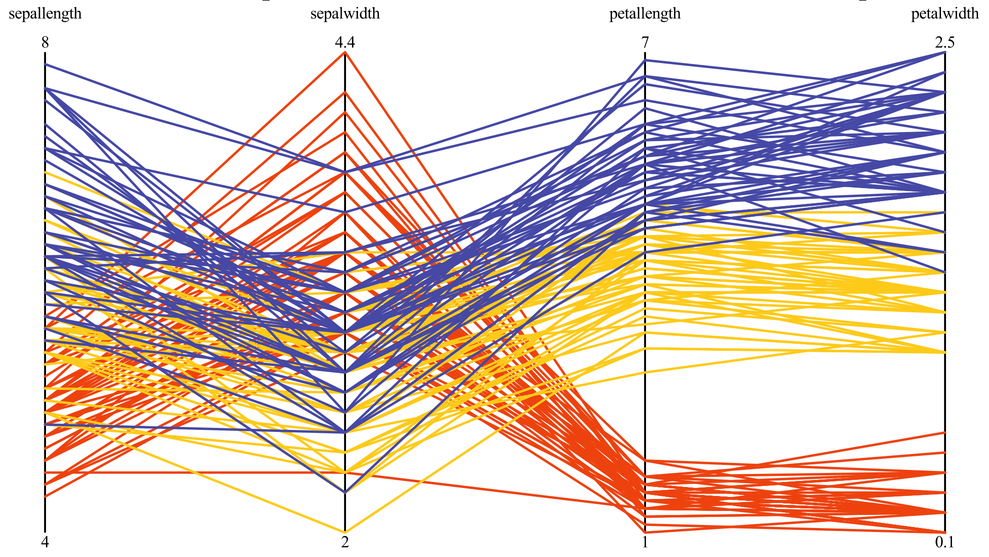
The use of parallel coordinates for visualization has been extensively studied [ID90; Ins09]. The challenging question here is how to arrange the coordinates, since patterns are visible only between direct neighbors. Inselberg [Ins09] discusses that $\mathcal{O}(\frac{N}{2})$ permutations suffice to visualize all pairwise relationships, but does not discuss approaches to choose good permutations automatically. The complexity of the arrangement problem has been studied by Ankerst



(a) Pairwise scatterplots



(b) 3D scatterplot



(c) Parallel coordinates plot for Iris data set

Figure 4.16: Visualization examples for Iris data set.

et al. [ABK98]. They discuss linear arrangements and matrix arrangements, but not tree-based layouts. While they show that the linear arrangement problem is NP-hard – the traveling salesman problem – this does not hold for hierarchical layouts. Guo [Guo03] introduces a minimum-spanning-tree-based heuristic, that actually is more closely related to single-linkage-clustering, to find a linear arrangement. Yang et al. [Yan+03] discuss integrated dimension reduction for parallel coordinates, which builds a bottom-up hierarchical clustering of dimensions, using a simple counting and threshold-based similarity measure. The main focus is on the interactions of hiding and expanding dimensions. Wegenkittl et al. [WLG97] discuss parallel coordinates in 3D; however their use case is time series data and trajectories, where the axes have a natural order or even a known spatial position. As such, their parallel coordinates remain linear ordered. A 3D visualization based on parallel coordinates [FCI05] uses the third dimension for separating the lines by revolution around the x axis to obtain so called star glyphs. A true 3D version of parallel coordinates [Joh+06] does not solve or even discuss the issue of how to obtain a good layout: one axis is placed in the center, the other axes are arranged in a circle around it and connected to the center. Tatu et al. [Tat+12] discuss measures of interestingness to support visual exploration of large sets of subspaces.

4.6.2 Arranging Dimensions

4.6.2.1 Similarity and Order of Axes

An important ingredient for a meaningful and intuitive arrangement of data axes is to learn about their relationship, similarity, and correlation. In the 3DPC extension to ELKI [Ach+13], we provide different measures and building blocks to derive a meaningful order of the axes. A straightforward basic approach is to compute the covariance between axes and to derive the correlation coefficient. Since strong positive correlation and strong negative correlation are equally important and interesting for the visualization (and any data analysis on top of that), only the absolute value of the correlation coefficient is used to rank axis pairs. A second approach considers the amount of data objects that share a common slope between two axes. This is another way of assessing a positive correlation between the two axes but for a subset of points. The larger this subset is, the higher is the pair of axes ranked. Additionally to these two baseline approaches, we adapted measures from the literature: As an entropy-based approach, we employ MCE [Guo03]. It uses a nested-means discretisation in each dimension, then evaluates the mutual information of the two dimensions based on this grid. As fourth alternative, we use SURFING [Bau+04], an approach for selecting subspaces for clustering based on the distribution of k nearest neighbor distances in the subspace. In subspaces with a very uniform distribution of the kNN distances, the points themselves are expected to be uniformly distributed. Subspaces in which the kNN distances differ strongly from the mean are expected to be more useful and informative. HiCS [KMB12] is a Monte Carlo approach that samples a slice of the data set in one dimension, and compares the distribution of this slice to the distribution of the full data set. This method was actually proposed for subspace outlier detection, but we found it valuable for arranging subspaces, too. Finally, a recent approach specifically designed to support visual exploration of high-dimensional data [Tat+11] is ordering dimensions according to their concentration after performing the Hough transformation [Hou62] on the 2D parallel coordinates plot.

4.6.2.2 Tree Visualization

Based on these approaches for assessing the similarity of axes, we compute a pairwise similarity matrix of all dimensions. Then Prim’s algorithm is used to compute a minimum spanning tree for this graph, and one of the most central nodes is chosen as root of the visualization tree. This is a new visualization concept which we call 3D-parallel-coordinate-tree (3DPC-tree). Note that both building the distance matrix and Prim’s algorithm run in $\mathcal{O}(n^2)$ complexity, and yet the ordering can be considered optimal. So in contrast to the 2D arrangement, which by Ankerst et al. [ABK98] was shown to be NP-hard, this problem actually is easier in 3 dimensions due to the extra degree of freedom. This approach is inspired by Guo [Guo03], except that we directly use the minimum spanning tree, instead of extracting a linear arrangement from it. For the layout of the axis positions, the root of the 3DPC-tree is placed in the center, then the subtrees are layouted recursively, where each subtree gets an angular share relative to their count of leaf nodes, and a distance relative to their depth. The count of leaf nodes is more relevant than the total number of nodes: a chain of one node at each level obviously only needs a width of 1.

Figure 4.17 visualizes the layout result on the 2D base plane for an example data set containing various car properties such as torque, chassis size, and engine properties. Some interesting relationships can already be derived from this plot alone, that the fuel capacity of a car is primarily connected to the length of the car (longer cars in particular do have more space for a tank), or the number of doors being related to the height of the car (sports cars tend to have fewer doors and are shallow, while when you fit more people in a car, they need to sit more upright).

An alternate layout can be obtained by using metric learning methods on the similarity matrix, such as multidimensional scaling. MDS computes an optimal 2-dimensional projection that yields approximately the same distances as the input similarites. This however needs future work, as it may place highly similar axes too close to each other.

4.6.2.3 Outlier- or Cluster-based Color Coding

An optional additional function for the visualization is to use color coding of the objects according to a clustering or outlier detection result. Since our 3DPC-tree interactive visualization is implemented using the ELKI framework [Ach+11; Ach+12], a wide variety of such algorithms comes with it, such as specialized algorithms for high-dimensional data (e.g., SOD [Kri+09b], COP [Kri+12], or subspace clustering algorithms [Agg+99; AY00; Böh+04b; Böh+04a; Ach+06a; Ach+08]) but also many standard, not specialized, algorithms.

Using color-codes of some algorithm result in the visualization is useful for example to facilitate a convenient analysis of the behavior of the algorithm.

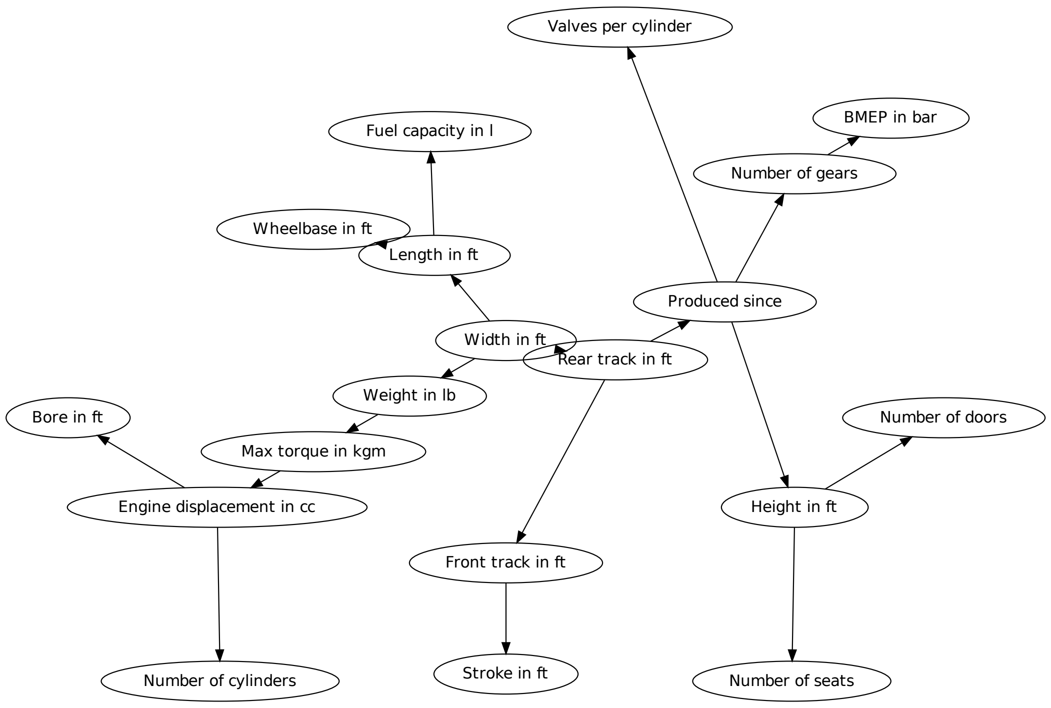


Figure 4.17: Axis layout for cars data set.

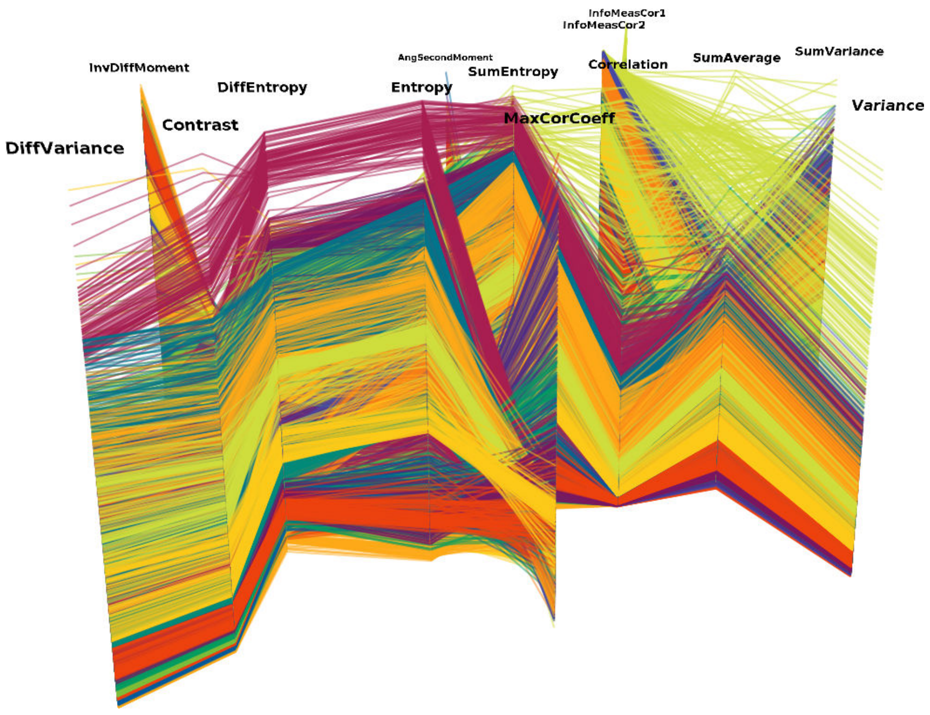


Figure 4.18: 3DPC-tree plot of Haralick features for 10692 images from ALOI, ordered by the HiCS measure.

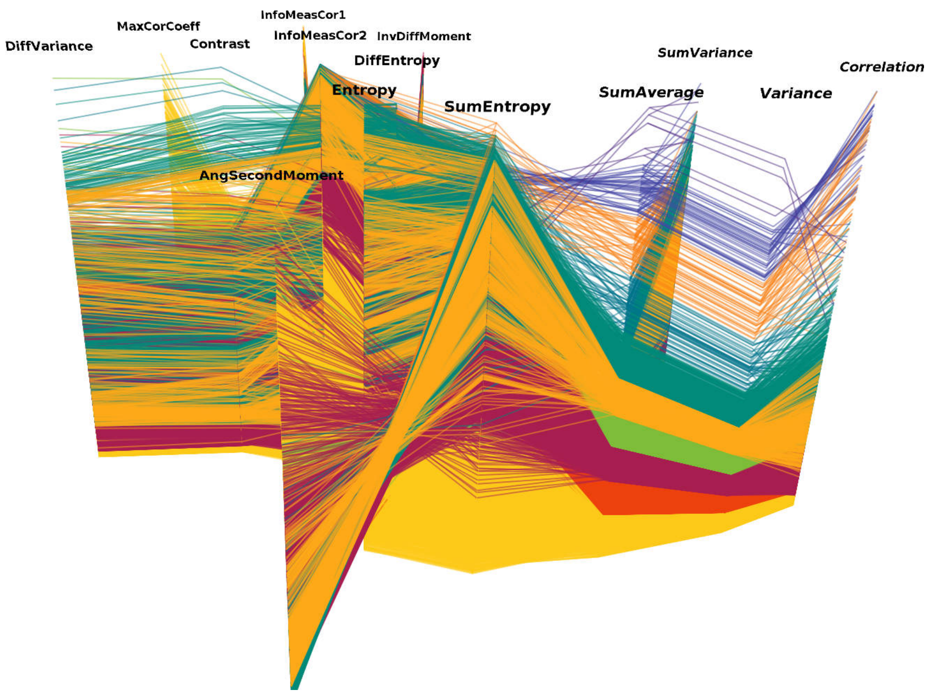


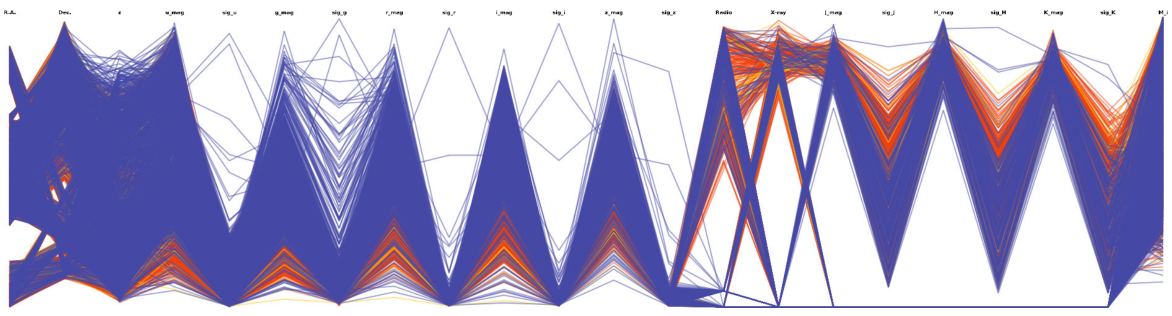
Figure 4.19: Degenerate k-means result on Haralick vectors.

4.6.3 Visualization Examples

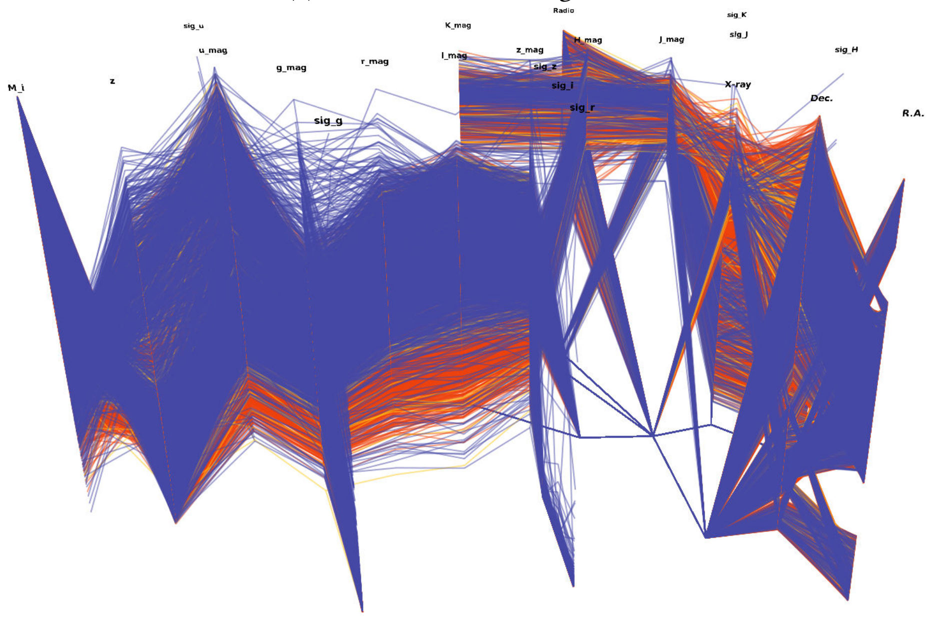
In this section, we demonstrate this novel visualization technique and its applicability to visualizing high-dimensional data sets and data mining results. Within the software tool, the view can be customized by selecting different arrangement measures as discussed before, and can be rotated and zoomed using the mouse. By using OpenGL accelerated graphics, we obtain a reasonable visualization speed even for large data sets (for even larger data sets, sampling may be necessary, but will also be expedient to get a usable visualization).

As an example data set analysis, Figure 4.18 visualizes Haralick [HSD73] texture features for 10692 images from the ALOI image collection [GBS05]. The color coding in this image corresponds to the object labels. Clearly there is some redundancy in these features, that can be intuitively seen in this visualization. Dimensions in this image were aligned using the HiCS [KMB12] measure. For a full 3D impression, rotation of course is required, which cannot be reproduced in printed media.

Visualization is an important control technique. For example, naïvely running k-means [For65] on this data set will yield a result that at first might seem to have worked. However, when visualized as in Figure 4.19, it becomes visible that the result is strict in both the attributes "Variance" and "SumAverage" – and in fact a one-dimensional partitioning of the data set. This of course is caused by the different scales of the axes. Yet, k-means itself does not offer such a control functionality.



(a) Default linear arrangement



(b) 3DPC-tree plot

Figure 4.20: Sloan SDSS quasar data set.

Figure 4.20 visualizes the Sloan Digital Sky Survey quasar data set[5]. The first plot visualizes the classic parallel coordinates view, the second plot displays the 3DPC-tree using covariance similarity. Colors are obtained by running COP outlier detection [Kri+12] with expected outlier rate 0.0001, and the colorization thresholds 90% (red) and 99% (yellow) outlier probability. The 3DPC-tree visualization both shows the important correlations in the data set centered around the near-infrared J-band and X-ray attributes, and the complex overall structure of the data set. The peaks visible in the traditional parallel plot come from many attributes in pairs of magnitude and error. In the 3DPC-tree plot, the error attributes are on the margin and often connected only to the corresponding band attribute. With a similarity threshold, they could be pruned from the visualization altogether.

[5]http://astrostatistics.psu.edu/datasets/SDSS_quasar.html

5 Improving Local Outlier Detection

Various methods to improve local outlier detection have been proposed over time (an overview of LOF variations is given in Section 3.3). Yet, these variations are largely algorithmically driven, by proposing and evaluating yet another formula, but without first trying to get a deeper understanding of the underlying mechanisms. We will first analyze LOF and some basic variations; then connect this to statistical kernel density estimation. Secondly, we inspect methods inspired by LOF for high-dimensional data. Then we will analyze the statistical meaning of the resulting outlier scores and finally derive a novel score-oriented evaluation method.

5.1 Improving the Robustness of the Local Outlier Factor

This section expands on ideas and concepts that were initially published as part of the outlier detection algorithm LoOP in:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "LoOP: Local Outlier Probabilities". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China.* 2009, pp. 1649–1652. DOI: `10.1145/1645953.1646195`

LOF [Bre+00] is one of the most cited "density-based" outlier detection algorithms (for details, see the introduction in Section 3.2). Yet it is also distance-based, since the density estimation in LOF is based on the inverse of the distance. At the heart of LOF, the local reachability density is estimated using Equation 3.6:

$$\mathrm{lrd}_{\mathrm{k}}(p) = 1\big/\left(\mathrm{mean}_{o\in N_k(p)}\,\text{reach-dist}_k(p \leftarrow o)\right).$$

I.e. the local density is estimated as the inverse of the mean local reachability distance. An occasionally overlooked detail (see [SZK12] and Chapter 6 for examples) of the formula is the reachability distance function, which is a slight misnomer, because it mathematically is not a distance function (as it is asymmetric) but a hybrid of the distance of the two objects o and p and the "core size" of the neighbor object o.

From the local densities, the outlier factor is then obtained by computing the quotient of the arithmetic mean density of the neighbors with the objects own density – or, intuitively, by comparing the expected to the actual density:

$$\mathrm{LOF}_k(o) = \mathrm{mean}_{o\in N_k(p)} \frac{\mathrm{lrd_k}(o)}{\mathrm{lrd_k}(p)} \equiv \frac{\mathrm{mean}_{o\in N_k(p)} \mathrm{lrd_k}(o)}{\mathrm{lrd_k}(p)}.$$

5.1.1 The Stabilization Effect of the Reachability Distance

The reachability distance of LOF, which emphasizes the relationship of this method to the clustering algorithms DBSCAN [Est+96] and OPTICS [Ank+99], is defined as

$$\text{reach-dist}_\mathrm{k}(p \leftarrow o) = \max\left\{k\text{-dist}(o), \mathrm{dist}(o,p)\right\}$$

which can be rewritten to a case distinction:

$$\text{reach-dist}_\mathrm{k}(p \leftarrow o) = \begin{cases} \mathrm{dist}(o,p) & \mathrm{dist}(o,p) > k\text{-dist}(o) \\ k\text{-dist}(o) & \text{otherwise} \end{cases}$$

This rewrite may seem pathetic at first, but it helps understanding the role of the reachability distance function: each objects p within the k nearest neighbors (called “core of o” in DBSCAN clustering) is treated the same way, while objects outside of the core – potential outliers – are given their true distance.

The consequences of using the reachability distance instead of the regular distance do not show up immediately: at first this function appears to (artificially) increase distances within the data set. However, the main effect is to smoothen out the difference between nearby objects. For example, if two objects p_1 and p_2 have many neighbors o_i in common, and both are in the cores of the o_i, then $\mathrm{lrd_k}(p_1)/\mathrm{lrd_k}(p_2) \approx 1$.

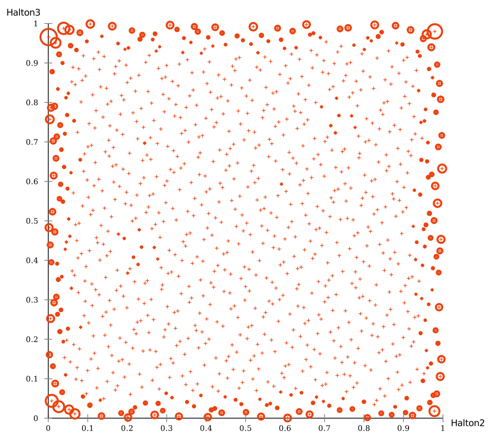
On a formal level, it can trivially be shown that if $N_k(p_1) = N_k(p_2) =: N'$ and

$$\forall_{o\in N'} \quad \mathrm{dist}(o,p_1) \le k\text{-dist}(o) \wedge \mathrm{dist}(o,p_2) \le k\text{-dist}(o)$$

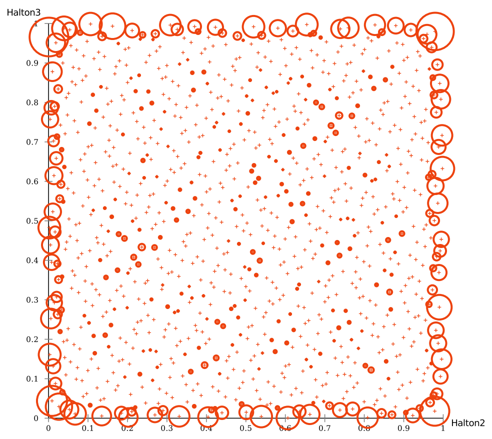
then $\mathrm{lrd_k}(p_1) = \mathrm{lrd_k}(p_2)$, even if we only have approximately the same distances $\mathrm{dist}(o,p_1) \approx \mathrm{dist}(o,p_2)$. For areas of approximately uniform density, the use of reach-dist yields more stable LOF values closer to 1. Note that for an outlier p we will usually have $\mathrm{dist}(o,p) > k\text{-dist}(o)$ for most neighbors o; and thus the lrd values will not change much. Only when other outliers are used as reference objects the lrd will be affected.

For comparison, we replace the lrd function with the following simplification to obtain an algorithm denoted as Simplified-LOF in [SZK12] and LOF' in [CF03]:

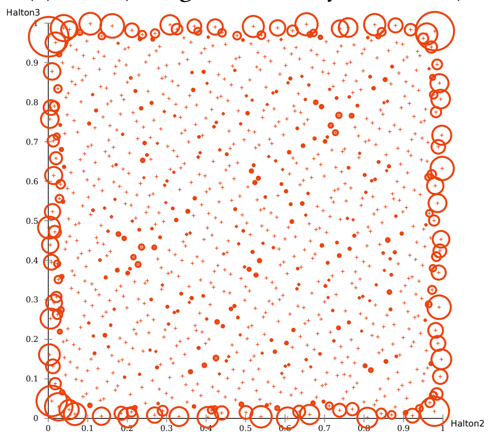
$$\begin{aligned} \text{simplified-lrd}_\mathrm{k}(p) &= 1/\left(\mathrm{mean}_{o\in N_k(p)} \mathrm{dist}(p,o)\right) \\ \text{Simplified-LOF}_k(o) &= \left(\mathrm{mean}_{o\in N_k(p)} \text{simplified-lrd}_\mathrm{k}(o)\right)/\text{simplified-lrd}_\mathrm{k}(p) \end{aligned}$$



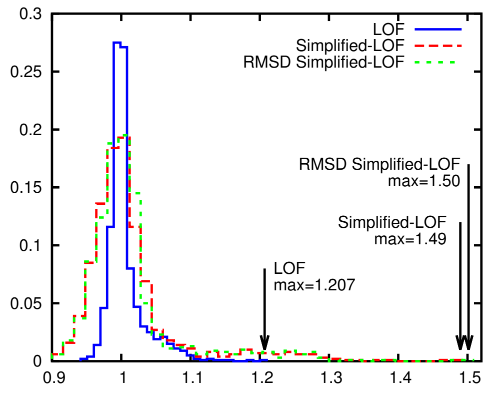
(a) LOF (using reachability distances)



(b) Simplified-LOF (using direct distances)



(c) Simplified-LOF with RMS



(d) Score histograms of LOF and Simplified-LOF

Figure 5.1: Stabilization on pseudo uniform data produced by Halton sequences.

The stabilization effect is visualized in Figure 5.1a (for LOF) and Figure 5.1b (Simplified-LOF) along with a histogram of the resulting outlier scores in Figure 5.1d. The data set is generated using Halton sequences [Hal64], low discrepancy pseudo-random sequences that appear to be uniform although in fact they are too evenly distributed for true uniform data. It can be seen that Simplified-LOF scores in particular have higher variance on this data set. The reduced variance of LOF on evenly distributed data is expected to make true outliers more pronounced, and reduce the number of false positives due to fluctuations inside a cluster. In the following sections, we will discuss two approaches for further stabilizing the results of LOF, by on one hand varying the function used for averaging values, and on the other hand by choosing a different local density estimation.

5.1.2 Stabilization with Different Averages

Instead of using the arithmetic mean in the local reachability density, we can plug in a different statistic. In the following, we will be focusing on the method called Simplified-LOF above. The reason is that both the reachability distance and alternate averages have the same goal: stabilizing the results.

The method LoOP [Kri+09a] is a LOF variation which uses a distance estimation motivated by a normal distribution and titled “probabilistic set distance”. Its definition can be simplified (the parameter λ is redundant after computing the quotient, so we leave it out) to:

$$\mathrm{pdist}(p) := \sqrt{\mathrm{mean}_{o \in N_k(p)} \mathrm{dist}(o,p)^2} \tag{5.1}$$

and is obviously the quadratic mean (see Section 2.1) distance. Root mean square (RMS) is a statistic that has been used in various disciplines to measure the *magnitude* of an error and is closely related to least squares estimation in statistics. Here, it can be interpreted as the distance which contains the majority (but not necessary all) of the neighbors.[1] Putting this density estimation into the definition of Simplified-LOF (and subtracting 1) yields the raw scores of LoOP, denoted as PLOF (for probabilistic LOF):

$$\mathrm{PLOF}_k(o) := \frac{\mathrm{mean}_{o \in N_k(p)} \mathrm{pdist}_\mathrm{k}(o)}{\mathrm{pdist}_\mathrm{k}(p)} - 1 \tag{5.2}$$

In order to obtain the final LoOP score, an additional score normalization step is needed, which we will discuss later in Section 5.3.2.2 (Equation 5.7). In the following, we will use the term Simplified-LOF with RMSD to refer to Simplified-LOF using pdist (i.e. $\mathrm{PLOF} + 1$).

The effect of using RMS distance estimation is best described as giving the nearby neighbors a lower weight, while emphasizing far away neighbors. However, at least for uniform data such as the Halton sequence data, this makes virtually no difference compared to Simplified-LOF (compare Figure 5.1c to Figure 5.1b). The reachability distance of the original LOF algorithm clearly achieves the desired result better, but we will also see results on real data sets later.

In general, any kind of mean can be used – geometric, harmonic, trimmed mean, power means, median – but one cannot expect a significantly different performance. After all, all means try to estimate the same kind of value, a kind of “typical” distance. There are two things why a more robust mean might (surprisingly) not improve outlier detection results: First of all, we are *particularly* interested in the outliers. We do not want to remove them from the data set, but to identify them. A robust mean that removes the influence of outliers can make the detection harder. Secondly, we are not working with the data coordinates here, but with the distances between data points. The distribution of distances if often much more complex than the distribution of points. With increasing dimensionality – due to the “curse of dimensionality” discussed in Chapter 4 – distances become more and more similar, and we might need to rely on subtle differences that must not be smoothened out by statistics.

[1]Compare to the “68–95–99.7 rule”, Section 1.2.

We can, however, also reinterpret the reachability distance of LOF differently. The local reachability density (Equation 3.6) can trivially be rewritten to the harmonic mean:

$$\begin{aligned}\mathrm{lrd_k}(p) &= |N_k(p)| \Big/ \sum_{o\in N_k(p)} \text{reach-dist}_k(p \leftarrow o)\\ &= \text{harmonic-mean}_{o\in N_k(p)} 1/\text{reach-dist}_k(p \leftarrow o)\end{aligned}$$

where $1/\text{reach-dist}_k(p \leftarrow o)$ is a density estimation. The harmonic mean was not an intentional choice,[2] but it is a consequence of using the arithmetic mean distance as presented in Equation 3.6 and [Bre+00]. This rewritten form offers an interesting insight into the behaviour of LOF, in particular by connecting it to yet another well known concept: kernel density estimation (see Section 2.3). Using this interpretation, we can abstract the formula to the general pattern of kernel density estimation:

$$\text{local-density}_{\text{neighborhood}}(p) = \text{some-mean}_{o\in\text{neighborhood}(p)}\text{some-density}_o(p).$$

5.1.3 Local Outliers Using Kernel Density Estimation

Contents of this section have since been published in:

> E. Schubert, A. Zimek, and H.-P. Kriegel. “Generalized Outlier Detection with Flexible Kernel Density Estimates”. In: *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA.* 2014

It is not very surprising that many of the methods discussed here are using some kind of density estimation. This relationship has already been mentioned in Section 3.1. We will now try to analyze the density estimations of LOF and Simplified-LOF as a variant of kernel density estimation (KDE). The first thing to note is that both use a different mean than KDE: The harmonic mean (whch is the power mean M_{-1}) is said to emphasize small values (densities) and reduce the influence of large values (densities). From the point of view of kernel density estimation, the arithmetic mean would appear to be much more appropriate. However, it may indeed be a desired property to lessen the influence of rare high densities, and to take low density influences more into account.

Where LOF and Simplified-LOF differ more importantly are the kernel-like[3] function they use: for Simplified-LOF, the kernel-like function used is $1/\text{dist}(x, x_i)$ while LOF uses $1/\text{reach-dist}(x \leftarrow x_i)$. In Figure 5.2 we visualize these kernel functions for $k\text{-dist}(x_i) = 1$. This figure visualizes why LOF produces more stable results than Simplified-LOF: it does not allow close objects (as in clusters) to have a strong effect on the density estimation by clipping the kernel at a certain height. Local Outlier Probabilites (LoOP) [Kri+09a] when rewritten the same way yields the M_{-2} power mean and the same kernel as Simplified-LOF; which explains why it often performs

[2] As discussed with one of the authors of LOF [Bre+00]

[3] These functions are not proper kernel functions, because they do not have the property $\int_{-\infty}^{\infty} K(x)\,\mathrm{d}x = 1$!

more similar to Simplified-LOF than LOF. The Local Density Factor (LDF) [LLP07] is motivated from kernel density estimation, and thus uses the arithmetic mean, but to mimic LOF by using a modified Gaussian kernel which is clipped to the density value at the $k\text{-dist}$. This, however, only indicates that the mechanisms of LOF were not well understood yet, by not having the kernel interpretation introduced above.

In Figure 5.2 we visualize these kernel-like functions to explain our concerns. In Figure 5.2a we plot the unclipped functions, i.e. Simplified-LOF and the Gaussian kernel. Figure 5.2b visualizes the kernel-like functions after clipping them at a maximum height of $K(1)$. For the LOF function, the clipping is expected to be beneficial, as it avoids the extreme values that would occur close to 0. For LDF however – which uses a modified Gaussian kernel – this argument does not hold. Not only is the Gaussian kernel finite, it also already has a flat top that does not discriminate small distances much anyway. Figure 5.2c tries to compare the four kernels with each other by adding a constant scaling factor that does not change relative densities. It can be seen that in the range of $[0.75; 1.5]$ the kernels are all very similar, whereas close to 0, the Simplified-LOF kernel is substantially different, not necessarily for the better.

From the theoretical background of kernel density estimation, we should use a different rescaling: Kernel functions like the Gaussian kernel should have a unit integral of $\int_{-\infty}^{\infty} K(x)\,\mathrm{d}x = 1$. However, since the integrals of the LOF and Simplified-LOF functions are infinite,

$$\int \frac{1}{|x|}\mathrm{d}x = \operatorname{sgn} x \log x \qquad \longrightarrow_{x\to\{-\infty,-0,+0,+\infty\}} \pm\infty$$

we need to truncate them artificially to obtain a finite scaling factor. For Figure 5.2d, we assumed a maximum distance of 10 and for Simplified-LOF additionally a minimum distance of $1/10$. Intuitively, this means assuming the 1-nearest neighbor is at least $1/10$ the distance of the k-nearest neighbor, and the reverse k-nearest neighbor is at most 10 times the distance. For the LDF kernel, we can actually restore the unit integral for a fixed k-dist. From Figure 5.2d we can see that LOF puts the most emphasis on the long tail, Simplified-LOF most weight on close neighbors and LDF does not differ substantially from using the real Gaussian kernel.

In Chapter 9 we will be experimenting with application specific outlier detectors that make explicit use of kernel density estimation.

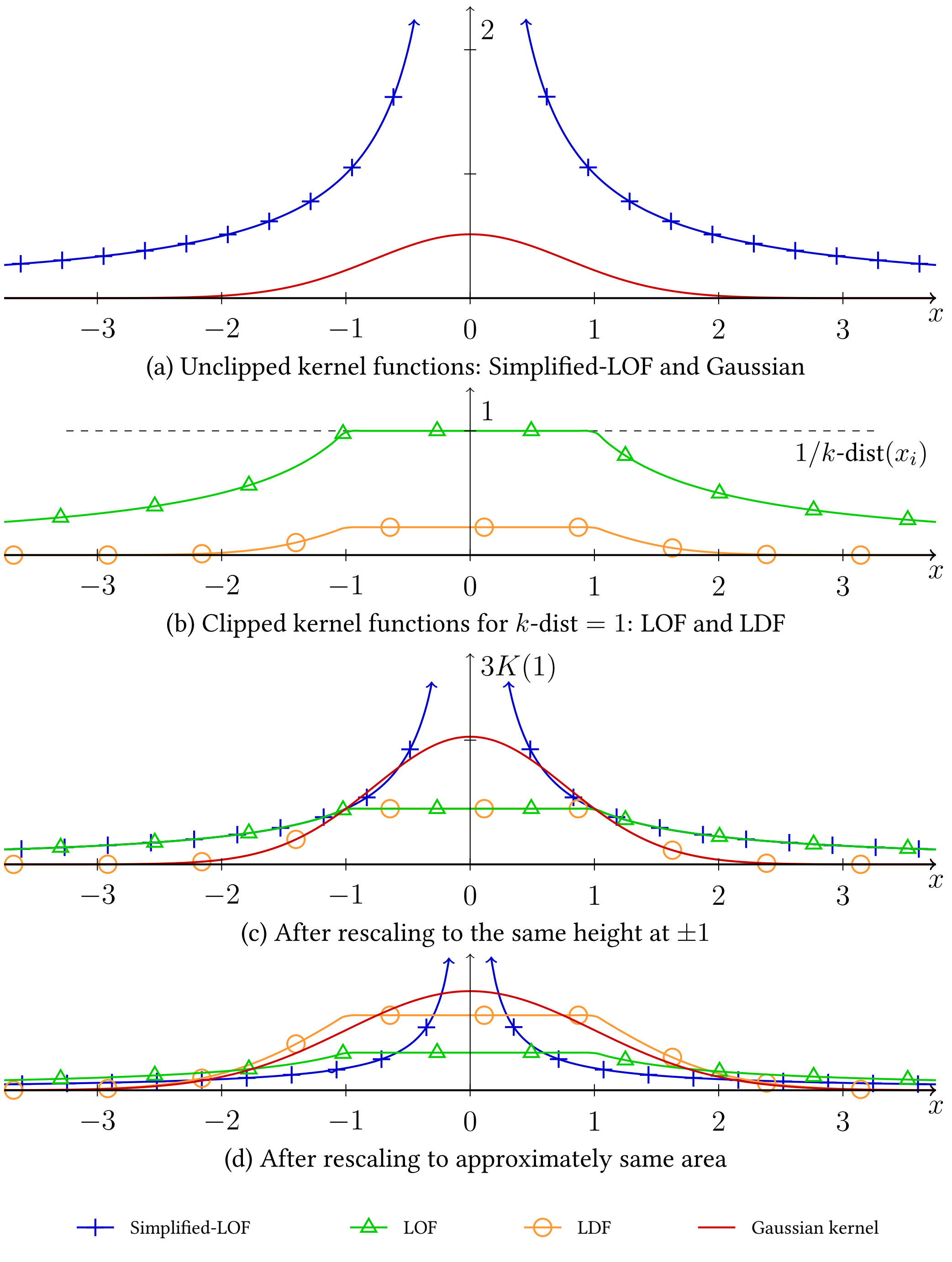


Figure 5.2: Simplified-LOF, LOF, and LDF kernel-like functions in comparison to the regular Gaussian kernel.

5.2 Local Outlier Detection in Higher Dimensions

The previously discussed methods do not work too well in high-dimensional data sets. Chapter 4 gives an overview of the reasons why the approaches that work well in 2 to 10 dimensions start falling apart in data sets of much higher (intrinsic) dimensionality.

In the following, we will discuss two methods that were developed specifically to address challenges of outlier detection in high-dimensional data. But as highlighted in [ZSK12a], this remains an open research domain with numerous unsolved challenges.

5.2.1 Detecting Local Outliers in Axis Parallel Subspaces: SOD

The method outlined in this section was published as:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data". In: *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Bangkok, Thailand.* 2009, pp. 831–838. DOI: 10.1007/978-3-642-01307-2_86

The method is based on an idea of locality inspired by LOF. Each object o is compared to its local neighbors, but SOD does not employ a simple distance function for finding neighbors, and does not use a density-based score. Since distance functions lose contrast in high-dimensional data, SOD uses the notion of shared-nearest neighbor similarity [JP73] which was shown to be more robust in high-dimensional data (see Chapter 4 and [Hou+10]) to obtain a reference set $N(o)$ for each object.

This reference set $N(o)$ of object o is then analyzed to compute a local subspace. Instead of searching through all 2^d possible subspaces, each attribute is independently compared to the total variance. Define total variance and variance in attribute i as:

$$\begin{aligned}
\mu_N &:= \mathrm{mean}_{p\in N}\, p \\
\mathrm{Var}(N) &:= \mathrm{mean}_{p\in N} \sum_{i=1}^{d} |p_i - \mu_{Ni}|^2 \\
\mathrm{Var}_i(N) &:= \mathrm{mean}_{p\in N}\, |p_i - \mu_{Ni}|^2
\end{aligned}$$

Then attribute i is considered relevant for outlier detection, if it has a variance below a threshold. Attributes are selected by constructing a weight vector ω by:

$$\omega_i := \begin{cases} 1 & \text{iff } \mathrm{Var}_i(N(o)) < \alpha \frac{1}{d} \mathrm{Var}(N(o)) \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

The threshold parameter α is relative to the expected variance $\frac{1}{d}\mathrm{Var}(N)$ if the total variance were distributed equally on all dimensions.

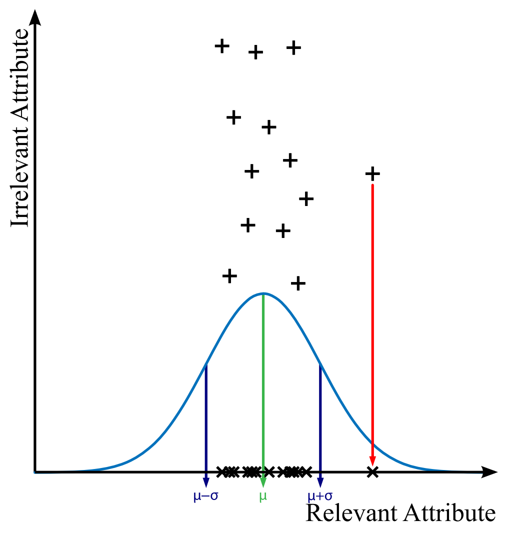


Figure 5.3: Outlier detection in local subspace projections via SOD. The outlier (red) becomes more pronounced when projected to the relevant attribute only.

The basic idea of subspaces in SOD is visualized in Figure 5.3. Axes of high variance are likely not useful for outlier detection (but represent the in-cluster position), while axes with a low variance will cluster points when projected to this subspace. Within this subspace, the distances are more useful to measure the deviation from the cluster.

While this threshold-based approach is a rather simple heuristic, the parameter α is not too hard to choose ($\alpha = 1.1$ appeared to work on a large variety of data sets). Nevertheless, there is clearly room for improving this algorithm with more clever heuristics (some of which will be discussed in Section 5.2.2). But as discussed in Chapter 4 we do actually not need to get the subspace perfectly right, but we only need to improve the signal-to-noise ratio enough to measure usable deviations.

The projected deviation of o from the mean $\mu_{N(o)}$ can then be computed by using the Euclidean distance weighted with ω, and the final SOD score is then computed by division through the number of dimensions retained (i.e. $\omega_i = 1$):[4]

$$\mathrm{dist}_{\mathrm{SOD}}(o) := \sqrt{\sum_{i=0}^{d} \omega_i |o_i, \mu_{N(o),i}|^2}$$

$$\mathrm{SOD}(o) := \frac{\mathrm{dist}_{\mathrm{SOD}}(o)}{||\omega||_1}$$

SOD does not only produce an outlier score, but the ω_i values also indicate in which subspace the object was found to be an outlier. This information can be valuable for analyzing the outliers in detail.

[4]In hindsight, a more appropriate normalization would have been $||\omega||_2$, but in Section 5.3 much more advanced normalizations will be discussed.

5.2.2 Detecting Outliers in Arbitrarily Oriented Subspaces: COP

The method outlined in this section was published as:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Arbitrarily Oriented Subspaces". In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 379–388. DOI: `10.1109/ICDM.2012.21`

Real data will often have clusters that are not restricted to axis-parallel subspaces. A common idea to detect such arbitrarily oriented clusters is to compute principal component analysis (PCA) on a local subset of the data, as done for example by the methods ORCLUS [AY00], 4C [Böh+04b], HiCO [Ach+06c] and ERiC [Ach+07]. Achtert et al. [Ach+06b] discuss the benefits of PCA for analyzing correlation clusters. A modified variant of PCA is introduced in [Kri+08] for use in correlation clustering that is less susceptible to outliers by reducing the weight of far away points and trying different k. See the survey by Kriegel et al. [KKZ09] for an in-depth overview of correlation clustering methods.

In order to solve the same problems in outlier detection – for example detecting outliers close to an arbitrarily oriented subspace cluster – a similar approach based on PCA can be employed. While this brings along the same bootstrapping problems that still plague many other correlation-based methods – in order to find the proper neighborhood in high-dimensional data, one needs to find the correlations, but in order to find the correlation one needs to already have the proper neighborhood – there are good reasons to stick to the common best practice of using the k nearest neighbors. Key benefits include that this initial neighborhood is usually unbiased (due to symmetry of the query sphere), and using index structures it can often be retrieved in reasonable computation time. Furthermore, the parameter k allows a good control of computational cost.

The method COP does not rely on a prior cluster analysis (since in particular correlation clustering itself is a rather computationally intensive task) but instead evaluates individual objects one at a time. It also does not need the correlation models of the neighbors, but the score of each object only depends on the neighbor set itself. It can therefore also be evaluated for single candidate observations only, if these have been preselected by a different algorithm.

The COP score is computed by first determining the neighbor set $N(p)$, then performing principal component analysis (PCA). For PCA, the covariance matrix Σ of the neighborhood $N(p)$ is computed and then decomposed into a rotational matrix V and a diagonal scaling matrix Λ such that $V\Lambda V^{-1} = \Sigma$. Each column v_i of the matrix V corresponds to an eigenvector and entry λ_i of the diagonal matrix Λ to an eigenvalue of the matrix Σ. Intuitively, the eigenvectors point into the principal directions of the data set, and the associated eigenvalues give the variance on this axis. We build another diagonal matrix Ω with $\omega_i := 1/\sqrt{\lambda_i}$, which obviously satisfies $\Omega\Omega = \Lambda^{-1}$. Using $\Omega = \Omega^T$ and $V^{-1} = V^T$, we can decompose Σ^{-1} as follows:

$$\Sigma^{-1} = \left(V\Lambda V^{-1}\right)^{-1} = V\Lambda^{-1}V^{-1} = V\Omega\Omega^T V^T = (\Omega V^T)^T(\Omega V^T)$$

By mapping $x \mapsto (\Omega V^T)(x - \mu)$ we can transform the original data so that the dimensions are pairwise uncorrelated and have zero mean and unit variance. Furthermore, the dimensions are ordered by their original variance. By only retaining the last d' dimensions of this space, we can measure the deviations within the subspace orthogonal to the assumed cluster (modeled by the first $d - d'$ dimensions). This is somewhat the opposite of the common approach for (global) dimensionality reduction, where one would focus on the first eigenvectors. But for outlier detection, we do not want to represent the structure of the (assumed) cluster, but instead measure the deviation from the normal. Furthermore, when doing this on a global level, the user can experimentally choose the dimensionality d' for optimal results. However, in the context of correlation clustering and correlation outliers, we must assume that different parts of the data set require different dimensionality. Therefore, we need an automatic method for choosing the dimensionality.

Heuristics for Choosing a Subspace Dimensionality: In the correlation clustering algorithm ERiC [Ach+07], the authors suggest to use a relative variance threshold $\alpha \in]0,1[$ and compute d' such that it satisfies the condition

$$d'_{\text{percentage}} := \min_{\delta} \left\{ \delta \middle| \sum_{i=1}^{\delta} \lambda_i \geq \alpha \sum_{i=1}^{d} \lambda_i \right\}.$$

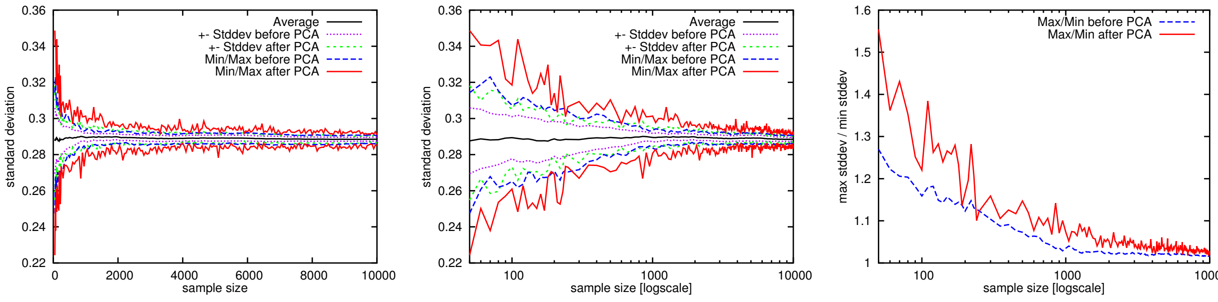
Intuitively, this states that the dimensionality should explain the relative share α of the total variance. The authors reported good results with $\alpha = 85\%$.

While SOD [Kri+12] looked at the original data dimensions only, it essentially faced the same problem of choosing a subspace dimensionality. The heuristic proposed in Equation 5.3 can trivially be rewritten for use in correlation clustering and correlation outlier detection:

$$d'_{\text{weak}} := \max_{\delta} \left\{ \delta \middle| \lambda_\delta \geq \alpha_{\text{weak}} \operatorname{mean}_{i=1}^{d} \lambda_i \right\}$$

This heuristic is based on the assumption that the variances of all dimensions should be approximately equal, and selects all dimensions that have α_{weak} times the average variance. For SOD a threshold of $\alpha_{\text{weak}} = 1.1$ worked well, but for correlation outlier detection much lower values such as $\alpha_{\text{weak}} = 0.95$ performed much better. There is a simple reason for this: while SOD used axis parallel subspaces – without changing the orientation – PCA will actively rotate the data to maximize the variance in the first dimensions, and minimize it in the later dimensions. This way, PCA actually maximizes the differences between the projected variances. For small sample sizes, there exist natural differences in variance, and PCA will unfortunately emphasize these effects. To show this effect, we generated uniform $\mathcal{U}[0;1]$ data of 20 dimensions and up to 10000 samples. We computed the standard deviations[5] before and after applying PCA. Figure 5.4 visualizes the mean standard deviation (which is not affected by PCA, and the expected value for a uniform distribution is $\sqrt{1/12.} \approx 0.289$), the minimum and maximum standard

[5]Standard deviations are easier to read than variances, but obviously they are equivalent to variance.



(a) Linear scale (b) Logarithmic scale (c) Relative

Figure 5.4: PCA emphasizes the differences in variance of the axes.

deviation and the standard deviation of the standard deviations before and after applying PCA to this (uniform) data set. In Figure 5.4c we display the quotient of maximum and minimum standard deviation. By design, PCA maximizes this difference. At around 1000 samples (for 20 dimensional data), the results stabilize, which aligns with the rule of thumb of needing $3 \cdot d^2$ samples for PCA.

Furthermore, when for example the data set contains one strong correlation, it can easily happen that the first dimension already explains 85% of the variance, and both of these heuristics would only choose this one, maybe even global, correlation. In order to take this into account, instead of comparing each eigenvalue to the total mean, we can compare it to the remaining dimensions only:

$$d'_{\text{relative}} := \max_{\delta} \left\{ \delta \middle| \lambda_\delta \geq \alpha_{\text{relative}} \operatorname{mean}_{i=\delta}^{d} \lambda_i \right\}$$

This threshold needs to be chosen larger than 1: since the eigenvalues are ordered, the largest can obviously not be smaller than the mean. In order to eliminate this parameter, we can also curry this threshold and choose the dimension with the largest coefficient, based on the idea that when going from relevant to irrelevant dimensions, the relative drop should be maximal:

$$d'_{\text{significant}} := \operatorname{argmax}_{\delta} \lambda_\delta / \operatorname{mean}_{i=\delta}^{d} \lambda_i$$

This last heuristic, however, is again easily distracted by a strong global correlation (which would cause the largest decrease to be at $\delta = 1$). Instead of fixing a dimensionality, we can also test all d dimensionalities and see when the result was best. This however meant we can no longer use the same scoring measure that was used for example in SOD (see Section 5.2.1): distances measured at different dimensionalities are not comparable. Therefore, we need to normalize the deviations in a way that makes the deviations comparable even when they have different dimensionality.

Distribution of Distances: In order to obtain a comparable value for different dimensionalities, we need to look at the expected deviations in different dimensionalities. Let us initially assume a very basic case, in which the data is approximately standard normal distributed

$X_i \sim \mathcal{N}(0;1)$ and i.i.d. in each dimension i. The Euclidean norm of d' such random variables is then $\sqrt{\sum_{i=1}^{d'} x_i^2}$. If we remove the square root and thus look at the squared Euclidean norm, we immediately get the chi squared distribution with d' degrees of freedom by definition:

$$\sum_{i=1}^{d'} X_i^2 \sim \chi^2(d') \qquad \text{if } \forall_i X_i \sim \mathcal{N}(0;1).$$

The actual Euclidean norms then of course are $\chi(d')$ distributed, but we will continue to use the squared norms instead. Chi squared distributions are well understood, since they are a special case of the Gamma distribution: $\chi^2(d') \sim \Gamma\left(d'/2, 2\right)$.

Assuming that our data were actually not normally distributed around the mean, we can try to improve results by estimating them from the data instead of using the expected value d'. There exists no known closed form solution for a maximum likelihood estimation of the parameters of the Gamma distribution; however the function is numerically stable and the parameters can be found for example via Newton's method or the algorithm described by Choi and Wette [CW69] which we implemented for our experiments. The results using estimated parameters were however not substantially different from the naïve approach solely based on the dimensionalities. We did not yet exploit the robust statistics based on the median average deviation (MAD) and L-Moments (LMM) that we used in later experiments.

Assuming that the squared deviations from the mean were thus $\Gamma(\_,\_)$ distributed, we can now compute the cumulative density function (cdf), which yields the quantile at which the potential outlier observation lies. A low percentile indicates that the observation is central, while a high value indicates it has an unusually large deviation. But most importantly, these values now are on a probabilistic scale and are comparable across different dimensionalities.

This allows us to iterate over the different dimensions and obtain the correlation outlier score (COS) as the maximum quantile (using either χ^2 or Γ distributions):

$$\mathrm{COS}(o) := \max_{\delta} \mathrm{cdf}_{\Gamma(\_,\_)}\left(d_\delta(x-\mu)^2\right). \tag{5.4}$$

In contrast to most earlier methods, this score has a strong probabilistic interpretation. However, since this does not align with the intuitive interpretation for end users, COP [Kri+12] added an additional normalization to a scale that is easier to interpret. Details on this will follow in the next Section 5.3 with Equation 5.9.

5.3 Interpreting and Unifying Outlier Scores

This section discusses material based on the following publications in a more refined way:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Interpreting and Unifying Outlier Scores". In: *Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.* 2011, pp. 13–24
>
> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Arbitrarily Oriented Subspaces". In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 379–388. DOI: `10.1109/ICDM.2012.21`

An earlier attempt at producing an intuitively usable score can be found in:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "LoOP: Local Outlier Probabilities". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China.* 2009, pp. 1649–1652. DOI: `10.1145/1645953.1646195`

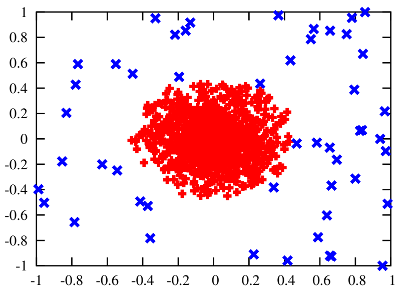
Since our intuition for normalizing outlier scores is motivated by probabilities, the desire for calibrated outlier scores is self-evident, because the concept of calibration has been used to assess the reliability of probability estimates or confidence values in classification. An optimally calibrated prediction algorithm is expected to be correct in a portion x of all predictions that are subject to a confidence value of x [Bri50]. Obtaining more or less well calibrated probability estimates has gained a lot of attention in the context of classification (e.g. [Pla00; ZE01; ZE02; Pie05]). Here, in the unsupervised context of outlier detection, we intend to use some form of calibration to reconvert plain outlier scores into probability estimates to make them both easier to understand and to improve algorithms that need to process the scores. In an unsupervised setting, however, previous approaches cannot be pursued since one cannot learn or fit a mapping but has to assume certain properties for a certain outlier algorithm. However, we can try to assess typical distributions of outlier scores and propose suitable scaling methods to convert outlier scores into probability estimates though not optimized and thus not perfectly calibrated for each data set. In general, the pure concept of calibration is questionable anyway as discussed by [MW77; DF83; Win96]. For example, a weather forecaster A is optimally calibrated in the long run if he announces every day the same chance of precipitation for an area, where that is the historical relative frequency of precipitation. His poorly calibrated colleague B, whose decisions are not strongly correlated with his announced "chance of rain" but e.g. overconfident for higher probabilities, may actually be of better help for judging whether or not it will be raining tomorrow. Indeed, in the context of classification, calibration has not always been regarded as a value in itself. In [DP96], the authors explicate (regarding the naïve Bayesian classifier), that correct classification is possible albeit based on largely erroneous probability estimates. The difference between ranking and calibration of scores is also highlighted in [ZE02].

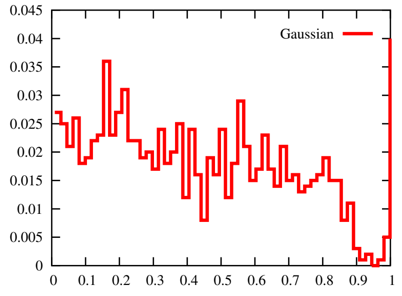
Here, however, our goal is not perfect calibration but we have two objectives that will show to be somewhat conflicting: on one hand, we want to obtain scores that allow improved reasoning when processing multiple scores for example in an ensemble. On the other hand, we also want improved interpretability of scores, for which an improved contrast between outlier scores (i.e., high outlier probability) and inlier scores (i.e., low outlier probability) is desired. The latter relates to the concept of refinement or sharpness, i.e., the forecast probabilities should be concentrated near 1 and near 0 (still being correlated with the actual occurrence of the forecast event) [San67; DF83]. It should be noted, though, that there may be cases of intermediate outlier probabilities. If these are indeed borderline cases requiring closer inspection, to assign an intermediate probability estimate to these data objects may be desirable. In the classification area, abstaining classifiers are adequate in such cases (see e.g. [Pie05]). In the outlier detection area, this problem has not found much attention in recent methods but is the genuine merit of the original statistical approaches [BL94; Haw80].

5.3.1 Interpreting Outlier Scores

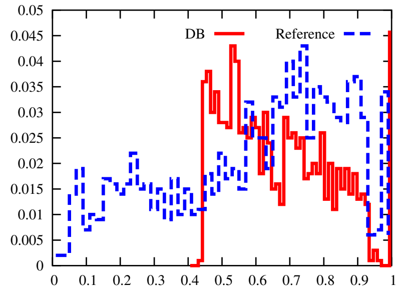
Outlier scores provided by different outlier models differ widely in their meaning, range, and contrast between different outlier models and, hence, are not easily comparable or interpretable. Histograms of outlier scores for a toy data set and multiple algorithms can be seen in Figure 5.5. The Gaussian model scores (Figure 5.5b) were at least somewhat interpretable, but the scores computed by newer algorithms have an increasingly meaningless score: the output of the scoring version of DB-Outlier ([KN98], see Equation 3.1) is naturally bounded to the domain $[0; 1]$ (representing the relative share of neighbors in the given ε radius), but will not use the whole value range. Reference-based outlier [PZG06] includes a naïve rescaling to ensure the maximum possible score is 1. kNN-Outlier [RRS00] no longer has an upper bound on the outlier score, and the values are highly data set dependent. Approaches from the LOF family (Figure 5.5e) are distributed around a mean value that is considered normal: for LOF [Bre+00] the expected value for inliers is 1, for LDOF [ZHJ09] anything between 0.5 and 1 must be considered inlier, while LOCI [Pap+03] scores again should be centered around 1, but usually will be higher than this, since the maximum deviation is used. ABOD [KSZ08] uses an inverted score – large values are the most central objects, while very small values are outliers.

This trend towards less meaningful outlier scores has reasons rooted in the way the algorithms are both used and evaluated. Obviously, in using outlier detection, the focus is on the outlier objects. Once we have found all outliers, the scores of the inliers are usually considered to be of little interest. There exists a whole array of algorithms to only find the top-n outliers for different outlier scores. HilOut [AP05] uses Hilbert curves to compute weighted kNN-Outlier scores (essentially the same approach can also be used for finding the top-n kNN-Outlier results). Jin et al. [JTH01] discuss a similar strategy for pruning LOF inliers based on micro-clusters in order to find the top-n LOF outliers only. Similarly on the evaluation side: often outlier detection methods are only evaluated with respect to their accuracy in finding the top-n known outliers, and also the very popular ROC AUC measure only considers the ranking, not the actual values of the scores. See Section 5.4 for the discussion of evaluation methods and a novel evaluation method that focuses on evaluating the scores.

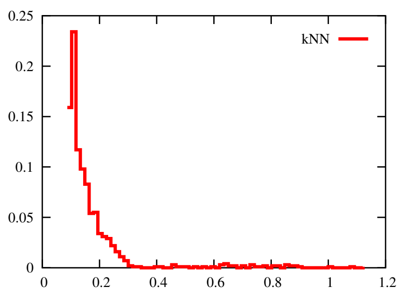

(a) Synthetic data set



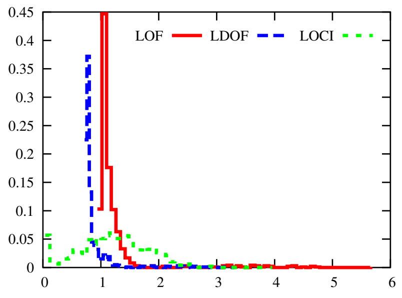
(b) Gaussian model scores



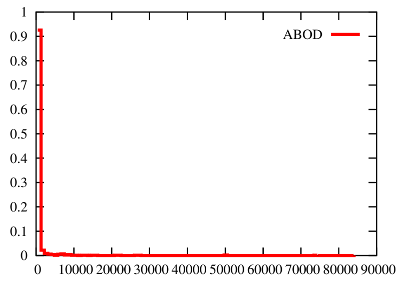
(c) DB-Outlier [KN98], reference-based [PZG06]



(d) kNN-Outlier scores [RRS00]



(e) LOF [Bre+00], LDOF [ZHJ09], LOCI [Pap+03]



(f) ABOD [KSZ08] outlier scores

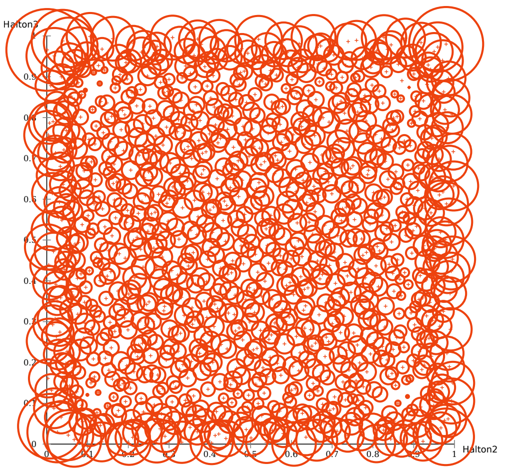
Figure 5.5: Outlier scores for several recent methods. For the data depicted in (a), the histograms (b)–(f) show bins of similar outlier scores of different algorithms along the x-axis and the relative number of objects in each bin along the y-axis.

The motivation for normalizing outlier scores to a “meaningful” scale arises from several use cases. First of all, in order to be able to combine the output of multiple methods, their scores must be on the same scale for the combination to be meaningful (see Chapter 7 for ensembles and the associated challenges). The existing methods for outlier ensembles mostly neglected this fact (see Section 3.6). Secondly, and much simpler, is the visualization use case. For a visualization such as seen in Figure 5.1a, a good visual contrast is needed. In this particular figure, we chose an empirical fixed scaling based on the observation that an outlier score of 1 or below for LOF does not need to be visually indicated (since they are detected as inliers):

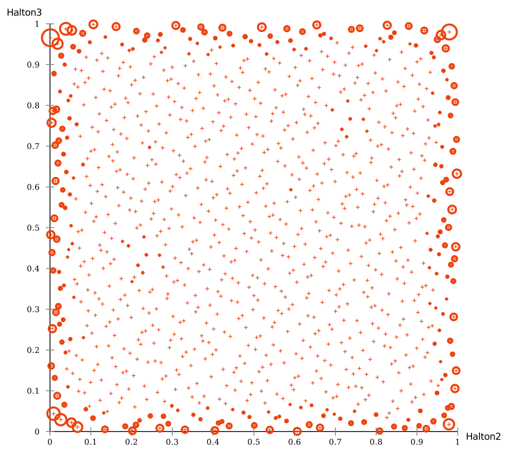
$$\mathrm{scaled}(x) := \begin{cases} 0 & x < 1 \\ x - 1 & 1 \leq x \wedge x \leq 2 \\ 1 & x > 2 \end{cases} \tag{5.5}$$

With a naïve linear scaling (mapping the minimum outlier score to 0 and the maximum to 1) the result would be much harder to interpret, as can be seen in Figure 5.6a.

Yet, the scaling proposed in Equation 5.5 is also just another naïve heuristic, and the rescaled scores are not much more “meaningful” than before. But when is a score better scaled than another? We identified the following desirable properties for a good outlier score:



(a) LOF outlier scores, naïvely scaled



(b) Figure 5.1a, repeated

Figure 5.6: Linear scaling compared to manually scaled outlier scores.

- Interpretable without context: the scores should have the same meaning independent of the algorithm and its parameters or the data set dimensionality.
- Processable: the scores must be usable by computer programs for automatic analysis such as visualization and combinations (e.g. ensembles, see Chapter 7).
- Intuitive: the scores should map to a human intuition of the numerical scale.

From these desired goals, we derive that it would be best to have scores that allow probabilistic reasoning. A *probabilistic score* ranging $[0;1]$ that intuitively gives the likelihood that an object is an outlier clearly should be interpretable without context, and should allow for example reasoning with Bayesian statistics.

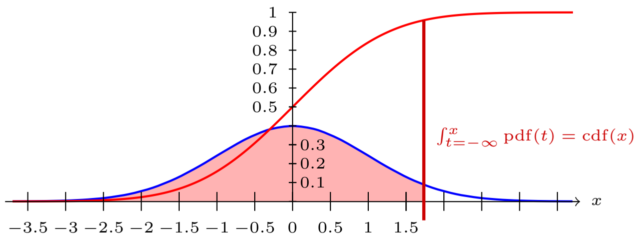
The desire of a probabilistic outlier score motivated the initial approaches for score normalization in LoOP [Kri+09a] and the refined scaling heuristics proposed in [Kri+11]. The normalization used in COP [Kri+12] is the most advanced variant of this idea, which will be detailed after introducing the basic ideas first.

5.3.2 Unifying Outlier Scores

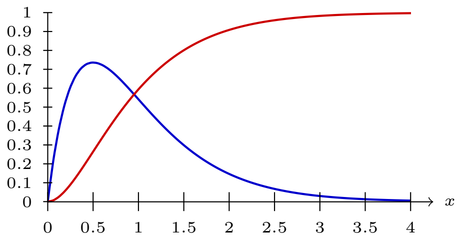
5.3.2.1 Unifying One-Dimensional Distributions

One-dimensional distributions are usually described using their probability density function (PDF) or cumulative density function (CDF), where for continuous distributions the relationship between these two simply is:

$$\mathrm{pdf}(x) := \frac{\mathrm{d}}{\mathrm{d}x}\,\mathrm{cdf}(x) \qquad \Leftrightarrow \qquad \mathrm{cdf}(x) = \int_{-\infty}^{x} \mathrm{pdf}(t)\mathrm{d}t$$



(a) Normal distribution $\mathcal{N}[0,1]$



(b) Gamma distribution $\Gamma[k=2,\theta=0.5]$

Figure 5.7: Probability density function pdf (blue) and cumulative density function cdf (red).

The pdf and cdf for two typical distributions are shown in Figure 5.7. The probability densities are ensured to be only non-negative, but the pdf can assume maximum values above 1 for distributions with a small variance, or have maximum values significantly below 1 for distributions with high variance. The cumulative density function behaves much nicer here: since the pdf has a total area of 1 (by definition), the cumulative density will start at 0 for $x \to -\infty$ and goes up to 1 for $x \to \infty$. As such, the cumulative density function (which in contrast to the pdf is also defined for non-continuous distributions) provides a much more unified view across different distributions than the pdf. In particular, for any one-dimensional distribution, we know that if $\mathrm{cdf}(x) = 0.01$; then 1% of observations are expected to be smaller than x, whereas if $\mathrm{cdf}(x) = 0.99$, 1% of observations are expected to be larger or equal x. This is already very close to the desired properties outlined in the previous subsection. It has the desired value range, is well processable, and has a probabilistic interpretation, but it turns out that this value is far from intuitive to use: the median of the distribution will by definition have a score of 0.5: half of the observations are below and half of them are above. But intuitively, an observation in the very center of the data should still have a value close to 0. For the normal distribution, there exists a transformation of the cdf that has this property, which is known as the *(Gaussian) error function*, and which can be defined as:

$$\mathrm{erf} := 2\,\mathrm{cdf}\left(x\sqrt{2}\right) - 1 \tag{5.6}$$

This transformation has the additional benefit that the function can easily be clipped to only the upper or lower half, or by using the absolute value to treat both extremes the same way. A similar transformation could be applied to the Gamma distributions, but since this distribution is not symmetric and the mean, median and mode no longer coincide it is not commonly used.

The Gaussian error function erf was a key inspiration for starting to investigate probabilistic outlier scores. Even though it later turned out there are good reasons to actually not use erf, it was proposed in the LOF variant LoOP [Kri+09a], and we will use this variant to introduce the basic ideas.

5.3.2.2 LoOP Probabilistic Score

The distribution of LOF scores observed in Figure 5.1d appears to the naïve user as "approximately normal distributed". In fact there are many reasons why this does not hold: first of all, in particular on real data, it becomes visibly positively skewed, whereas a true normal distribution is supposed to be symmetric, and secondly there are no negative values – the score distribution is bounded by 0. Nevertheless, for this early attempt we assumed that a normal distribution would be a good enough approximation.

In LoOP [Kri+09a] the idea of using the erf (Equation 5.6) to map the fraction-based outlier score to a "probabilistic score" was used the first time. The LOF-style outlier score PLOF (Equation 5.2) was centered by assuming a fixed mean value of 1; then it was rescaled using $1/(\lambda \cdot \mathrm{nPLOF})$ where nPLOF is the observed standard deviation from 1, and λ is a user dependent scaling factor with a suggested value of $\lambda = 3$. Formally,

$$\mathrm{nPLOF} := \lambda \sqrt{\frac{1}{n} \sum_{o_i \in D} \mathrm{PLOF}(o_i)^2} = \lambda M_2 \left\{ \mathrm{PLOF}(o_i) \mid o_i \in D \right\}$$

This yields a distribution that can roughly be assumed normal distributed. Finally, values below the mean are clipped to 0. The overall scaling function obtained this way is:

$$\begin{aligned} \mathrm{LoOP}(o) &:= \max \left\{ 0, \mathrm{erf}\left(\frac{\mathrm{PLOF}(o)}{\mathrm{nPLOF} \cdot \sqrt{2}} \right) \right\} \\ &= \max \left\{ 0, 2 \cdot \mathrm{cdf}\left(\frac{\mathrm{PLOF(o)}}{\lambda M_2(\mathrm{PLOF}(\_))} \right) - 1 \right\} \end{aligned} \tag{5.7}$$

which is an ad-hoc rescaling to get more desirable results, yet it was only an initial sketch of the concept of a probabilistic outlier score. Since this method was published, we developed a much deeper understanding of how scores are distributed, and how the scores can be made comparable, which we will discuss in the next sections.

5.3.2.3 Outlier Score Unification

In "Interpreting and Unifying Outlier Scores" [Kri+11] we refined and extended the rescaling approaches of LoOP to other score distributions and outlier detection models. The rescaling function was split into two steps, the first step being the *Regularization* so that inlier scores are close to 0, and outlier scores are $\gg 0$. The second step then is the actual *Normalization* that maps the outlier scores to $[0; 1]$. For some algorithms, the first step is not needed, but for example ABOD [KSZ08] has outlier scores where a value close to 0 indicates outlierness; the histogram of ABOD scores in Figure 5.5f after appropriate rescaling becomes much more regular, as can be seen in Figure 5.8a. We will, however, add a third step *Interpretation* to this, as first used in [Kri+12], which maps the already probabilistic outlier scores to the final, user-oriented score value. We also use a modified statistical scaling approach here, which has a stronger theoretical support and with improved statistical approaches.

Along with discussing different regularization and normalization functions, we will also check whether they have the property of being *ranking stable*, which we defined as either of the conditions below hold for a transformation T_S of the outlier score S:

$$\begin{aligned} \forall_{o_1,o_2}: &\quad S(o_1) \leq S(o_2) \Rightarrow T_S(o_1) \leq T_S(o_2) \\ \text{or alternatively: } \forall_{o_1,o_2}: &\quad S(o_1) \leq S(o_2) \Rightarrow T_S(o_1) \geq T_S(o_2) \end{aligned}$$

Where the second version is for handling inverted scores. Mathematically, this is the non-strict monotonicity criterion; we deliberately do not require strictness, since we do accept some loss of contrast in information for inliers (where the ranking information is not interesting) if that helps increasing the contrast for outliers.

5.3.2.4 Regularization Functions

Different scores need different transformations to become regular. In the following, we outline regularization procedures for different classes of outlier scores. Note that most of these regularization are no longer necessary when robust statistical scaling is later used. However, logarithmic inversion for example will still improve results on ABOD substantially.

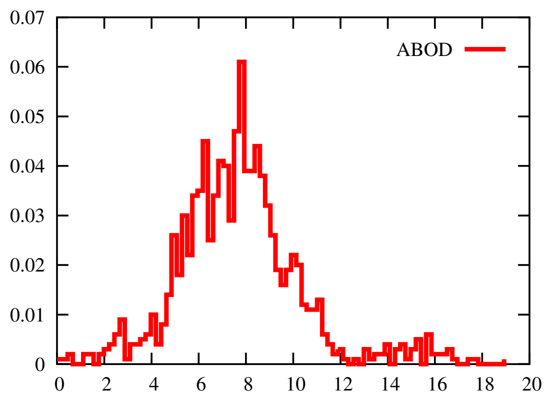
Baseline Regularization: The local outlier scores LOF, LDOF, and their variants are not yet regular, since the expected value for non-outliers is not 0. In case of LOF and its variants, the expected inlier value is $\text{base}_{\text{LOF}} = 1$. For LDOF the expected inlier score is $\text{base}_{\text{LDOF}} = \frac{1}{2}$. The expected outlier score is $\gg$ base in both cases. These scores can however be regularized with a very simple transformation. Let base_S be the baseline (expected inlier value) of the outlier score S. The idea for a regular transformation is to take the difference of the observed value $S(o)$ of an object o and the baseline value base_S. This transforms any interval $[\text{base}, \infty)$ to the interval $[0, \infty)$. Since the considered scores may also produce scores that are smaller than base_S indicating also inliers, we need some adjustment not to get negative scores after transformation:

$$\text{Reg}_S^{\text{base}_S}(o) := \max\{0, S(o) - \text{base}_S\}.$$

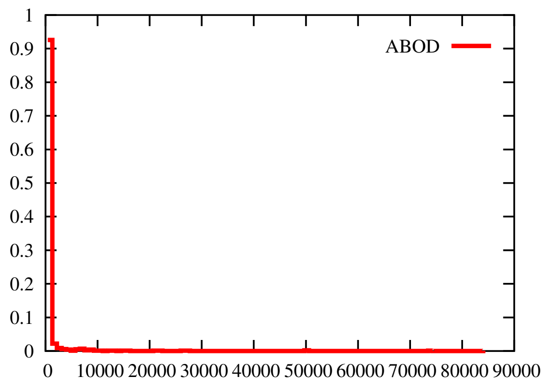
This regularization is ranking-stable:

$$\begin{aligned} S(o_1) \leq S(o_2) &\Leftrightarrow S(o_1) - \text{base}_S \leq S(o_2) - \text{base}_S \\ &\Rightarrow \max\{0, S(o_1) - \text{base}_S\} \leq \max\{0, S(o_2) - \text{base}_S\} \\ &\Leftrightarrow \text{Reg}_S^{\text{base}_S}(o_1) \leq \text{Reg}_S^{\text{base}_S}(o_2). \end{aligned}$$

In other words, if o_1 has a lower score than o_2 which means o_1 is less an outlier than o_2 for LOF, its variants and LDOF, then it cannot have a higher score after a baseline regularization. Note that for $S(o_1) < S(o_2) < \text{base}_S$ we lose information, since $\text{Reg}_S^{\text{base}_S}(o_1) = 0 = \text{Reg}_S^{\text{base}_S}(o_2)$, but this is intentional. It is also easy to see that this regularization does not enhance the contrast between inlier and outlier scores.



(a) ABOD scores, after logarithmic inversion



(b) Figure 5.5f, before logarithmic inversion

Figure 5.8: ABOD score regularization.

Linear Inversion In some outlier models, high scores are inliers. For example, if using the density function of a Gaussian model, high density identifies inliers, while a density close to 0 indicates outliers. To regularize such models, we need to invert them. For that purpose, we simply take the difference between the observed score $S(o)$ and the maximum possible or the maximum observed score $S_{\max}$.

$$\mathrm{Reg}_S^{\mathrm{lininv}}(o) := S_{\max} - S(o).$$

Since $S_{\max} \geq S(o)$, this transformation is regular. Ranking-stability for inverted scores can also easily be shown: $S(o_1) \leq S(o_2) \Leftrightarrow -S(o_1) \geq -S(o_2) \Leftrightarrow \mathrm{Reg}_S^{\mathrm{lininv}}(o_1) \geq \mathrm{Reg}_S^{\mathrm{lininv}}(o_2)$. It is also easy to see that this regularization does not enhance the contrast between inlier and outlier scoring.

Logarithmic Inversion A simple linear inversion as mentioned above is not appropriate for algorithms with very low contrast such as ABOD. A more useful regularization for ABOD that also addresses the enhancement of contrast between inliers and outliers uses the logarithm function:

$$\mathrm{Reg}_S^{\mathrm{loginv}}(o) := -\log(S(o)/S_{\max}).$$

Note that this regularization is defined only if $S(o) > 0$ for all objects o and $S_{\max}$ finite. ABOD is such a score that produces only positive values greater than zero; there is no upper bound so we have to choose $S_{\max}$ from the observed scores or bound them to a baseline S_{base}. Since the logarithm is a monotone function, the logarithmic inversion is ranking-stable. As it can be seen in Figure 5.8a, this regularization can significantly increase the contrast (compare to Figure 5.5f, repeated as Figure 5.8b).

5.3.2.5 Normalization

Linear scaling: The simplest way of bringing outlier scores onto a normalized scale is to apply a linear transformation such that the minimum (maximum) occurring score is mapped to 0 (1). A simple linear normalization can be obtained by

$$\mathrm{Norm}_S^{\mathrm{linear}}(o) := \frac{S(o) - S_{\min}}{S_{\max} - S_{\min}}.$$

Note that, if S is a regular or regularized score, $S_{\min} = 0$ can be used to simplify the formula. Obviously, this normalization does not improve contrast or interpretability of the scores significantly. Thus, we will next study possibilities to enhance the contrast between inliers and outliers based on some exemplary statistical scaling methods, which will ultimately lead to probabilistic interpretability of the unified scores.

Statistical Scaling of Outlier Scores: Outlier scores usually are not equally distributed in their value range but follow a much more complex distribution which is hard to grasp analytically. Even if we assume that our data is for example uniform or normal distributed, what is the distribution of the distances, of the distance to the k nearest neighbor, the mean distance to all kNN, and finally of the quotient of these last two values? Then enhance this solution to more than one mechanism, since we are interested in outliers generated by a different mechanism than the majority of the data set. In particular ratio distributions (and e.g. the ratio used in all LOF variations will produce such a ratio distribution) are hard to work with analytically. In many cases, the mean and other moments may no longer exist. For example in LOF, when we have k duplicate points, the LOF of these points will become undefined (due to a division by 0), cause other LOF values to become infinite and thus the mean LOF may no longer exist.

Therefore, we have to resort to analyzing the resulting score distribution, without being able to proof what the true distribution of the scores is supposed to be. On the up side, this also means we do not have to make any assumptions on the input data for such a proof to hold. Note that the linear scaling we discussed before can be interpreted as assuming a uniform distribution of the outlier scores. In practice, the scores rarely will be uniform distributed, and the resulting normalization thus not be beneficial either. Instead, we are interested in normalizations based on other well-known statistical distributions, which better resemble the observed shape of the score distribution, and whose parameters can be estimated from this data. In [Kri+11] we discussed a number of such normalizations, albeit at a rather ad-hoc level. Therefore, we want to elaborate and improve on this concept in this section, as well as provide a theoretical background for reasoning with the normalized scores to further advance this contribution.

The previous work by Gao and Tan [GT06] can be seen as closely related. They used an expectation-maximization (EM) style optimization to fit a mixture model or sigmoid function to the data, whereas we try to fit different classic distributions to the data. Since sigmoid functions naturally occur as the cdf of some distributions such as the normal and logistic distributions, one may argue that their work is a special case of our more general approach. However, there is a substantial difference in the goals when fitting the distributions: their aim was to rescale scores to get a sharp contrast from 0 and 1 – producing an almost a binary decision – while we aim at rescaling the scores to produce an almost uniform distribution. This difference results from the semantics of the score that we want to achieve. Gao and Tan aimed immediately at getting a clear binary decision of whether an object is an outlier or not, while we aim at getting a score that allows reasoning also with *intermediate* values.

The key concept of this statistical normalization has been introduced at the beginning of this section: the cumulative density function, cdf is the key ingredient to our approach. The cdf is at the same time our target and our tool in this scaling. Ultimately, we want to *estimate the value of the cumulative density function of each object*, so that the innermost objects have a value of 0, objects at an average level have a value of $.5$ and objects that are extreme have a value of 1, just as the values of the cdf on a single-dimensional normal distribution would behave. Yet, we are also going to use the cdf as our tool to achieve this, more precisely we will be using the cdf of a distribution fitted to the outlier score distribution as best estimation of the true cdf. By estimating the distributions of the outlier scores, we can then judge how usual or unusual a particular score is. The resulting score is then not only a rescaled score, but it is in fact a cdf estimate – a value that sits in the very heart of probability theory and allows the reuse of many statistical arguments that were unavailable to be used with outlier scores before.

Since we cannot model the true distribution of the outlier scores analytically, we will instead try to *empirically* fit a number of popular distributions to the observed data. However, since we are in the context of outlier detection, we need to pay attention to the robustness of our fitting methods. The early attempts we published in [Kri+11] did not yet take this into account, and were much less successful in normalization than the approaches we will discuss here, both by taking additional distributions into account as well as using more robust estimators.

Distributions that we used in our experiments include:

- Normal distribution $\mathcal{N}[\mu;\sigma]$
- Log-normal distribution $\mathcal{LN}[\mu;\sigma]$ (with $X \sim \mathcal{LN}[\mu;\sigma] \Leftrightarrow \log X \sim \mathcal{N}[\mu;\sigma]$)
- Shifted log-normal distribution $\mathcal{LN}[\mu,\sigma] + \lambda$
- Generalized normal distribution with an additional skew parameter (which also generalizes the log-normal distribution) $\mathrm{GNO}[\mu,\sigma,s]$
- Gamma distribution $\Gamma[k;\theta]$ (which includes the χ^2 distribution: $\chi^2[\nu] \sim \Gamma[\nu/2;2]$)
- LogGamma again in a shifted variant $X \sim \mathcal{L}\Gamma[k;\theta;\lambda] \Leftrightarrow \log(X-\lambda) \sim \Gamma[k;\theta]$
- Weibull distribution $\mathrm{Weibull}[\lambda,k]$
- Gumbel distribution $\mathrm{Gumbel}[\mu,\beta]$
- Generalized Extreme Value (GEV) distribution, also known as Fisher–Tippett distribution, which generalizes the Fréchet, Gumbel and reversed Weibull distributions $\mathrm{GEV}[\mu,\sigma,\xi]$
- Exponential distribution, including a shifted variant $\mathrm{Exp}[\beta,\lambda]$
- Laplace or Double-Exponential distribution $\mathrm{Laplace}[\mu,b]$
- Exponentially modified Gaussian distribution (a mixture of a Gaussian and an exponential distribution): $X = (Y+Z) \sim \mathrm{ExG}[\mu,\sigma,\lambda] \Leftrightarrow Y \sim \mathcal{N}[\mu;\sigma] \wedge Z \sim \mathrm{Exp}[\lambda]$)
- Cauchy distribution $\mathrm{Cauchy}[x_0,\gamma]$

- Logistic distribution $\mathrm{L}[\mu, s]$
- Generalized logistic distribution $\mathrm{GLO}[\mu, \alpha, k]$
- log-Logistic distribution $\mathrm{LL}[\alpha, \beta]$
- Inverse Gaussian (Wald distribution) $\mathrm{IG}[\mu, \lambda]$
- Uniform distribution $\mathcal{U}[a, b]$

Some distributions used elsewhere in this thesis were not used in this experiment because of either having finite support only – such as the beta distribution – or being discrete distributions such as Poisson and Bernoulli. These distributions – as well as the many others we did not experiment with – could however be included if desired, since our approach can be combined with an arbitrary distribution where reasonably good parameter estimators are available.

For these distributions we evaluated a number of parameter estimation techniques, with special attention to robust estimators. For example for the normal distribution, the usual maximum likelihood estimation (MLE) via $\hat{\mu} = \mathrm{mean}(X)$ and $\hat{\sigma}^2 = \mathrm{mean}_{x \in X}(x - \hat{\mu})^2$ is known to be sensitive to values from outliers. Among the estimators we experimented with were:

- Method of moments (MOM) based estimators, including the common MLE estimates for normal, Gamma and exponential distributions, the inverse Gaussian distribution and the ExGaussian estimator by Olivier and Norberg [ON10].
- Estimators based on median and median average deviation (MAD) for normal [Ham74], Gamma [CR86], Gumbel [Oli98], exponential [Oli98], Laplace [Oli98], Cauchy [Oli98], logistic [Oli98], log-logistic [Oli98] and uniform distributions.
- Method of moment and MAD estimators in logspace, for estimating the log-normal, log-Gamma and Weibull distributions.
- A slight variation of these estimators proposed in [Kri+11] using a fixed mean μ set to the expected inlier value (e.g. $\mu = 1$ for LOF), with MOM and MAD type of estimators.
- L-Moments (LMM, using probability weighted moments [HWW85; Hos91]) estimates for normal [Hos91], log-normal [Hos91; Bil12], Gamma [Hos91], exponential [Hos91], GEV [HWW85], logistic [Hos91], Generalized logistic [Hos91], Gumbel [Hos91], Laplace, skewed generalized normal [Hos91], Weibull and uniform distributions.
- Choi and Wette [CW69] maximum-likelihood estimation for the Gamma and log-Gamma distributions.
- Minimum and maximum observed value for the uniform distribution.
- Levenberg-Marquardt (LM) [Lev44; Mar63] numeric optimization for the normal and log-normal distributions (a modification of the Gauss-Newton gradient descent method, and substantially more expensive than the other statistical parameter estimations).

To further improve the robustness, we experimented with trimming (removing the most extreme observations) and Winsorization [Has+47] (replacing the most extreme values with less extreme data points).[6] This yields a total of 370 distribution estimates computed for each data

[6]Note that Winsorizing does not affect median and MAD based estimators, and also the combination of trimming with MAD estimators is not expected to substantially improve results.

set. However, not all estimators succeed on each data set: L-Moment estimates may bear inadmissible sign, and the logarithm is not defined for $x \leq 0$. When estimating the statistical moments on large data sets, we need to pay attention to numerical stability. The naïve estimation of the variance using $E(X^2) - E(X)^2$ – while widely used in textbooks and popular algorithms such as BIRCH [ZRL96] – is prone to catastrophic cancellation. Instead, we implemented the approaches discussed by Welford [Wel62], Terriberry [Ter08] and Pébay [Péb08], which we will also extend to weighted covariance in Appendix 1 for other purposes. There are other numerical problems involved. For example, since $\log(0)$ is not defined, some of the parameter estimation methods will fail for methods such as kNN-Outlier and real data that contains duplicates. One can define $\log(0) := -\infty$, or use an extreme (but finite) negative float value. But again, these values will yield numerical instability, in particular when estimating the parameters of shifted log-based distributions. However, doing the optimal estimation while avoiding these numerical issues is beyond the scope of this thesis.

Further methods worth investigation include, but are not limited to: The S_n and Q_n estimators of scale by Rousseeuw and Croux [CR92; RC92; RC93], which have the same 50% breakdown that makes MAD attractive, but have a higher Gaussian efficiency. Cohen [Jr51] discusses parameter estimation for the three-parameter log normal distribution. Nelder and Mead [NM65] published a general optimization method known as downhill simplex method, that can be used for optimizing distribution parameters as an alternative to the Levenberg-Marquardt method we evaluated on normal distributions. These approaches could also be used to refine the parameters of other distributions (however it should be noted that the robust estimators worked better than the L-M optimized distributions). Additional distributions that could be of interest include but are not limited to Snedecor's F distribution, Erlang distribution, inverse generalized exponential (IGE) distribution and Kappa distribution. 4- and 5-parameter distribution families are expected to be able to model the data better, while still not being prone to overfitting due to the low degrees of freedom. However, we cannot expect to have a perfect match on real data with outliers.

Notice that there is an obvious choice that we deliberately left out:
we could also use the empirical cdf function, which can be defined as

$$\text{empirical-cdf}_X(x) := \mathrm{rank}_X(x)/(n+1).$$

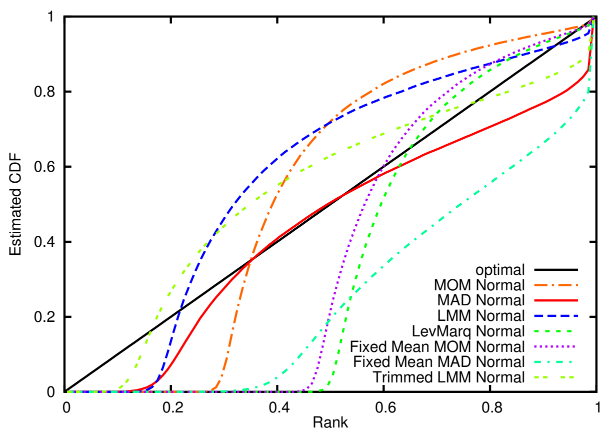
While this function will (except for ties) map the data to a perfect uniform distribution, it is highly dependent on the data, and thus prone to overfitting. In particular, this function will always map one object to the maximum cdf of 1, and not be able to abstain. Because we want to avoid this kind of overfitting, we restrict ourselves to use scaling functions with few degrees of freedom. The distributions we chose for our analysis had up to three degrees of freedom, and will thus not be able to overfit.

In order to test the quality of these rescalings, we employed three data sets: two baseline data sets consisting of 10000 samples from a 2-dimensional uniform respectively a 2-dimensional standard normal distribution. The third data set is the ALOI image data set used in multiple experiments throughout the thesis. For the normal distributed data set, we also include a "true cdf"

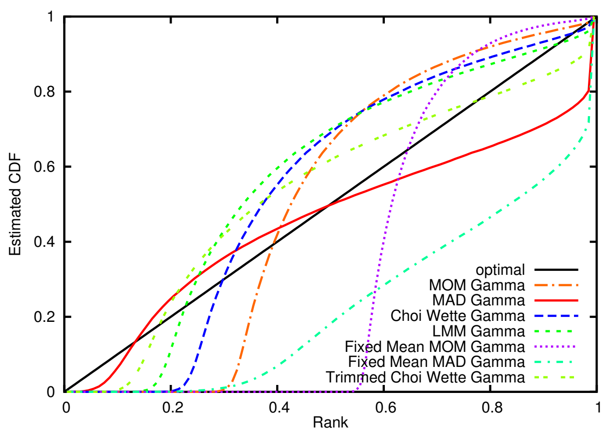
method, which uses the cdf of the generating distribution and which we expect to be approximately uniform distributed. As test statistic we use the Kolmogorov-Smirnov-Lilliefors [Kol33; Smi48; Lil67] test statistic, i.e. the maximum deviation from the desired uniform distribution $\mathcal{U}[0;1]$. We do, however, not perform a full K-S-test (or Lilliefors test) – none of the tested distributions will fit well enough to satisfy a reasonable α significance threshold. Instead, we want to get a rough impression of how well the various parameter estimations and distributions fit to the observed data and to choose a “best effort” rescaling. In the next section we will then discuss more refined evaluation measures for the purpose of outlier detection, while the K-S-statistic serves primarily as an *unsupervised* indication of which scaling can be expected to work better than the other. Again, we could as well have used other test statistics such as Anderson-Darling test [AD52], the Cramér-von Mises criterion or Shapiro-Wilk test [SW65]. While one of the weakest tests, the K-S-statistic is also very general (it can be used with arbitrary distributions) and efficient to compute – and strong enough to actually *reject* all distributions tested: as noted before, we cannot expect real data to exactly follow an ideal distribution.

Figure 5.9 shows Probability-Probabilty-plots (P-P-plots) for assessing the quality of fit for different distributions, as estimated using the methods discussed above. The x axis is the true quantile of the observed data, while the y axis is the estimated quantile using the distribution. On a perfect match, the two values will agree and produce a diagonal line. Intuitively, the K-S-statistic is the maximum vertical deviation in this plot – a simple worst case estimate for the quality of the fit. In Figure 5.9a we can see the quality for normal distributions. Obviously, the common method-of-moments estimation (using the sample mean μ and sample standard deviation σ) does not fit the data very well. The fixed-mean approaches we suggested in [Kri+11] are not substantially better. The probably best fit is obtained using the robust MAD estimator, but still does not fit the data well. Similar results can be observed for Gamma distributions in Figure 5.9b: neither the native estimation nor the fixed mean fit the data well, and again the MAD estimator is substantially better. However, if we widen up the search space to other distributions such as the three parameter log-normal distribution, and use robust estimations with Winsorization and L-Moments, we can model the data substantially better. In Figure 5.9c, some of the lines are visibily close to the diagonal line. For completeness, Figure 5.9d and Figure 5.9e visualize the best fits for exponential family and other distributions, while Figure 5.9f repeats the best results across different distribution families.

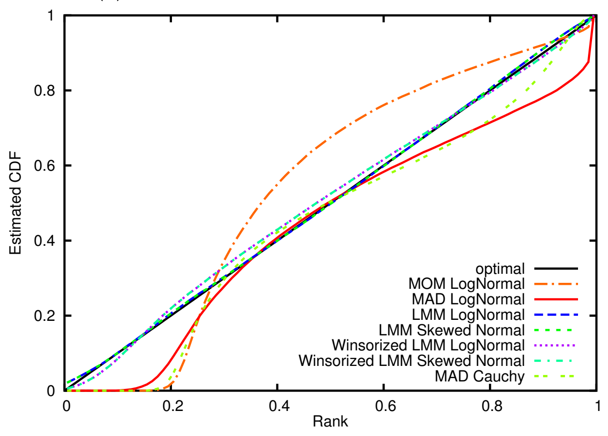
A different view on the quality of fit can be obtained by plotting a histogram of the actual data as opposed to the pdf of the fitted distribution. The results can be seen in Figure 5.10a. Again, the naïve MOM estimation for the normal distribution can easily be recognized as a bad model of the data, whereas the log-normal distribution is a reasonably close match. Figure 5.10b then is the opposite transformation: this chart shows histograms of the transformed data using the cdf function for each distribution. As discussed before, the optimal transformation would yield a uniform distribution (indicated by the black line at density 1). Again the log-normal distribution produces the best result. All curves fit best in the central area, with some error on the low end, and a spike in the end. Since the data is real – and does contain outliers – the spike at the very end probably is something that we will not be able to get rid of: after all, we assume that there are some objects in the data set that do not belong to the same distributions. If the fit



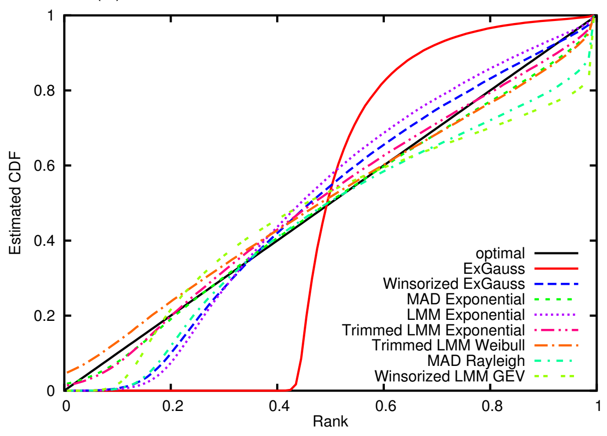
(a) Normal distribution estimators



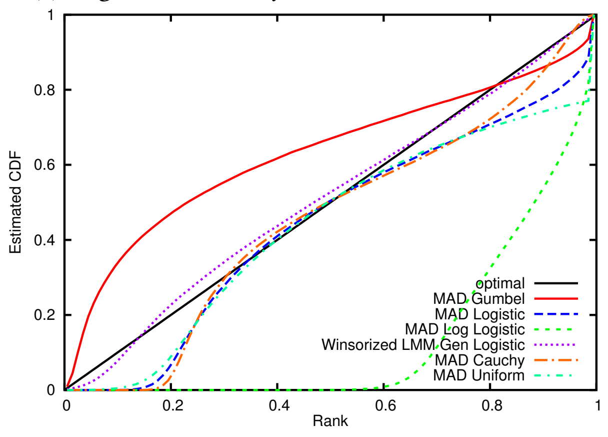
(b) Gamma distribution estimators



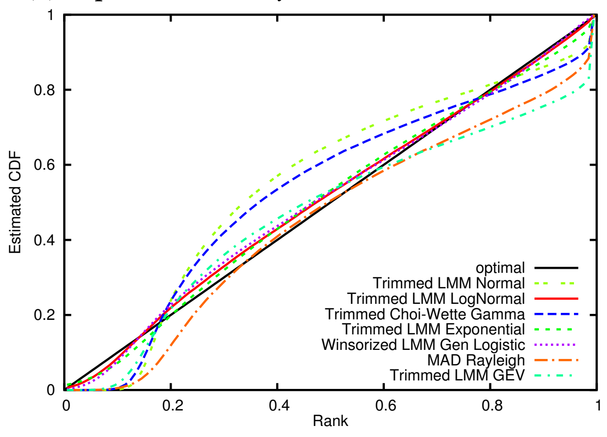
(c) LogNormal family distribution estimators



(d) Exponential family distribution estimators

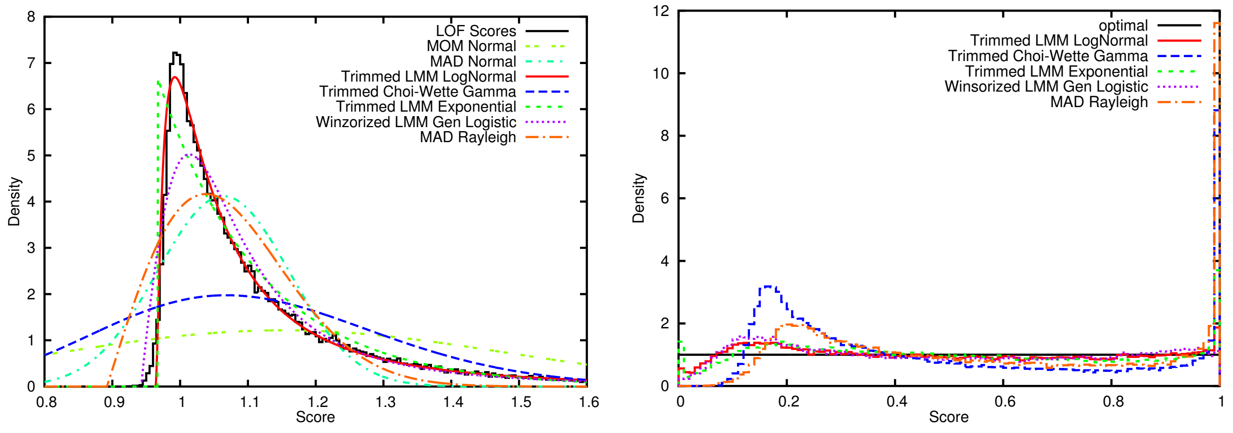


(e) Other estimators



(f) Best fitting distributions

Figure 5.9: Probability-Probability-Plots of fit for different distributions (LOF, ALOI).



(a) Histogram and pdf curves (b) cdf Histogram

Figure 5.10: Quality of fit for different distributions (LOF, ALOI).

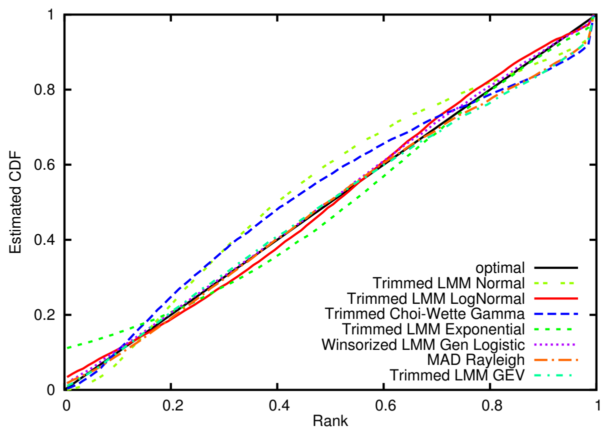
were perfect, there would be a reason to doubt the existence of outliers. Yet, we must be aware that it will likely overestimate the outlierness of objects.

For control, we also performed the same analysis on synthetic data, both using 2-dimensional normal distributed data as well as uniform data. The results are summarized in Figure 5.11, giving the best matching distributions, their pdf and the transformed data after applying the cdf function. Interestingly, the results are not substantially different from those obtained on real data. While the generalized logistic distribution now provided the best fit, the log-normal distribution was a close second best approximation. For the other quotient based distributions the results are also similar; only kNN-Outlier scores were best modeled using the exponential distribution instead. While the parameter estimation for the Rayleigh and GLO distributions worked well for synthetic data, the results for the ALOI data set were not competitive with the log-normal distributions.

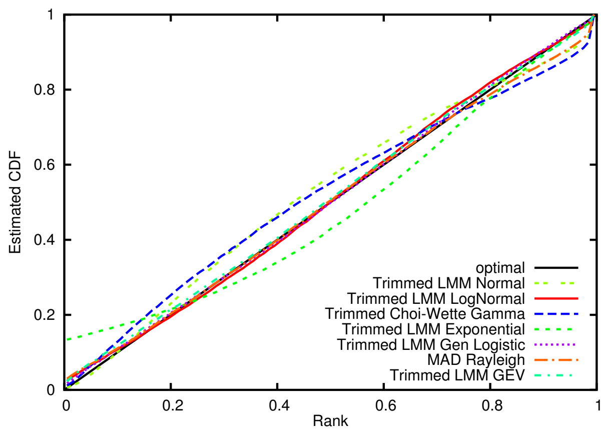
Fortunately, the quality of the transformation can be empirically determined using the K-S-statistic mentioned before. This statistic (which is used by the Kolmogorov–Smirnov [Kol33; Smi48] and Lilliefors tests [Lil67]) is easy to compute and does not make assumptions on the distribution function. Formally, it is given as the maximum deviation of the empirical and the estimated cumulative density functions:

$$d_{KS} := \max_x |\text{empirical-cdf}_X(x) - \text{estimated-cdf}(x)|$$

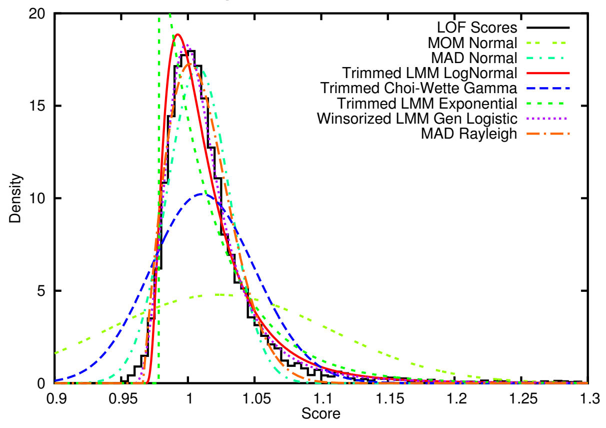
In an unsupervised context such as outlier detection, the best we can do is to choose the distribution which most closely matches our observed data. By only allowing distributions with few degrees of freedom, we can largely prevent overfitting in this step. If at the same time, the d_{KS} statistic is reasonably low, we can expect it to improve results later in the processing pipeline; even when the hypothesis that the data is really distributed this way must be rejected by reasonable significance levels.



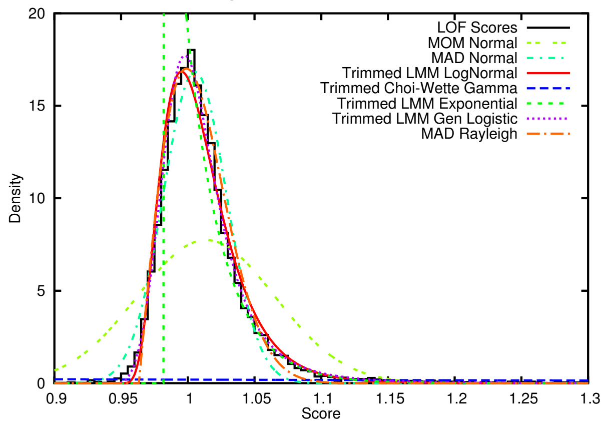
(a) Best fitting distributions (Gaussian)



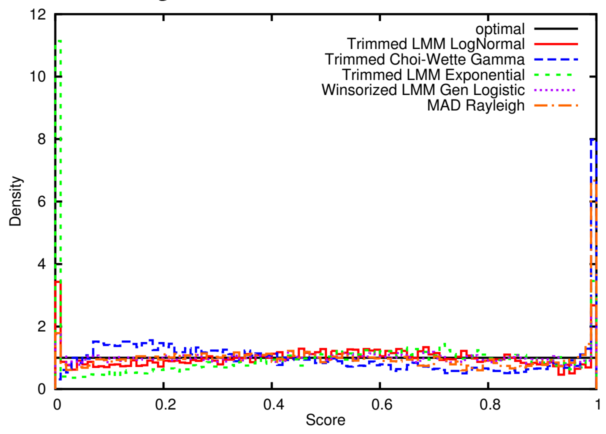
(b) Best fitting distributions (Uniform)



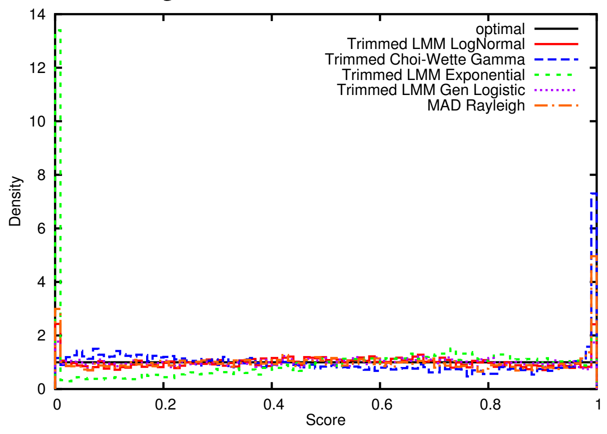
(c) Histogram and pdf curves (Gaussian)



(d) Histogram and pdf curves (Uniform)



(e) cdf Histogram (Gaussian)



(f) cdf Histogram (Uniform)

Figure 5.11: Quality of fit for different distributions (LOF, 2d Gaussian / Uniform).

In Table 5.1 we summarize the maximum deviations obtained as K-S test statistics. In the last column, we give the best estimation found along with the K-S statistic. Results within $.001$ of this best result are marked in bold. The first columns give the results for the naïve estimations we used in the initial publication of this research. It can be seen that while the method of moments based rescalings improved the error over the naïve uniform distribution, the newer rescalings using the method of L-Moments (LMM) and advanced distributions such as generalized normal and generalized logistic yield much smaller errors. The last row serves control purposes: here we have a known normal distribution, and can compute the true cdf values. The resulting scores are uniformly distributed by generation, and the K-S statistic of $.006$ does not allow rejecting this hypothesis. We can roughly treat this value as a typical K-S error on this data set size.[7] As we can see, the other best fitting distributions produce an error on a similar magnitude of .01 on the K-S statistic, which indicates that we have found a reasonably good standardization for our purposes.

5.3.2.6 Bayesian Scaling for Scores

In the previous section, we put a lot of effort into normalizing the outlier scores to be approximately uniformly distributed. Measuring the outlierness of an object with respect to the objects cdf bears strong mathematical semantics: if an object has a normalized outlier score of $x = 1 - p$, this indicates that this amount of objects is to be considered *more normal* – i.e. if an object has a score of $x = .99$, then it is considered more unusual than 99% of objects.

Statistically, this score is what is known as p-values; however, p-values are commonly misunderstood in various different ways (twelve of which are listed e.g. in [Goo08]). In particular, they are *not* the probability that the object is an outlier, but the probability that a normal observation is as extreme as this probability. Therefore, the user may have something different in mind when he is thinking of the "outlier probability" of an object. Intuitively, users may assume that an object that is assigned a score of 50% should have a 50–50 chance of actually being an outlier – but given uniform scores, 50% of the objects are expected to have a higher score than 50%, but outliers obviously should be rare instances only. In COP [Kri+12] we proposed an ad-hoc rescaling of these scores based on inspiration from statistical testing. What a user may intuitively be thinking of is a statistical test, comparing the hypothesis "the observation is caused by a different mechanism" to the alternative "the observation is regular". We cannot perform a proper statistical test here, as it is not obvious how to define the hypotheses mathematically. The approach proposed to be used with COP is a simple heuristic, that requires a user parameter: the *expected* rate of outliers, φ. To obtain the intuition discussed above that an object with a score of 50% should have a 50–50 chance, we want to have outliers where $1 - p = \varphi$ to be mapped to 0.5. This parameter φ can also be seen as a kind of significance

[7]Note that the ALOI data set is much larger than the artificial data sets used here.

Table 5.1: Goodness of Fit for different algorithms, parameter estimations and data sets.

Outlier Method	Data Set	Normal MOM	Log-normal MOM	Gamma MOM	Uniform	Best Log-normal	Best gen. logistic	Automatic Best Fit		
LOF [Bre+00]	ALOI	0.278	0.189	0.301	0.914	**0.010**	0.036	0.010	Trimmed LMM	Generalized normal
	Normal	0.260	0.225	0.257	0.872	0.029	**0.016**	0.016	Winsorised LMM	Generalized logistic
	Uniform	0.183	0.158	0.178	0.738	0.018	**0.012**	0.012	Trimmed LMM	Generalized logistic
Simplified-LOF [SZK12]	ALOI	0.258	0.114	0.295	0.942	0.013	**0.003**	0.003	Trimmed LMM	Generalized logistic
	Normal	0.123	0.079	0.111	0.798	0.014	**0.010**	0.010	Winsorised LMM	Generalized logistic
	Uniform	0.120	0.079	0.106	0.691	0.013	**0.009**	0.009	Winsorised LMM	Generalized logistic
Simplified-LOF with RMSD [SZK12]	ALOI	0.221	0.117	0.239	0.917	0.013	**0.004**	0.004	Trimmed LMM	Generalized logistic
	Normal	0.137	0.091	0.127	0.806	0.014	**0.011**	0.011	Winsorised LMM	Generalized logistic
	Uniform	0.122	0.083	0.111	0.677	0.014	**0.010**	0.010	Winsorised LMM	Generalized logistic
kNN-Outlier [RRS00]	ALOI	0.177	0.046	0.051	0.778	0.028	0.049	0.010	Trimmed LMM	Exponential
	Normal	0.252	0.124	0.295	0.821	**0.018**	0.032	0.018	Trimmed LMM	Generalized normal
	Uniform	0.088	0.051	0.074	0.602	0.014	**0.009**	0.009	Winsorised LMM	Generalized logistic
kNN-Weight [AP02; AP05]	ALOI	0.182	0.051	0.055	0.807	0.030	0.051	0.010	Winsorised LMM	Exponential
	Normal	0.257	0.116	0.309	0.843	**0.019**	0.022	0.019	Winsorised LMM	Generalized normal
	Uniform	0.094	0.046	0.078	0.689	**0.013**	**0.013**	0.013	Winsorised LMM	Generalized normal
Simplified-LOF with kernel density	ALOI	0.159	0.383	0.210	0.279	0.223	0.100	0.078	Trimmed LMM	Generalized normal
	Normal	0.066	0.198	0.117	0.182	0.104	0.064	0.053	Winsorised LMM	Generalized normal
	Uniform	0.060	0.181	0.105	0.147	0.129	0.070	0.056	LMM	Normal
LDF [LLP07]	ALOI	0.269	0.291	0.309	0.472	0.278	0.264	0.240	Trimmed LMM	Generalized normal
	Normal	0.332	0.238	0.362	0.904	0.026	**0.014**	0.014	Trimmed LMM	Generalized logistic
	Uniform	0.304	0.209	0.325	0.879	0.018	**0.010**	0.010	Trimmed LMM	Generalized logistic
True CDF	Normal	0.059	0.157	0.095	0.009	0.057	0.070	0.006	Winsorised LMM	Uniform

threshold as often used in statistical testing. An example heuristic that has this property is:

$$\text{adjust}(p,\varphi) := \frac{1}{1+\dfrac{1-p}{\varphi}} = \frac{\varphi}{\varphi+(1-p)} \tag{5.8}$$

The result of this transformation when applied to the cdf of normal and Gamma distributions can be seen in Figure 5.12a and Figure 5.12b. While the initial goal is achieved as desired – at $1-p=\varphi$, so when the observed extremeness matches the desired extremeness the adjusted score is 0.5 – it is not yet fully satisfactory, since the values will never become 0, but are always at least $\varphi/(1+\varphi) > 0$ (the maximum is 1). This can be corrected with linear rescaling to fully use the desired value range of $[0;1]$ as seen in Figure 5.12c and Figure 5.12d.

$$\begin{aligned}\frac{\varphi}{(1-p)+\varphi}\cdot(1+\varphi)-\varphi &= \frac{\varphi+\varphi^2}{(1-p)+\varphi} - \frac{\varphi-\varphi\cdot p+\varphi^2}{(1-p)+\varphi}\\ &= \frac{\varphi\cdot p}{(1-p)+\varphi} =: \text{norm}_{\text{COP}}(p,\varphi)\end{aligned} \tag{5.9}$$

This modified formula no longer has the property that $1-p=\varphi$ maps to 50%, but the result $(1-\varphi)/2$ will be close enough to 0.5 for reasonably small values of φ. The values 0 and 1 however are now mapped to 0 and 1. A different visualization of this transformation can be seen in Figure 5.14a for $\varphi \in \{50\%, 10\%, 1\%, .1\%\}$ showing how the transformation increases the contrast for large values. This is the score rescaling introduced in COP [Kri+12]; the initial score computed, the Correlation Outlier Score COS (Equation 5.4), is a variant of the unmodified cdf we obtained by normalization in the previous section.

However, we must remember that the scaling introduced in COP is an *heuristic* approach. The original values had an interpretation similar to p-values, and p-values are often misinterpreted [Goo08]. The semantics of this transformed score are thus weaker than that of the approximately uniform distributed scores. An alternate approach is based on the Bayes' factor, which arises from Bayes' rule for dichotomous problems in odds scale [KR95]:

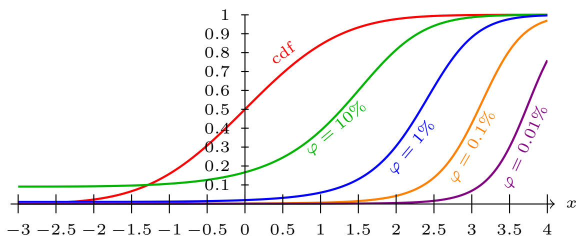
$$\text{Odds}(A:\neg A) = \frac{P(A)}{P(\neg A)} = \frac{P(A)}{1-P(A)} \tag{5.10}$$

with the inverse transformation being

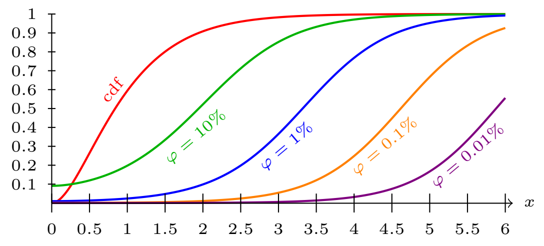
$$P(A) = \frac{\text{Odds}(A)}{1+\text{Odds}(A)} \tag{5.11}$$

By applying Bayes' theorem to A and $\neg A$ for evidence E

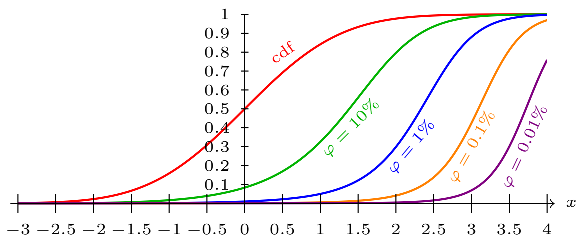
$$P(A|E) = \frac{P(E|A)\cdot P(A)}{P(E|A)\cdot P(A)+P(E|\neg A)\cdot P(\neg A)}$$



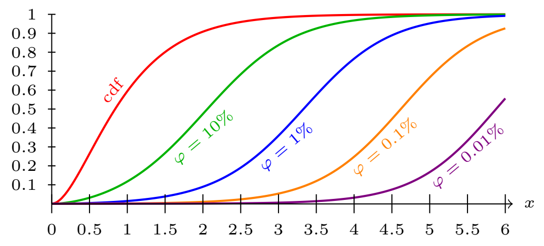
(a) Normal cdf, without correction (Equation 5.8)



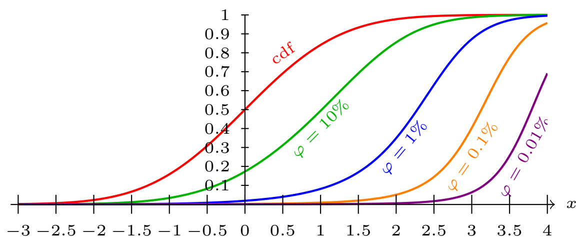
(b) Gamma cdf, without correction (Equation 5.8)



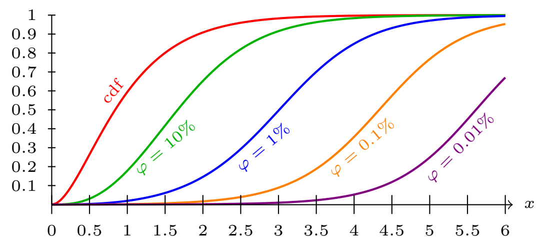
(c) Normal cdf, with correction (Equation 5.9)



(d) Gamma cdf, with correction (Equation 5.9)



(e) Normal cdf, with Bayes factor (Equation 5.15)



(f) Gamma cdf, with Bayes factor (Equation 5.15)

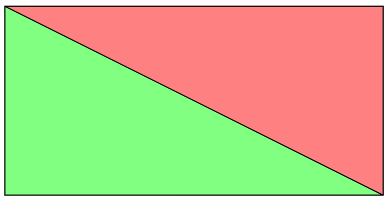
Figure 5.12: Adjusting cdf density scores for different values of φ.

we obtain the posteriori odds:

$$\mathrm{Odds}(A:\neg A|E) = \frac{P(A|E)}{P(\neg A|E)} = \underbrace{\frac{P(E|A)}{P(E|\neg A)}}_{\text{Bayes Factor}} \cdot \underbrace{\frac{P(A)}{P(\neg A)}}_{\text{Prior Odds}} . \tag{5.12}$$

Assuming an outlier rate of φ, we have the prior probabilities of $P(O) = \varphi$ and $P(\neg O) = P(I) = 1 - \varphi$, and thus prior odds of $\varphi/(1-\varphi)$. The Bayes factor is a likelihood ratio; in order to compute it we need to specify the probability density for the given evidence (i.e. the observed outlier score) for both the outlier and the inlier distribution. Since we estimated a distribution of the inlier scores just before, we can use the pdf of this distribution for $P(E|I) = \mathrm{pdf}(E)$.

Choosing a distribution model for the outliers is harder. The trivial choice of a uniform distribution (corresponding to the idea that outliers could be anywhere in the data) will also detect


(a) Densities for two outlier types: low and high values

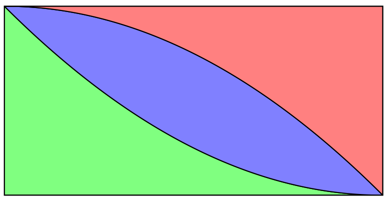

(b) Densities for three outlier types: low, central and high values

Figure 5.13: Beta distribution probability densities.

unusually low cdf values, i.e. unusually dense objects when using a density based score. Therefore, we need to choose a different model. Instead of treating all outliers the same, we can also assume there are two/three kinds of outliers: unusually low values, unusually high values and optionally unusual observations within the central area. Assuming these two/three classes exist in a uniform $\mathcal{U}[0;1]$ distribution, we can model this using a beta distribution: for the two-class situation, the smallest of two observations is $\mathrm{Beta}(1,2)$ distributed and the largest is $\mathrm{Beta}(2,1)$ distributed. The corresponding pdf functions are simply $2(1-x)$ and $2x$, and visualized in Figure 5.13a. Assuming three-classes, the distributions are $\mathrm{Beta}(1,3)$, $\mathrm{Beta}(2,2)$ and $\mathrm{Beta}(3,1)$, with pdf functions $3(1-x)^2$, $6x(1-x)$ and $3x^2$ as seen in Figure 5.13b. In the following, we will be assuming we are only interested in the high outliers of the three type model, which aligns with the notion of unusually sparse objects.

This yields a Bayes factor of

$$\text{Bayes-Factor}(O:I|p,\varphi) := \frac{3\,\mathrm{cdf}(x)^2}{\mathrm{pdf}(x)} \tag{5.13}$$

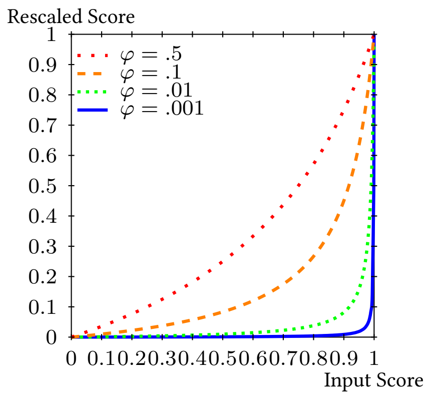
together with the prior odds this gives the posteriori odds

$$\mathrm{Odds}(O:I|p,\varphi) = \frac{3\,\mathrm{cdf}(x)^2}{\mathrm{pdf}(x)} \cdot \frac{\varphi}{1-\varphi} \tag{5.14}$$

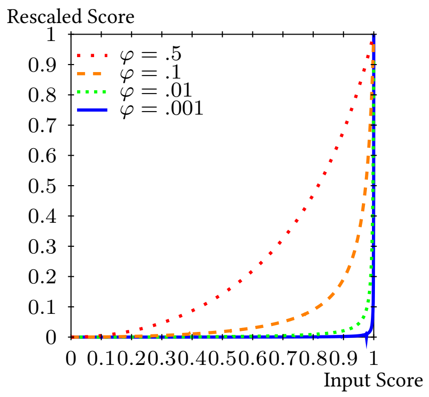
which can be transformed to a probability using Equation 5.11

$$\begin{aligned} P(O|x,\varphi) &= \left(\frac{3\varphi\,\mathrm{cdf}(x)^2}{(1-\varphi)\,\mathrm{pdf}(x)}\right) \Big/ \left(1 + \frac{3\varphi\,\mathrm{cdf}(x)^2}{(1-\varphi)\,\mathrm{pdf}(x)}\right) \\ &= \frac{3\varphi\,\mathrm{cdf}(x)^2}{(1-\varphi)\,\mathrm{pdf}(x) + 3\varphi\,\mathrm{cdf}(x)^2} \end{aligned} \tag{5.15}$$

In Figure 5.14b we visualize this rescaling function. Due to the lack of a ground truth pdf function for this visualization, we use the average low and medium density: $\mathrm{pdf}(x) = \frac{1}{2}[3(1-x)^2 + 6x(1-x)] = \frac{3}{2}(1-x^2)$ for visualization purposes only. In contrast to this, for Figure 5.12e and Figure 5.12f we have a true pdf. In these Figures we can see that for small values of φ there is little difference between the three rescalings.



(a) Rescaling with Equation 5.9, as proposed in COP [Kri+12].



(b) Rescaling with Equation 5.15, assuming $\mathrm{pdf}(x) = \frac{3}{2}(1 - x^2)$.

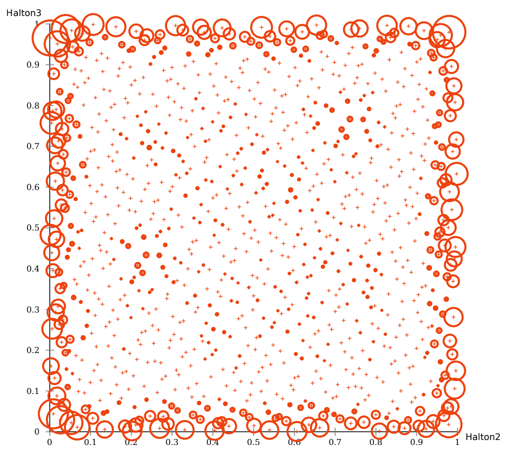
Figure 5.14: Visualization of rescaling functions.

An interesting test case arises from using the Halton low-discrepancy uniform data, which does not contain true outliers by definition. The results of running LOF with $k = 20$ and using COP scaling (Equation 5.9) with $\varphi = 1\%$ can be seen in Figure 5.15a. Using the Bayesian scaling function Equation 5.15 with the same $\varphi = 1\%$ yields even less outliers in Figure 5.15b. Compared to the naïve linear scaled outlier scores in Figure 5.6a the result has substantially improved. Intuitively, on a data set with 1000 instances, $\varphi = 1\%$ should yield 10 false positives. While both approaches produce much more than 10 non-zero scores in above Figures, note that neither COP nor Bayesian scaling score any object higher than 0.5.

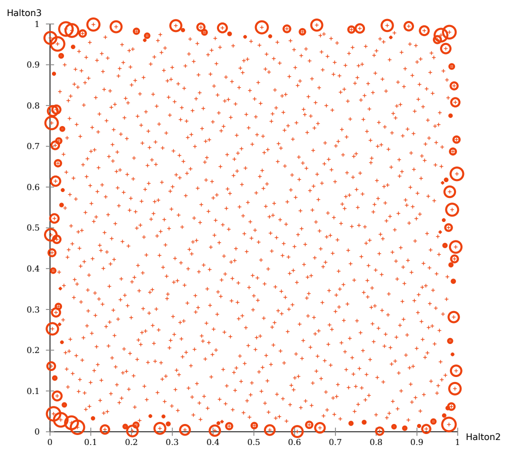
For numerical reasons, it would be beneficial to compute the inlier probability instead of the outlier probability: we want to have maximal numerical precision for the outliers, and do not need high precision for inliers. Floating point arithmetic is more precise close to 0 (where the theoretical precision is on the order of 10^{-324}; the precision at a more realistic 0.01 is on the order of 10^{-16}) than it is near 1 (where double precision is roughly 10^{-16}). An alternate approach to improve numerical precision is to work in logarithmic space, which also improves numerical accuracy. It can be beneficial to keep the Bayes-Factor value instead of transforming it to a probability using Equation 5.15 immediately: the Bayes Factor uses the numerical range of $[0; \infty]$, and the logarithm of it even the full numerical range $[-\infty; \infty]$.

However, as our current codebase does not yet include logarithmic versions for all cdf and pdf versions, we perform all our experiments with literal implementations using 1.0 as outlier and 0.0 as inlier score. Future practitioners implementing these approaches may however be advised to evaluate these implementation details in order to improve numerical precision.

In the remainder of this chapter, we will primarily use the cdf distributed scores. This value has the strongest semantics (in particular, it does not have the parameter φ) and needs the least assumptions.



(a) LOF $k = 20$ outlier scores, scaled using Equation 5.9, $\varphi = 1\%$



(b) LOF $k = 20$ outlier scores, scaled using Equation 5.15, $\varphi = 1\%$

Figure 5.15: Statistical scaling of outlier scores on Halton uniform data.

5.4 Evaluation of Outlier Scores

In the previous section, we focused on transforming scores, largely motivated by an intuition of what is an usable outlier score. However, we would also like to quantitatively measure whether a transformed outlier score is more useful than the raw data. In this section we will discuss novel evaluation methods. In Chapter 7 we will see that this advanced evaluation also allows more advanced combination of methods in ensembles.

The evaluation methodology discussed next is an extension of the following publications:

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Interpreting and Unifying Outlier Scores". In: *Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.* 2011, pp. 13–24

E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. "On Evaluation of Outlier Rankings and Outlier Scores". In: *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA.* 2012, pp. 1047–1058

5.4.1 Evaluation by Precision@k and ROC Curves

Traditional evaluation methods for outlier detection algorithms have one major limitation: all the commonly used methods ignore the actual outlier scores. These methods are largely obvious adaptations of existing evaluation methods from other domains such as information retrieval, classification and signal processing. This falls short of the aim of outlier detection for the following reasons:

- Some objects are obvious outliers, others are outlier candidates. The numeric outlier score should reflect this property of the objects.
- There might be no outliers at all, and an outlier detection method should be able to indicate this by assigning low scores only.
- The “ground truth” labels usually indicate some kind of “interestingness”, not necessarily “outlierness”. There might also be additional outliers within the majority classes.
- There might be multiple kinds of outliers in the same data set (not all of which are considered to be “interesting”).
- The “ground truth” labels are often incomplete, but only include known outliers.

Precision@k and Average Precision: The simplest evaluation is using the *precision* when looking at the top k objects only, where k is usually set to the number of known outliers. Formally, this number can be defined for a result R as

$$\text{precision@}k(R) := \left|\text{Top-}k(R) \cap \text{positive}\right| / k \qquad . \tag{5.16}$$

This is an easy to understand, but also very crude measure. First of all, k is not very large, since outliers are rare observations. Furthermore, if the elements at k and $k+1$ have the same score, the result is not uniquely determined. Last but not least, the order *within* the top k elements is ignored; in particular it does not make a difference whether errors occur at the beginning or at the end of the top k elements.

The result for this last situation can be somewhat improved by instead of taking a single k, one uses the average over all $k := 1 \dots r$ (i.e. measuring the area under the $\text{precision@}k$ curve, usually one chooses $r = |\text{positive}|$), known as the average precision:

$$\text{average-precision}_r(R) := \text{mean}_{k':=1}^{r} \text{precision@}k'(R) \tag{5.17}$$

When additionally averaging over many runs, this measure is also known as mean average precision (MAP). This measure is popular in information retrieval, where for example the first page of ten results of a search engine over a large set of queries is analyzed. Note that this measure does not have dedicated handling for imbalanced data sets, which is by definition the situation we face in outlier detection. Choosing k (or r) small enough is just a workaround.

Receiver Operating Characteristic (ROC): This method received its name from the use in signal transmission to analyze the quality of a transmission channel by plotting successfully vs. incorrectly transmitted signals such as the reflection of radar signals by airplanes. It has since been successfully adopted for machine learning [Spa89; PF97]. The ROC curve is obtained by connecting the coordinates of $(\text{false positive rate}, \text{true positive rate})$ pairs obtained when adding elements from the outlier ranking one after another. The curve begins at $(0,0)$ (when no object was processed yet) and ends at $(1,1)$ (when all positive and negative instances have been processed). Figure 5.16 shows an example ROC curve. The optimal ROC curve begins at $(0,0)$ and goes straight up to $(0,1)$ before continuing to $(1,1)$ – first all positive instances

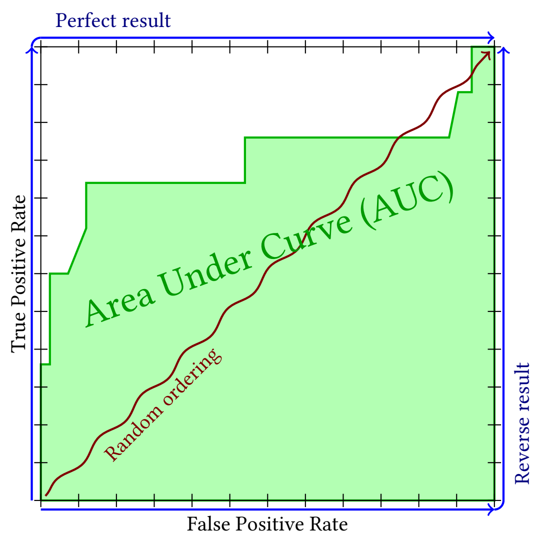


Figure 5.16: ROC curve example.

then all negative instances. The worst ROC curve however is not the one going through $(1, 0)$: this curve indicates a very good ranking, but exactly backwards. Instead, the worst curve remains close to the diagonal, because the true positive and false positive rates are expected to simultaneously increase on a random ordering.

Finally, in order to condense the ROC curve into a single score, the area under the curve (AUC) is often computed. The perfect result will get a score of 1.0, while a random result will get a score of approximately 0.5. The only difficult part in interpreting this score is that results significantly below 0.5 indicate that the method has been used incorrectly, since it returns the objects in reversed order. Technically, the AUC is an average: it is the arithmetic average true positive rate, averaged over all inliers $i \in I$, which can be rewritten further:

$$\mathrm{ROCAUC} := \mathrm{mean}_{i\in I} \frac{|\#\text{outliers before } i|}{\mathrm{rank}(i)} = \mathrm{mean}_{i\in I, o\in O} \begin{cases} 1 & \text{iff } \mathrm{rank}(o) < \mathrm{rank}(i) \\ 0 & \text{otherwise} \end{cases}$$

This last formula is an interesting probabilistic interpretation, which was shown in [HM82]: Given a randomly chosen inlier i and a randomly chosen outlier o from the data set, the AUC value is the likelihood of the objects being correctly ordered (i.e. o before i) in the ranking. Furthermore, the same quantity is also estimated by the Wilcoxon statistic / Mann-Whitney U statistic as shown in [Bam75]. With this interpretation it is easy to understand that 0.5 indicates a random ordering, while values smaller indicate that the method returns inliers first.

Similar to $\mathrm{precision@}k$ and $\mathrm{average\text{-}precision}_r$, the ROC curve does not use the actual outlier *scores*, but only the ordering imposed on the data by the scores. In order to get deterministic results, objects that are tied at the same score need to be processed at the same time when constructing the curve, which leads to steps at a sloped angle in the plot. However, the evaluation with ROC curves does not allow for scores or rankings on the reference set, but assumes a dichotomous problem (i.e. two classes, correct and incorrect).

Since ROC curves are based on the *rates* of true and false positives, this evaluation method is naturally well suited for handling a highly imbalanced problem such as outlier detection. Since the rate changes are relative to class sizes, an outlier will yield a larger step than a much more frequent inlier. The ROC curve sketched in Figure 5.16 exhibits this property: on the y axis, steps are 0.1 each, since the data set contains just 10 outliers. On the x axis, the step size is 0.01, corresponding to 100 inliers.

Smooth ROC Curves: Smooth ROC curves (smROC) [Kle+11] is an attempt to enhance ROC curves, taking the scores into account. However, the method shows severe anomalies when the midpoint is not at 0.5, due to a non-continuity in its inversion step as shown in [Sch+12]. This makes this method inapplicable in our problem setting until this issue with the method has been resolved.

Instead of trying to fix the problems of smROC algorithm, we do however propose to step back, and take a more abstract view of the problem on two lines of thought: on one hand, we need to revisit the objectives for our evaluation. Is it just the retrieval rate in the top k elements, as measured by $\text{precision@}k$, the ranking quality as measured by ROC, or do we maybe want to be able to measure arbitrary similarities of results, also across different results? On the other hand, can we really neglect the scores, and use only the ranking: in Section 5.3.2.6 we spent a lot of effort of rescaling outlier scores to a domain that supposedly is more useful to the end user, but we are so far lacking a way to measure these achievements. Therefore, in the next section, we will look at the evaluation on a more abstract and thus more flexible way, in order to obtain a quality measure that is both capable of comparing arbitrary results and evaluating scores. Furthermore, we will be able to base it on the most fundamental concept in the measurement of similarities: distance functions.

5.4.2 Calibration of Outlier Scores

Early attempts at "calibrating" outlier detection methods can be found for example in [GT06], which utilized calibration approaches (sigmoid functions and mixture modeling) to fit outlier scores provided by different detectors into probability values. The combination of probability estimates instead of mere ranks is demonstrated in [GT06] to slightly improve on the feature bagging methods of [LK05]. Notably, although their method should theoretically be applicable to combine different outlier detection algorithms, their experiments demonstrate the combination of different values for k of the kNN-distance as an outlier score only. Note that the sigmoid learning and mixture model fitting approaches proposed for calibration in [GT06] are based on the generalized EM algorithm and are rather unstable – there is no safeguard in place to prevent degeneration of results. In addition, they favor extreme values (0 and 1, largely removing any intermediate values from the data set), which is not favorable for combination, but cause the actual ensemble to degenerate to boolean combinations and counting.

An important difference between the probability estimation of outliers and of classes is the inherently imbalanced nature of the outlier detection problem. Since the data are largely dominated by the class of inliers and only a minimal number of data objects are truly outliers, assessing the root mean squared error in reliability as deviation of the probability estimates vs. the observed frequency (as e.g. in [MW77], for different classical evaluation measures see [ME67]) is not directly applicable to the scenario of outlier probability estimates. Any outlier detection procedure always estimating a zero (or a very small) outlier probability would already be almost perfectly calibrated. "Optimal" calibration can be achieved by merely estimating the proportion of outliers in a data set instead of assessing the outlierness of single data objects. In the supervised context of classification, much effort has been spent on methods of sanitizing imbalanced class prior probabilities for training as well as for evaluation of classifiers [BPM04; JJ04; RK04; Wei04]. Closely related to the problem of imbalanced classes are cost sensitive learning [WMZ07; Cha+08] and boosting of classifiers [JKA01]. Applying a cost-model to a prediction task means that errors are differently penalized for different classes during the learning procedure. Different costs for different types of error are a quite realistic scenario. Considering the example of weather forecasts, the costs for taking an umbrella when it will not rain and for not taking an umbrella when it will rain are probably different. This can be set-up as cost matrix:

	S_1	S_2
O_1	u_1	c_1
O_2	c_2	u_2

where c_1 is the cost of choosing option O_1 when state S_2 occurs and c_2 is the cost of choosing option O_2 when state S_1 occurs. For example, states S_1 and S_2 are "it rains" and "it does not rain", respectively, whereas options O_1 and O_2 are "take umbrella" and "take no umbrella", respectively. Hence, what truly counts is not the optimally calibrated probability estimation but the minimized costs of a decision (in the decision-theoretic sense of [vM53], see also [Mur66]), where the decision, of course, is based on the estimated probability. Instead of costs, the expected utility could also be modeled (setting u_1 and u_2 in the cost matrix to values other than 0), or both, utility and costs. The same is usually the case in applications of outlier detection. Sometimes, the cost of false-alarms is higher, sometimes it may be more important not to miss any potential outlier. Such requirements relate to differently weighting precision and recall. While in supervised scenarios classifiers can be optimized w.r.t. a certain cost model [Dom99; ZE01], in the unsupervised scenario of outlier detection, the assumed cost-model cannot be used to fit or train the algorithm but only to evaluate its results. It should be noted, though, that while calibration and purely calibration related scores in itself are not a sufficient evaluation measure, a useful cost-model-based evaluation of decisions will also encourage calibration [Win96]. In the context of imbalanced classes, it is customary to either sample the small class up, to sample the large class down, or to alter the relative costs according to the class sizes. In [JS02], the latter has been shown to be generally more effective than the alternatives. Hence we adopt this procedure here, though, since we tackle outlier detection as an unsupervised task, not for adapting a training procedure to differently weighted misclassification costs but merely to evaluate the impact of a probabilistic scaling and regularization of outlier scores. Aside from a quantitative improvement, the major motivation for such a probabilistic scaling is to revert the more

Table 5.2: Summarized rank comparison measures

Simplified Spearman's ρ			
Normalization:	rank	Weighting:	none
Vector space:	score vectors	Measure:	normalized squared Euclidean
Full Spearman's ρ			
Normalization:	rank	Weighting:	none
Vector space:	score vectors	Measure:	Pearson correlation
Kendall's τ			
Normalization:	none	Weighting:	none
Vector space:	order relation	Measure:	Pearson correlation
ROC AUC			
Normalization:	none	Weighting:	1 iff in different classes
Vector space:	order relation	Measure:	Regression coefficient
Unified Outlier [Kri+11]			
Normalization:	various normalizations proposed	Weighting:	by class sizes
Vector space:	score vectors	Measure:	weighted Manhattan distance
ClasSi [IWS11]			
Normalization:	rank	Weighting:	by class sizes
Vector space:	order relation	Measure:	normalized Pearson

and more deteriorated interpretability of modern outlier detection methods into a statistically interpretable context.

In summary, we want to be able to transfer the experiences with (i) probability estimates, (ii) imbalanced classification problems, and (iii) cost-sensitive learning reported in the context of supervised learning to the context of unsupervised outlier detection. However, neither the evaluation using Precision@k nor the evaluation using ROC curves is helpful for this aim, so we need a more powerful evaluation method that takes scores into account, and that allows us to assess the impact of our transformation methods for outlier scores w.r.t. the reduction of error-costs, taking into account the different cardinality of the class of inliers I and the class of outliers O.

5.4.3 Evaluation by Distance Measures in the Vector Space of Scores

The motivation in [Kri+11] was based on the idea that instead of assessing binary decisions we can multiply the assigned probability estimates with the corresponding costs for erring. Since the costs for the correct decisions are always 0, only errors account for the reported values. The corresponding accuracy values would be symmetric since the assigned probability estimates would just be the complementary probabilities of the ones accounting for error costs. Formally, the reported costs are for each dichotomous problem consisting of classes I and O:

$$\frac{1}{2}\sum_{x\in I} P(O|x)\cdot\frac{1}{|I|}+\frac{1}{2}\sum_{x\in O} P(I|x)\cdot\frac{1}{|O|},$$

based on probability estimates $P(C|x)$ for an object x to belong to class C as provided by the (unified) outlier score. Note that we could also easily use quadratic costs in this formula instead by using the squared probability estimates.

We can however also rewrite this formula using $P(I|x) = 1 - P(O|x)$ and $P(\_) > 0$ as follows:

$$\begin{aligned} &\sum_{x\in I} P(O|x)\cdot\frac{1}{2|I|} + \sum_{x\in O} P(I|x)\cdot\frac{1}{2|O|} \\ =&\sum_x \begin{cases} P(O|x)\cdot\frac{1}{2|I|} & \text{if } x\in I \\ 1-P(O|x)\cdot\frac{1}{2|O|} & \text{if } x\in O \end{cases} = \sum_x \omega_x \left|\mathcal{I}_O(x) - P(O|x)\right| \end{aligned}$$

where $\mathcal{I}_O(x) = 1$ iff $x \in O$ and ω_x is a weight vector assigning the appropriate costs for erring $\frac{1}{2|O|}$ and $\frac{1}{2|I|}$ to objects of O and I. The formula now, however, is nothing but a weighted Manhattan distance function of the score vector $P(O|x)$ to the target vector $\mathcal{I}_O$ indicating the desired true outliers. If we had used the squared costs, we would have obtained the weighted Euclidean distance.

The vector space, in which we compute the distance function, has n dimensions, each corresponding to one object of the data set. An alternate vector space outlined in [Sch+12] is the order relation (a vector space of size $\mathcal{O}(n^2)$), where each dimension corresponds to a pair of objects, indicating which should come before the other. ROC AUC and ClasSi [IWS11] can be interpreted as using this order relation vector space, where the dimensions are weighted depending on the classes of the objects represented by the dimension (usually 0, when the objects are in the same class). Table 5.2 summarizes how existing measures can be represented as a data normalization, a transformation into such a vector space and then the application of a distance function. The approach we proposed in [SZK12] for evaluating outlier scores is a straightforward instantiation of this framework. For normalization, any of the methods discussed in Section 5.3 can be used (and in the end, we want to be able to evaluate empirically which of these works best for particular algorithms). Since we want to evaluate the scores, the score vector space is the most appropriate space, and due to the imbalanced nature of the problem we need to weight the different classes appropriately. Finally, there is no clear indication of which distance function to prefer. We will discuss a number of choices next.

In [Kri+11], we proposed the linear cost function above, which is a weighted Manhattan distance. This is a straightforward cost estimation, but it treats small and large errors equally. In [Sch+12], we discussed alternate choices such as the weighted Euclidean distance, which computes the root mean square (RMS, M_2, see Section 2.1 for a discussion of different means) error. Obviously, any mean based norm can be used here in a weighted form – see Section 2.2 for the strong relationship of L_p-norms and generalized means. Another choice we proposed in [Sch+12] is the use of a weighted Pearson correlation. Formulas for the computation of weighted Pearson and weighted covariance (along with a numerically stable online computation method) can be found in Appendix 1. Pearson correlation is particularly useful when the scores are not normalized well, because of the built-in standardization. When X and Y are standardized to unit variance (i.e. $\sigma_X = \sigma_Y = 1$), then Pearson correlation trivially reduces to covariance. When X and Y are furthermore centered (i.e. $\mu_X = \mu_Y = 0$) then the resulting distance function becomes a variant of squared Euclidean distance (see Appendix 1).

Putting all together, the proposed evaluation method for a normalized outlier score S can be formalized as the seemingly trivial formula:

$$\mathrm{err}(S) := \mathrm{dist}_\Omega(S, T), \tag{5.18}$$

where S is the score vector, Ω the weight vector, T the target vector (indicating which objects should be considered outliers, e.g. $T = \mathcal{I}_O$, but we will discuss other choices for the target vector below) and dist_Ω the chosen weighted distance function to measure the severity of the error. Note that the target vector can also be modified to allow for "maybe" outliers and different outlierness degrees. Similarly, the weight vector can be modified to emphasize certain objects (e.g. in an active learning context) beyond handling the class imbalancedness.

As with many of the choices before (such as the function to use for density estimation and the score normalization method) there is no general winner. None of the solutions will work best in every situation. In many situations, the different measures will produce the same ranking. The rule of thumb we observed is the following:

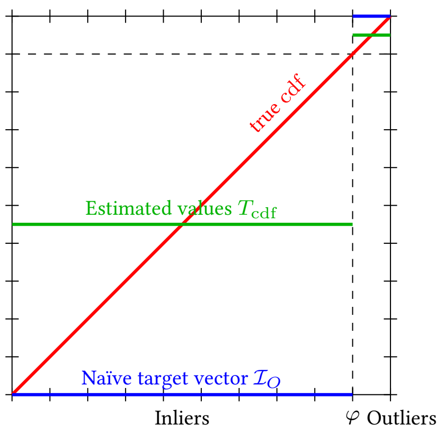
- If the scores are not well normalized, the built-in normalization of Pearson correlation often produces more comparable results. The other measures may be measuring too much the quality of the normalization.
- When the scores have been z-standardized, Pearson correlation distance actually reduces to using squared Euclidean distance (see Appendix 1).
- If the score is to be used for a binary decision, the linear (L_1, Manhattan) distance should be most appropriate, resembling error probabilities.
- If larger errors have a much larger effect, quadratic (L_2, Euclidean) distance may be more appropriate (by using squared errors).

This vector-space-based evaluation method offers additional use cases beyond evaluation. For example, the similarity of outlier detection results can be computed by using a second outlier detection result instead of the "truth" as target vector:

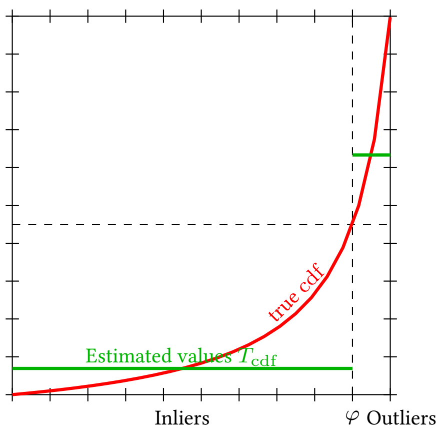
$$\mathrm{sim}(S_1, S_2) := \mathrm{dist}_\Omega(S_1, S_2), \tag{5.19}$$

This similarity will be useful for building ensembles (see Chapter 7), since it can serve as a measure for diversity, which was shown to be an important factor for the quality of unsupervised ensembles. It is, however, only an initial version for unsupervised quantification of diversity (see [Bro+05] for an overview of diversity in regression and supervised classification), and it remains to be seen whether we can also distinguish between "good" and "bad" diversity [BK10].

When we initially proposed this method of evaluation, we used the binary target vector $T = \mathcal{I}_O$, which was a reasonable first attempt. However, in the previous chapter we discussed that a good outlier score is in fact not binary, but uniformly distributed. This however implies that



(a) Naïve and expectation target vector for cdf scores.



(b) Expectation target vector for cdf scores with alternate scaling (for $\varphi = .1$).

Figure 5.17: Naïve target vector and expectation target vector for $\varphi = 0.1$.

no such outlier score can ever score well on this measure. To explain this, consider an ideal experiment: data generated from a Gaussian distribution, and the outermost $\varphi \cdot N$ objects are labeled as outliers. Then comparing the binary indicator vector to the ground truth (uniformly distributed) score yields:

$$\begin{aligned}
\mathrm{dist}_\Omega(\mathcal{I}_O, \mathrm{cdf}) &= \sum_{x\in I} x \cdot \frac{1}{2|I|} + \sum_{x\in O} (1-x)\cdot\frac{1}{2|O|} \\
\lim_{|I|\to\infty} &\to \frac{1}{2}\int_0^{1-\varphi} x \,\mathrm{d}x + \frac{1}{2}\int_{1-\varphi}^{1} 1 - x\,\mathrm{d}x \\
&= \frac{1}{4}((1-\varphi)^2 - 0) + \frac{1}{2}\left(1-(1-\varphi) - \frac{1}{2}(1-(1-\varphi)^2)\right) \\
&= \frac{1}{4}\left[(1-\varphi)^2 + 2\varphi - 1 + (1-\varphi)^2\right] = \frac{1}{4}\left[2(1-\varphi)^2 + 2\varphi - 1\right] \\
&= \frac{1}{4}\left[2 - 4\varphi + 2\varphi^2 + 2\varphi - 1\right] = \frac{1}{4}\left[1 - 2\varphi(1-\varphi)\right] \longrightarrow_{\varphi\to 0} \frac{1}{4}
\end{aligned}$$

This means that given a binary ground truth, even the true cdf score will score only slightly better than $1/4$. The simple reason is that the average score of an inlier even on the true cdf score is not 0, but just below 0.5. More precisely, the average value of an inlier is $\frac{1}{2} - \frac{\varphi}{2}$, whereas the average score of an outlier is $1 - \frac{\varphi}{2}$. Therefore, instead of using the naïve binary indicator vector above, we can instead construct a target vector using the expected scores using:

$$T_{\mathrm{cdf}} : t_i = \begin{cases} \frac{1}{2} - \frac{\varphi}{2} & \text{iff } x_i \in I \\ 1 - \frac{\varphi}{2} & \text{iff } x_i \in O \end{cases} \longrightarrow_{\varphi\to 0} \begin{cases} \frac{1}{2} & \text{iff } x_i \in I \\ 1 & \text{iff } x_i \in O \end{cases}$$

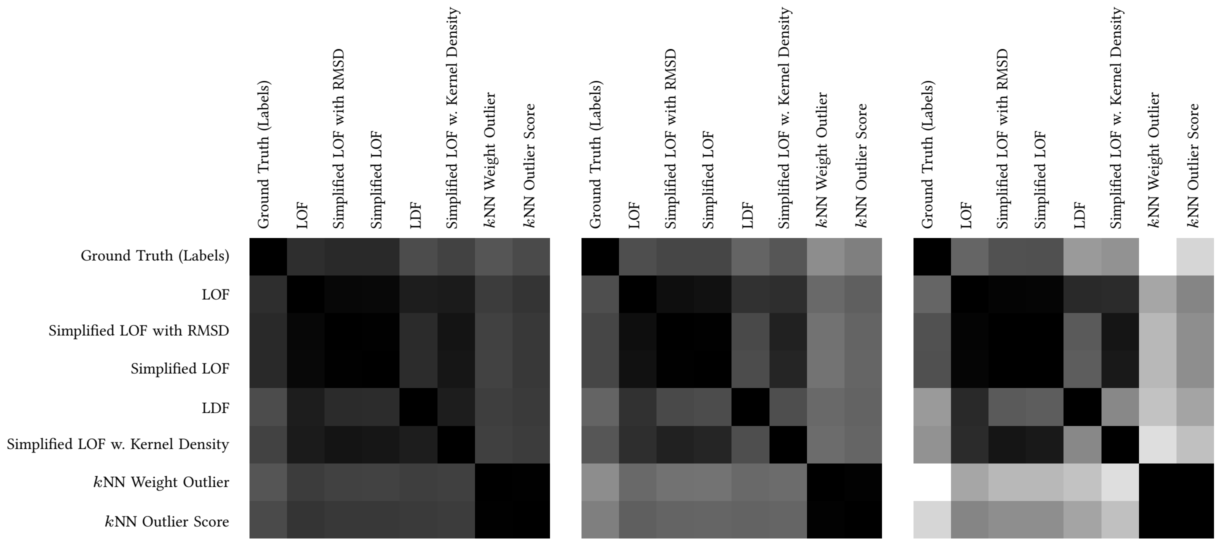
Performing the same computation as before, we get

$$\begin{aligned}
\mathrm{dist}_\Omega(T_{\mathrm{cdf}},\mathrm{cdf}) &= \frac{1}{2}\frac{1}{|I|}\sum_i \left|\frac{1}{2}-\frac{\varphi}{2}-x\right| + \frac{1}{2}\frac{1}{|O|}\sum_i \left|1-\frac{\varphi}{2}-x\right| \\
\lim_{|I|\to\infty} &\to \frac{1}{2}\int_0^{1-\varphi}\left|\frac{1}{2}-\frac{\varphi}{2}-x\right|\,\mathrm{d}x + \frac{1}{2}\int_{1-\varphi}^{1}\left|1-\frac{\varphi}{2}-x\right|\,\mathrm{d}x \\
&= \int_0^{\frac{1-\varphi}{2}} \frac{1-\varphi}{2} - x\,\mathrm{d}x + \int_{1-\frac{\varphi}{2}}^{1} x - \left(1-\frac{\varphi}{2}\right)\,\mathrm{d}x \\
&= \left(\frac{1-\varphi}{2}\right)^2 - \frac{1}{2}\left(\frac{1-\varphi}{2}\right)^2 + \frac{1}{2}\left(1-\left(1-\frac{\varphi}{2}\right)^2\right) - \left(1-\frac{\varphi}{2}\right)\frac{\varphi}{2} \\
&= \frac{1}{2}\left[\left(\frac{1-\varphi}{2}\right)^2 + \varphi - \frac{\varphi^2}{4} - \varphi + \frac{\varphi^2}{2}\right] = \frac{1}{8}\left[(1-\varphi)^2+\varphi^2\right] \\
&= \frac{1}{8}\left[1-2\varphi+2\varphi^2\right] = \frac{1}{8}\left[1-2\varphi\,(1-\varphi)\right] \longrightarrow_{\varphi\to 0} \frac{1}{8}
\end{aligned}$$

This result is not surprising: we are trying to find the best approximation of the linear function using two constant approximations. The difference between the two target vectors is visualized in Figure 5.17a for $\varphi = 0.1$: the naïve target vector $\mathcal{I}_O$ is binary, either 0 or 1, while the improved target vector T_{cdf} is the piecewise average for inliers and outliers. The cdf of the true distribution would be a diagonal line. There is no rescaling of a *binary* ground truth that will be an exact match for the diagonal line; and it is known that the best constant approximation of a line is the mean value; which is why we chose the means of the two segments each. Note that the cost functions weights inliers with 50% of the distance and outliers with 50%, so an error in the inliers is not the same as an error in the outlier area. Figure 5.17b shows the same vector when the scores have been rescaled as suggested in Section 5.3.2.6 (rescaled with the same $\varphi = 0.1$). It can be seen that when such a rescaling is used, the median value for inliers decreases, while the median value for outlier decreases towards $.5$. Applying the rescaling to the target vector yields the median value, not the mean value of each segment. This median is the optimal choice for linear error costs (i.e. Manhattan distance), the mean would be the optimum for quadratic deviations (i.e. using Euclidean distance).

Reducing a cost bias from approximately $1/4$ to $1/8$ itself is not much of a benefit. However, without this correction, scores that do not exploit the full value range will appear to have a better performance, when in fact they are not substantially better, but are "overfitted" for a binary target vector. Choosing the target vector T_{cdf} encourages methods to return intermediate values, and more closely resembles the idealized target vector, which has a uniform distribution.

Figure 5.18 visualizes the similarity of outlier detection results on the ALOI data set, as measured with Manhattan distance, Euclidean distance and Pearson correlation. The algorithm scores have been statistically scaled (using the scaling with the lowest K-S-score each, as per Table 5.1); then the interpretability has been increased according to the rate of labeled outliers using COP rescaling (Equation 5.9) with φ set to the true outlier rate.

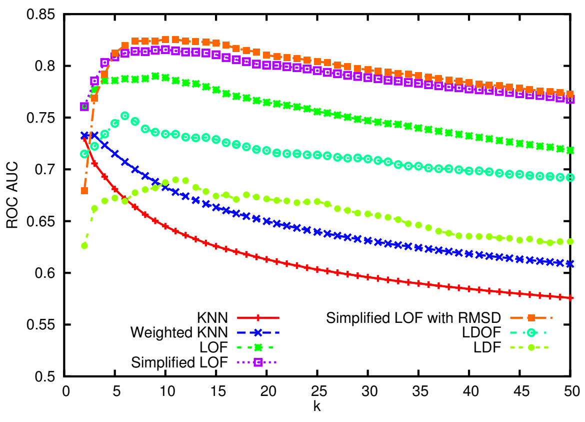


(a) Weighted Manhattan (b) Weighted Euclidean (c) Weighted Pearson

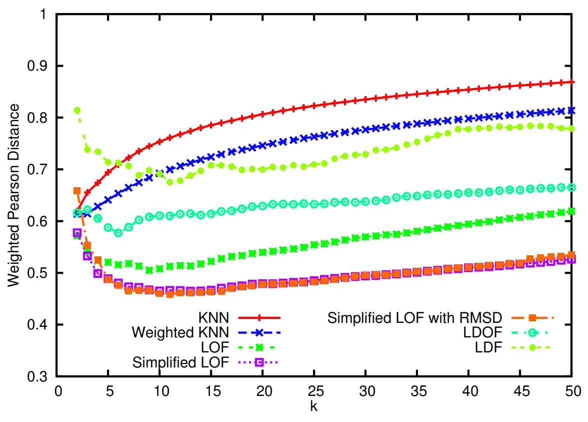
Figure 5.18: Outlier method similarities on ALOI data set.

Table 5.3: Outlier detection evaluation scores on ALOI data set (score and rank).

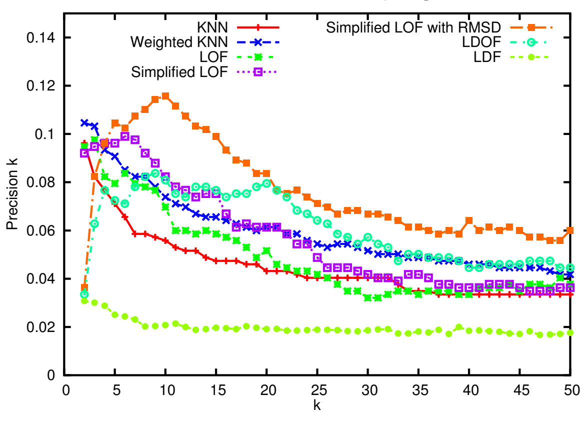
Method	ROC AUC		Precision@k		Average P@k		$L_{1,\Omega}$		$L_{2,\Omega}$		ρ_Ω	
LOF	.7789	3	.0600	4	.0535	5	.2331	3	.2954	3	.5180	3
Simplified-LOF with RMSD	.8070	2	.0669	2	.0622	4	.2195	2	.2820	2	.4711	2
Simplified-LOF	.8106	1	.0753	1	.0650	3	.2177	1	.2811	1	.4667	1
LDF	.7137	5	.0363	6	.0426	6	.2915	6	.3295	5	.6261	5
Simplified-LOF with Kernel Density	.7594	4	.0181	7	.0193	7	.2741	4	.3080	4	.6093	4
kNN Outlier	.6289	7	.0488	5	.0751	2	.3066	7	.3849	7	.7801	7
kNN Weight Outlier	.6635	6	.0656	3	.1035	1	.2890	5	.3680	6	.7237	6



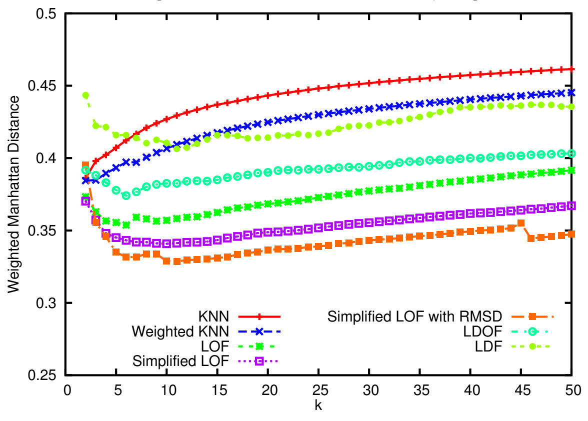
(a) ROC AUC with varying k



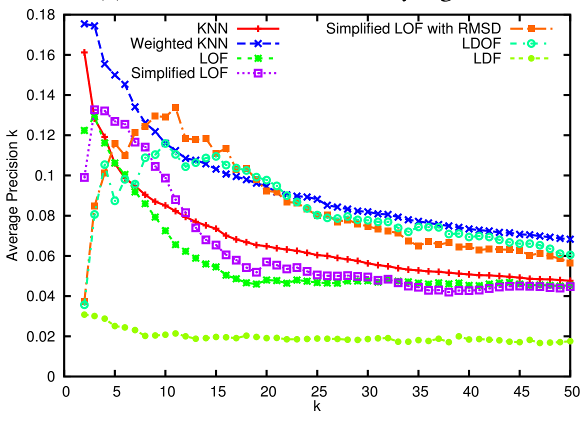
(b) Weighted Pearson with varying k



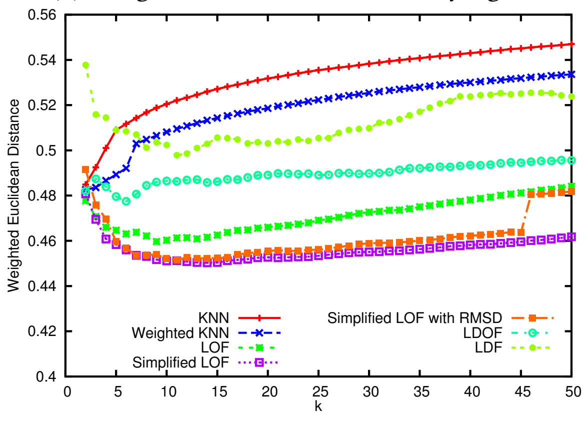
(c) Precision at o with varying k



(d) Weighted Manhattan with varying k



(e) Average precision @o with varying k



(f) Weighted Euclidean with varying k

Figure 5.19: Different evaluation measures still agree on the overall trends when varying the algorithm parameter k (neighborhood size).
Left column: high values are better – right column: low distances are better.

Figure 5.19 visualizes the performance of different algorithms when varying the neighborhood size parameter k. On the left, we first see the classic ROC AUC measure (Figure 5.19a), then precision@o (where o is the number of true outliers, Figure 5.19c) and the average precision for $1 \dots o$ (Figure 5.19e). The precision@o measure is less stable (having $o + 1$ discrete values only), and the average precision is dominated by the precision in the first ranks. Neither of these measures used the actual scores, but only the ranks and labels. On the right we see distance based measures. Note that for distances, low values are best. It can clearly be seen that the distance measures produce a similar result to the ROC AUC measure, just upside down. Weighted Pearson correlation (Figure 5.19b) is less sensitive to score rescaling, but Manhattan and Euclidean measures (Figure 5.19d and Figure 5.19f) are expected to provide a more meaningful result for use in ensemble methods (see Chapter 7) where the actual scores are used for combination. In these two we observe an instability in the curve of the Simplified LOF variant with RMSD (which is essentially the LoOP method). This artifact, however, is neither caused by the actual outlier detection method nor the evaluation method, but by the automatic score distribution fitting process switching from a log-normal score model to a generalized extreme value distribution model. By enforcing a log-normal model, this can trivially be resolved.

The differences between the different similarity functions are not substantial: Manhattan visually has the lowest contrast, and Pearson correlation the highest, but there is no clear indication to prefer one over the other. The linear and squared error of Manhattan respectively Euclidean give these a theoretical benefit, the built-in standardization of Pearson may on the other hand be useful when the scores are not well normalized, and visually offers the better contrast. Also not surprisingly, the weighted evaluation methods largely agree on the relative quality of the methods, as it can be seen in Table 5.3. Only the precision@k and average precision @k measures divert substantially from the ranking of the methods on this data set. $L_{1,\Omega}$ switches position 5 and 6 but with a score difference of just 0.0025.

However, a key benefit of the new evaluation measures – as opposed to ROC AUC, and the precision based measures – is that we can also compute the similarity of two different results, as previously seen in Figure 5.18, which not only computed the score, but also the pairwise similarities. As seen there, LOF and Simplified-LOF are highly similar, and so are the results of kNN-Outlier and kNN-Weight. LDF is most similar to LOF: this is not surprising, as it uses a capped kernel similar to LOF, and not an unbounded kernel, as the Simplified LOF variations. Figure 5.20 compares the similarities of methods across $k = 2 \dots 20$. In this plot, we can both observe the high similarity between methods such as LOF and Simplified-LOF, and the two kNN methods. Surprisingly, LDOF is more similar to LOF in practice than the kernel density based variation LDF.

We can also analyze the effect of different distance functions on the results. In Figure 5.21 we see the similarity of LOF results with $k = 11$ and different distance functions. We start with the Minkowski L_p family of norms. The results of these are highly similar, which comes at little surprise (except for minimum distance, which did not work at all due to the sparsity of the vectors – most vectors share at least one attribute that is 0 in both). For this particular data set (with each vector having a sum of 1), some of the variants like histogram intersection

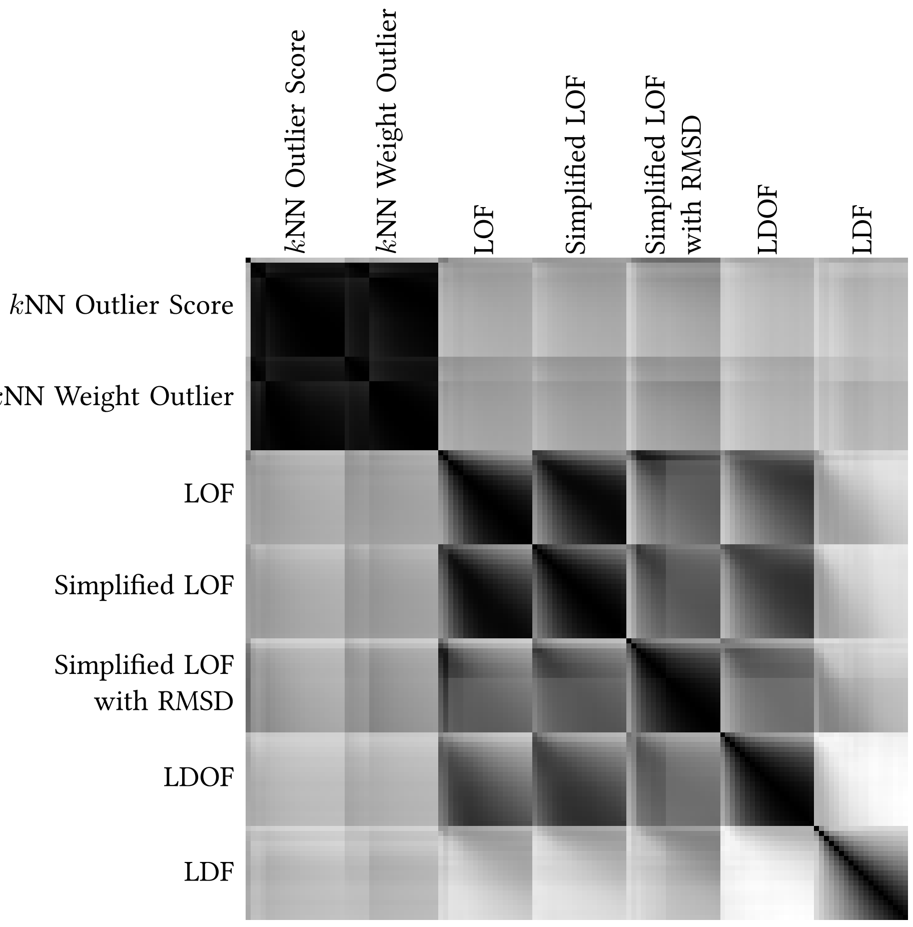


Figure 5.20: Similarity of different methods for different values of k.

distance [SB91], Bray-Curtis distance [BC57] (closely related to the Sørensen–Dice coefficient [Sør48; Dic45] and Hellinger distance [Hel09]), Kulczynski-1 and Lorentzian distance return next to identical results to Manhattan distance; this will however not hold for arbitrary data sets. Minkowski distance with $L_{0.8}$, sometimes called a "fractional" L_p-distance, is no longer a metric, but it can be used with LOF nevertheless and on this data set works slightly better. Cosine and covariance based metrics form another group that is highly similar; Pearson can be seen as angle of the z-standardized vectors and is thus closely related. The next group consists of different divergence measures, which have a good theoretical foundation on histogram data. Jeffrey divergence and Jensen-Shannon divergence only differ by a linear factor, and are asymptotically equal to the χ^2 measure [ES03]. Taking the square root of J-S-divergence will yield a metric [ES03]. Canberra distance (which scored best in this experiment) is closely related Manhattan distance, but less scale sensitive (see Section 2.2) and Clark distance is a similar modification of Euclidean distance. Details on all distance functions can be found in literature [DD06] and implementations are available in ELKI [Ach+13].

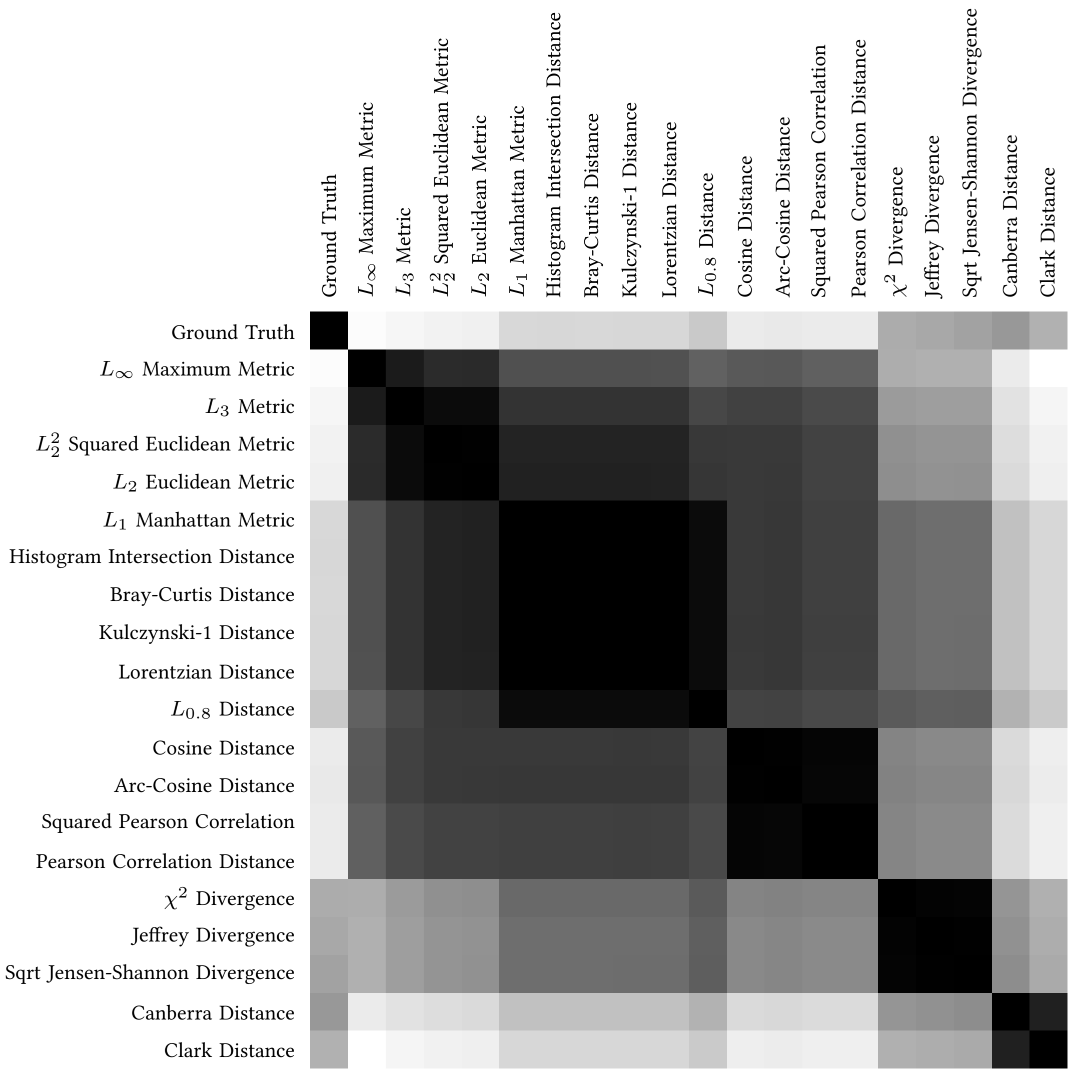


Figure 5.21: Similarity of LOF results based on different distance functions.

When desired, the similarity measure can also be split into two parts: one measuring the cost on inliers only, and the other part measuring the cost on outliers. This may be beneficial for some use cases, but we have not yet investigated this in detail.

This capability of comparing *different* results will be essential for Chapter 7 to improve outlier detection ensembles by removing redundant results and improving ensemble diversity.

6 Generalization and Modularization

“An idea is always a generalization, and generalization is a property of thinking. To generalize means to think. — *Georg Wilhelm Friedrich Hegel*”

“Complexity that works is built up out of modules that work perfectly, layered one over the other. — *Kevin Kelly* in [Flo95]”

“An abstraction is one thing that represents several real things equally well. — *Edsger W. Dijkstra* [Cra07]”

“Everything should be as simple as possible but not simpler. — *Albert Einstein* [Cra07]”

In the previous chapters, we have investigated a number of algorithms that are closely related, and that try to solve the essentially same problem – detecting outliers – with different, yet similar, algorithms.

The differences between algorithms such as LOF, Simplified-LOF and LoOP are hard to understand, and the recent developments of adding a score normalization further add to this. In the spirit of adding a normalization “module” to the existing algorithms, can we also represent them in a modular structure that allows them to share such modules? It turns out that this indeed works. We already used some of this modular structure in the previous chapters, for example when we investigated the relationship of LOF to kernel density estimation in Section 5.1.3. In this chapter we will introduce a formalism to describe an existing outlier algorithm as a series of modules, and discuss how this modular structure allows the algorithms to be easily adopted to different data domains.

The difficult challenge is finding the right level of abstraction – as simple as possible but not simpler – but still capable at representing the needs for outlier detection, and allowing it to be easily applied to new domains. And yet: When we formalized this abstraction, it appeared almost too simple and too general. To value the beauty of this result, we need to compare it to the uncontrolled growth of outlier detection algorithms that have been published the previous decade. It’s this abstraction that brings them together into variants of a general scheme, which allows new recombinations and easy adaption to new problems.

“Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better. The computing industry is not the only one that has discovered that sore truth: so has the academic world. — Edsger W. Dijkstra [Dij87]”

This chapter is based on the ideas of generalization published as:

> E. Schubert, A. Zimek, and H.-P. Kriegel. "Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection". In: *Data Mining and Knowledge Discovery* (2012). DOI: 10.1007/s10618-012-0300-z

6.1 Different Notions of Locality

In a rather general sense, the very nature of outlier detection requires the comparison of an object with a set of other objects w.r.t. some property (e.g. the k nearest neighbor distance or a density model). When comparing different outlier detection methods, we find different levels of restriction of the set to compare with. Furthermore, the property to be compared is usually also derived from the data set, taking into account, again, a set of other objects. Both sets, set A from which to derive the property for an object, and set B to compare with, need not be identical. We can name set A the *context set* for model building, and set B the *reference set* for model comparison.

This decomposition has been implemented gradually (and probably only to a certain extent intentionally) during the development of outlier detection methods as surveyed in Chapter 3. Consider the fundamental statistical methods. They are modeling the complete data set by a single distribution and judging an object basically by the probability of whether it could have been generated by the corresponding model. In this case, both the model building set and the reference set are the complete data set. The first approach to distance-based outlier detection (DB-outlier) already considers the local neighborhood by means of a range-query but compares the property thus derived with the complete data set. The same is true for k nearest neighbor-related outlier models: the model building set are the k nearest neighbors while the derived property is compared with the properties of the complete data set as a reference. Thus the meaning of "locality" introduced in LOF [Bre+00] relates to the locality of the reference set as well as the model building set. LOF uses the same neighborhood for both situations, but it could easily be abstracted to use different neighborhoods.

As opposed to the methods reflecting k nearest neighbor distances, in truly local methods the resulting outlier score is adaptive to fluctuations in the local density and, hence, intended to be comparable over a data set with varying densities. The central contribution of LOF and related methods is hence to enhance the comparability of outlier scores for a given data set.

Based on this fundamental distinction of (1) the context and (2) the method used for building the model, (3) the context to use as reference and (4) the method used for comparison of models of different objects. Finally (5) a normalization step for the values (i.e., the outlier scores). Note that we are only interested in unsupervised learning procedures, to highlight this restriction, we name this also "building" or "computing" a model instead of "learning", which usually indicates a supervised approach.

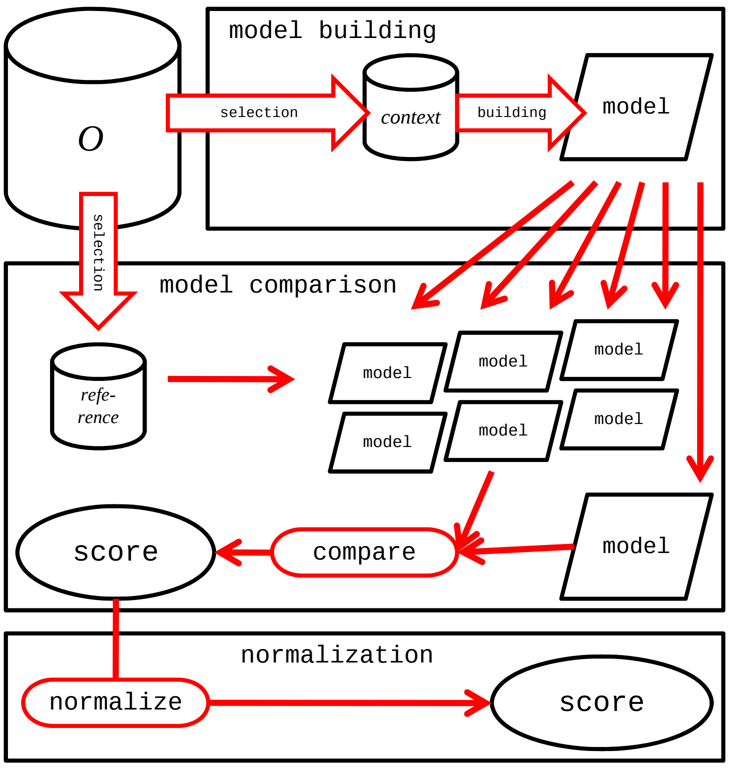


Figure 6.1: Workflow.

Algorithm 1: Typical Algorithmic Schema of Local Outlier Detection.

```
Data: database $O$
foreach $o \in O$ do                                          /* STEP 1:  model building  */
    select context($o$) ;                                     /* model building context (1) */
    model($o$) := buildModel(o, context($o$)) ;                /* model learning (2) */
end
foreach $o \in O$ do                                          /* STEP 2:  model comparison  */
    select reference($o$) ;                                   /* comparison context (3) */
    score($o$) := compare(model($o$), $\{$model($r$)$\}_{r \in \text{reference}(o)}$) ;   /* comparison method (4) */
end
if normalization required then
    foreach $o \in O$ do                                      /* STEP 3:  normalization  */
        normalizedScore($o$) := normalize(score($o$)) ;       /* normalization method (5) */
    end
end
```

This general algorithmic scheme, as visualized in Figure 6.1, can accommodate many different outlier detection methods. Essentially, as the figure suggests, we can group these five elements in three algorithmically separable steps. First, the model building step assigns a model (or some simple property as, e.g., a distance or density value) to each object $o \in O$ based on some set $\mathrm{context}(o) \subseteq O$. Second, the model comparison step compares the model of each object $o \in O$ with the models (built in the first step) of some set $\mathrm{reference}(o) \subseteq O$. This step can be figured as omitted by simple methods just performing a ranking of the "models" (in this case usually consisting of one-dimensional values like distances) retrieved in the first step – but the ranking procedure can also be seen as comparison step with a global reference set. Finally, the score retrieved in step 2 can be normalized (step 3). All these steps are performed on the given data set in order to identify (possible) outliers in an unsupervised manner.

In order to describe the exact methodological approach which some outlier detection method pursues, we therefore need to identify the procedures or definitions behind the method names used in the framework (Algorithm 1): *context*, *buildModel*, *reference*, *compare*, and *normalize*. Let us exemplify these methods considering the "local outlier factor" (LOF) [Bre+00], because it is a well-known outlier detection algorithm and many variants have been based on this basic approach. Also it uses most of these components. LOF uses the k nearest neighbors for both $\mathrm{context}(o)$ and $\mathrm{reference}(o)$ of an object o. The density model used by LOF, "local reachability density" was given in Equation 3.6, is based on this local context. The final score is then obtained by comparing this density model to the neighbor density models as per Equation 3.7. The only step not used in LOF is a global normalization. LoOP [Kri+09a], for example, is a LOF variation that uses this additional step.

Some variants define and use these building blocks in different ways. INFLO [Jin+06] uses more or less the same elements as LOF except for the reference set, which is defined as the intersection of the k nearest neighbors and the k reverse nearest neighbors (i.e., the set of those objects that list the query point among their k nearest neighbors). In the approach based on reference points [PZG06], the model is computed in a similar way as in LOF but based on the context of approximated k nearest neighbors (based on the neighbors of reference points) and compared over the complete set of reference points. Also, they are implementing a normalization. LOCI [Pap+03] is explicitly using two different radii for the context and the reference set. Approaches for outlier mining in high-dimensional data [e.g. AY01; Kri+09b; MSS10; Kri+12] usually assess neighborhoods based on distances in subspaces.

Aside from choosing different contexts and references for model building and model comparison, also the procedures of model building and model comparison can be quite diverse.

The model building step often is something as simple as using the object count in a particular radius (as a simple estimation of density). However, there are also much more complex models possible. A statistical baseline approach is an EM-like fitting of a Gaussian (inliers) and a uniform (outliers) distribution on the complete data set [described by TSK06]. Here, context and reference set are global but the model definition is statistically refined. Opposed to that, database-oriented approaches often try to simplify the model building for the sake of efficiency. ABOD [KSZ08], for example, computes the pairwise angles that local objects appear under. This model is then condensed to a single feature, the variance of the angle spectrum. LOF [Bre+00] assumes an OPTICS-like model [Ank+99] and estimates the density level at which the point became a cluster member. LOCI [Pap+03] compares object counts for different regions.

LoOP [Kri+09a] models local Gaussian distributions and estimates density using the variance of the resulting Gaussian distribution. LDOF [ZHJ09] computes the pairwise distances of local neighbors, and reduces this to the mean value as single-valued feature. The definitions of outliers of [RRS00] and [AP02] (regardless of their algorithmic merits) differ only marginally in the model definition (distance to the kth nearest neighbor vs. aggregated distances for the first k neighbors — this sum of distances provides a certain smoothing effect on the outlier scores but essentially measures the same property).

While single-dimensional values such as distance or density can easily be used for ranking directly, methods such as LOF [Bre+00] very successfully use more complex comparison methods. While the simple methods usually just take the maximum or minimum global value as the most prominent outlier, the advanced methods usually use a "local" context again as reference. LOF, for example, computes the quotient of the object's reachability density and the average reachability density of its neighbors. It thus no longer detects the globally least dense point as outlier, but those that are significantly less dense than their neighbors. This advanced model comparison marks some of the so-called "local" methods as truly local. Examples for local methods, aside from LOF, are INFLO [Jin+06], reference point-based [PZG06], LOCI [Pap+03], and LoOP [Kri+09a]. Let us note that LDOF [ZHJ09], though acclaimed to be local, is actually local only in the context but global in the model comparison reference set. This example of a possible misunderstanding of locality already demonstrates that a detailed scrutiny of the meaning of "local" in outlier detection could be rather useful in order to better understand the scope and contribution of different methods. These different notions – and complexities – of locality are the core interest in the present study.

Table 6.1 gives an overview on some well-known outlier detection methods represented in this framework. We list here only keywords or short terms reminding on the basic ideas of the listed approaches which mostly have been already discussed in Chapter 3 and Chapter 5, to informally identify the fundamental algorithmic building blocks of these methods for a first overview. We will discuss the models behind these algorithmic building blocks in a more formal manner in the subsequent section. Since the normalization step is not present in many of these examples, let us note again that some methods to provide a normalization for those outlier detection models have been proposed by [Kri+11]. Some methods (e.g., DB-Outlier) directly derive a label (outlier vs. inlier) instead of performing some score normalization.

In summary, the identified common algorithmic scheme for local outlier detection consists of the following components:

1. **Context:** a "local" context of an object o for model building ($\mathrm{context}(o)$)
2. **Model:** the method used for building the model
3. **Reference:** a "reference" context of object o for model comparison ($\mathrm{reference}(o)$)
4. **Comparison:** the method used for model comparison
5. **Normalization:** a (global) normalization procedure

These are the common algorithmic building blocks of local outlier detection approaches and they enable us at the same time to see global approaches as special cases of local approaches

Table 6.1: Overview: local outlier methods.

Method	Context	Model
	Reference	Comparison
		Normalization
Outlier EM [TSK06]	global	EM-fitting of two models (Gaussian vs. uniform)
	global	cond. prob.
DB-Outlier [KNT00]	range	object count
	global	threshold
kNN-Outlier [RRS00]	kNN	maximum distance
	global	descending
kNN-Weight [AP02]	kNN	sum of distances
	global	descending
LOF [Bre+00]	kNN	reachability density
	kNN	avg. quotient
INFLO [Jin+06]	kNN	reachability density
	$k\mathrm{NN} \cap \mathrm{rev}.k\mathrm{NN}$	avg. quotient
reference points [PZG06]	k approx. NN	density estimate
	reference points	descending
		$1 - \frac{\mathrm{value}}{\max\{\mathrm{values}\}}$
LOCI [Pap+03]	range r_1	obj. count at any $r < r_1$
	range r_2	quot. with avg.
LDOF [ZHJ09]	kNN	distances quotient
	global	descending
LoOP [Kri+09a]	kNN	$1/\mathrm{RMSD}$
	kNN	$1/\mathrm{RMSD}$
		erf
LDF [LLP07]	kNN	Modified KDE
	kNN	Scaled avg. quotient
ABOD [KSZ08]	kNN	angle variance
	global	ascending
High.-dim. [AY01]	subspaces	object count
	global	threshold
SOD [Kri+09b]	SNN-based kNN	subspace model
	global	descending
subspace outlier ranking [MSS10]	adaptive range in subspace	subspace density model
	global, but subspace	deviation from exp. density
		$\frac{\mathrm{dens.}}{\mathrm{dev.}}$ if dev.> 1

(and, as we will discuss in the following, spatial outlier detection and even outlier detection in special data like video streams or network data can be seen as a specialization of this framework as well). Though not every component is actually used or present in every instance of outlier detection methods, many existing methods can be unified using this algorithmic framework.

6.2 Formalized Analysis of Outlier Detection Models

Before we inspect specializations (Sections 6.3 to 6.5) of the algorithmic framework, we now take a higher perspective and discuss a framework for formal analysis of outlier detection models (Section 6.2.1). We discuss common context functions, that are used to derive context sets or reference sets (Section 6.2.2). Applying the framework, we then derive formal descriptions of outlier detection algorithms, based on the successive execution of model functions (Section 6.2.3). Based on this framework for formal analysis, we discuss in two case studies similarities and differences among variants of LOF (Section 6.2.4) and improved understanding of a complex method (Section 6.2.5). Finally, we show by means of dependency graphs in a formal way the level of locality actually used in “local” outlier detection algorithms (Section 6.2.6).

6.2.1 Generalized Outlier Detection Model Framework

As we have seen in the analysis and generalization of existing work (Section 6.1), there are reoccurring patterns in outlier detection. The most prominent pattern is the computation of a model, based on a reference set of objects. Many established methods can be formulated to apply this pattern twice, and the normalization step can also be formatted to follow this pattern. As we will show here, this results in a general formal framework for analysis of outlier detection models. Within the framework, we focus on the properties of outliers, not on the actual computation of the scores. The formalization into multiple steps however allows the construction of generic algorithms that usually are of the same (or lower) complexity than the originally proposed algorithms.

In order to define the general model, we first need to define its basic components and building blocks. Let O be the objects in the database uniquely identified.

Definition 6.2.1 (Context Function)**:**
A context function c_i is a function to the powerset $\mathcal{P}$ of O

$$c_i : O \to \mathcal{P}(O)$$

that maps objects o to their context set $c_i(o) \subseteq O$, usually objects that are considered to be relevant for assessing the outlierness of o.

Definition 6.2.2 (Intermediate Data)**:**
For some value domain V_i, let $D_i(o) \in V_i$ with $i = 0 \dots n$ be the **intermediate data** of step i for object o, with $D_0(o) = o$ the initial data (identity map).

Definition 6.2.3 (Available Data)**:**
Let the collected intermediate data $\mathfrak{D}_i(o) := \{D_k(o)|k \leq i\}$ be the **available data** after step i.

Definition 6.2.4 (Model Function)**:**
A model function f_i is a function

$$f_i(o, c_i(o), \mathfrak{D}_{i-1}) =: D_i(o),$$

where o is the current object, $c_i(o)$ is the context set of the object, $\mathfrak{D}_{i-1}(o)$ is the available data before executing function f_i, and $D_i(o)$ is the output (model) data for object o.

Definition 6.2.5 (Algorithm step)**:**
An algorithm step p_i is the task of computing a model function for the whole database, and can be formalized as:

$$p_i : \mathfrak{D}_{i-1} \mapsto \{D_0, \ldots, D_i\} = \mathfrak{D}_i,$$

where the intermediate data D_i is the output of f_i for all objects:

$$D_i := \{o \mapsto f_i(o, c_i(o), \mathfrak{D}_{i-1})\}$$

Each step computes a new intermediate data set based on the existing intermediate maps and the new data obtained by computing f_i on all objects to obtain the new data set D_i. When executed in sequence, they transform the data as follows:

$$\underbrace{\{D_0\}}_{=\mathfrak{D}_0} \mapsto_{p_1} \underbrace{\{D_0, D_1\}}_{=\mathfrak{D}_1} \mapsto_{p_2} \underbrace{\{D_0, D_1, D_2\}}_{=\mathfrak{D}_2} \ldots \mapsto_{p_i} \underbrace{\{D_0, \ldots, D_i\}}_{=\mathfrak{D}_i}$$

Executing the steps p_i one after the other is called the **canonical algorithm** for computing the outlier result.

Definition 6.2.6 (Generalized Outlier Detection Model)**:**
A generalized outlier detection model is a series of model functions and context functions

$$[(f_0, c_0), \ldots, (f_i, c_i)] \,,$$

such that $D_i : O \to \mathbb{R}$ is the map onto the objects' outlier score.

Any outlier detection can trivially be captured in this model by using the outlier detection as an arbitrarily complex function f_0 with $c_0 = \mathit{global}$ being the full data set. In fact, the formalization allows to use any computable function this way. However, in the following we will show that we are able to model many well-known methods using much more primitive functions. Usually functions of complexity $\mathcal{O}(1)$ or $\mathcal{O}(|c_i(o)|)$, with a focus on analyzing the notion of locality used in the analyzed methods.

Definition 6.2.7 (Linear Generalized Outlier Detection Model)**:**
A Generalized Outlier Detection Model is called linear, if and only if for each step $p_i = (f_i, c_i)$ the complexity of f_i is at most in $\mathcal{O}(|c_i(o)|)$.

The canonical algorithm implied by this formalization can then compute a linear generalized outlier detection model in $\mathcal{O}(i \cdot |O| \cdot |c_i(o)|)$ plus the time needed to compute the context sets. This definition rules out using an existing complete, possibly complex, outlier detection algorithm as algorithm step. Note that we do not impose a constraint on computing the c_i, and indeed many of the popular vector space methods will take $\mathcal{O}(|O|^2)$ time without index support to compute all context sets and reference sets. However for graph data, the c_i are part of the input data, so it makes sense to treat the context set computation separately. To fully control complexity, it may be desirable to put a limit on the context set size, for example by assuming $|c_i| \ll |O|$. However we will see a number of cases where it is more understandable to specify the context set as global, even when the total computation is still linear in the number of objects – for example when outlier scores are normalized. Therefore, this can only be used as a rough estimation of the total complexity of the canonical algorithm, or this optimization of sharing a computation among multiple observations as an optimization of the canonical algorithm. From a mathematical point of view, in such a normalization step we do have a dependency of each object to every other object. This captures the fact that a single object change in the data set may change the normalization (for example by changing the minimum or maximum value). The main motivation of the context set is to capture this *dependency*, not the algorithmic complexity. Hence, we do not propose this formalization as a generalization of whatsoever outlier detection models. Rather, the formalization allows to decompose many existing outlier detection models in their simple steps and, by means of this decomposition, it allows to *analyze* many existing methods and to state their similarities to each other or their essential differences and individual merits. By extracting the simple building blocks, using the formalism also allows for simple complexity analysis of the baseline algorithm given implicitly by executing the algorithm steps one after the other.

Let us therefore emphasize that we are not actually proposing this general model as a new method that is able to express everything, but as a *means of analysis of existing (and future) methods.* Accordingly, in the following, we analyze existing methods by this formalization, in order to survey important abstract outlier detection methods; then we relate this to specialized methods by demonstrating the applicability of the model to specialized notions of locality (spatial data, video sequences, graph data) in Sections 6.3, 6.4, and 6.5reproducing the results of state of the art methods in these quite diverse fields with a straightforward baseline method (built as composition in the formalized general outlier detection model).

In the following, we introduce a number of example functions that can be used to define many well-known outlier detection models in a uniform manner. The functions are summarized in Table 6.2 (context functions) and Table 6.3 (model functions).

6.2.2 Fundamental Context Functions

The definition of the local context is essential for local outlier detection; however it is essentially an input parameter to most outlier detection methods. Locality is commonly defined using the k nearest neighbors for a given distance function d, a range query with a radius of ε, a

Table 6.2: Common context definitions.

Context function	Definition
$\mathrm{range}_{d,\varepsilon}$	range query with distance function d and radius ε
$k\mathrm{NN}_{d,k}$	k nearest neighbor query with distance function d
$k\mathrm{NN}'_{d,k}$	k nearest neighbor query with d, excluding query object
$k\text{-distinct}_{d,k}$	k nearest distinct neighbors with distance function d
$\mathrm{r}k\mathrm{NN}_{d,k}$	reverse k nearest neighbor query with distance function d
$\mathrm{SNN}_{d,s,k}$	k best with respect to the shared $k\mathrm{NN}_{d,s}$
global	O (the complete database)
$\emptyset$	the empty set
spatial	spatial neighborhood (predefined)
prev_k	previous k objects (temporal)

Table 6.3: Common model functions.

Model function	Definition
$\mathrm{count}(o, c(o), \mathfrak{D})$	$\lvert c(o)\rvert$
$\mathrm{maxdist}_{i,d}(o, c(o), \mathfrak{D})$	$\max_{n\in c(o)} d(D_i(o), D_i(n))$
$\mathrm{avgdist}_{i,d}(o, c(o), \mathfrak{D})$	$\mathrm{mean}_{n\in c(o)} d(D_i(o), D_i(n))$
$\mathrm{pairdist}_{i,d}(o, c(o), \mathfrak{D})$	$\frac{1}{\lvert c(o)\rvert\cdot(\lvert c(o)\rvert-1)} \sum_{n\in c(o)} \sum_{m\in c(o), m\neq n} d(D_i(n), D_i(m))$
$\mathrm{lrd}_{i,j,d}(o, c(o), \mathfrak{D})$	$1/\,\mathrm{mean}_{n\in c(o)} \max\{D_j(n), d(D_i(o), D_i(n))\}$
$\mathrm{mean}_i(o, c(o), \mathfrak{D})$	$\mathrm{mean}_{n\in c(o)} D_i(n)$
$\mathrm{stddev}_i(o, c(o), \mathfrak{D})$	$\mathrm{stddev}_{n\in c(o)} D_i(n)$
$\mathrm{frac}_{i,j}(o, c(o), \mathfrak{D})$	$\frac{D_i(o)}{D_j(n)}$
$\mathrm{pdist}_{i,d}(o, c(o), \mathfrak{D})$	$\lambda\sqrt{\mathrm{mean}_{n\in c(o)} d(D_i(o), D_i(n))^2}$
$\mathrm{erf}'_{i,\lambda}(o, c(o), \mathfrak{D})$	$\max\left\{0, \mathrm{erf}\left(\frac{1}{\sqrt{2}} \frac{D_i(o)}{\lambda\cdot\sqrt{\mathrm{mean}_{n\in c(o)} D_i(n)^2}}\right)\right\}$
Utility function	**Definition**
$\mathrm{mean}_{o\in O} f(o)$	$\frac{1}{\lvert O\rvert} \sum_{o\in O} f(o)$ (arithmetic mean of f in O)
$z(p, f, O)$	$(f(p) - \mathrm{mean}_{o\in O} f(o))/\mathrm{stddev}_{o\in O} f(o)$ (standard score)

spatial neighborhood based on graph adjacency or polygon adjacency, or a temporal context, e.g. in terms of a sliding window. Sometimes, there are slight variations. For example the k nearest neighbors may or may not include the query object itself, may consist of exactly k neighbors (which might not be uniquely defined) or may include additional neighbors that share the identical distance with the kth neighbor. We do not cover all of these variations here. Some methods implicitly assume that there are no objects with a distance of 0, and may even divide by 0 when there are more than k objects at distance 0.

Table 6.2 lists these contexts without a detailed formalization (which is trivial in these cases). For completeness, we also include the complete database (denoted as global) or no objects (denoted as $\emptyset$) as trivial contexts. This allows for an improved reuse of model functions and restricts the number of required specializations.

6.2.3 Fundamental Model Functions

The key building blocks of an outlier detection model are the model functions that compute key properties. Many will output into the real number domain, although complex models such as covariance matrices are possible. Here, we define a number of commonly used functions. Additional functions are summarized in Table 6.3. We loosely follow chronological order for these methods, which largely reflects their complexity as well. When these blocks are then combined into outlier detection models, overlaps and similarities between models will become visible.

The initial distance-based outlier (DB-outlier) definition by [KNT00] did not yet address "local" outlier detection, but provided a binary decision based on a threshold on the relative number of objects outside a given radius. By turning the density threshold at which a point would become a DB-outlier into a score it becomes a ranking outlier detection method, as introduced by [Kri+11].[1] While locality was not discussed explicitly, the dependence on the distance function implies a certain degree of locality. When formalizing DB-outliers, the essential building block is to count the number of objects within the query range (the context of the object), which will be the first example for a model function:

Definition 6.2.8 (Object Count Model Function)**:**

$$\mathrm{count}(o, c(o), \mathfrak{D}) := |c(o)|$$

Definition 6.2.9 (Scoring DB-outlier)**:**
Scoring DB-outlier [KNT00; Kri+11] is a linear generalized outlier detection model for distance function d and range ε with

$$\text{DB-Outlier}(d, \varepsilon) = \big[(\mathrm{count}, \mathrm{range}_{d,\varepsilon})\big]$$

Instead of using the number of objects as a score, a different way of turning DB-outliers into a scoring method is to use the radius at which the number of neighbors would suffice the DB-outlier definition as score. For outliers, a much larger neighborhood would be required, for inliers a smaller distance would be sufficient. [RRS00] formalized this notion of outliers, out of which we extract the next model function, which computes the maximum distance to an object of the context set:

Definition 6.2.10 (Maximum Distance Model Function)**:**

$$\mathrm{maxdist}_{i,d}(o, c(o), \mathfrak{D}) := \max_{n \in c(o)} d(D_i(o), D_i(n))$$

Definition 6.2.11 (kNN-Outlier)**:**
kNN-Outlier [RRS00] is a linear generalized outlier detection model for distance function d and neighborhood size k with

$$k\mathrm{NN}(d, k) = [(\mathrm{maxdist}_{0,d}, k\mathrm{NN}_{d,k})]$$

[1]This adaptation could also be considered "Schönfinkeling" (or, tastier, "Currying") of the DB-outlier model.

This work was then again generalized and extended by [AP02] to improve stability by taking the average (or sum) instead of the maximum distance of the neighborhood:

Definition 6.2.12 (Average Distance Model Function)**:**

$$\operatorname{avgdist}_{i,d}(o, c(o), \mathfrak{D}) := \operatorname{mean}_{n\in c(o)} d(D_i(o), D_i(n))$$

Definition 6.2.13 (kNN-Weight)**:**
kNN-Weight outlier [AP02] is a linear generalized outlier detection model for distance function d and neighborhood size k with

$$\mathrm{A}k\mathrm{NN}(d, k) = \left[(\operatorname{avgdist}_{0,d}, k\mathrm{NN}_{d,k})\right]$$

Up to now, our framework was only able to represent the known two methods. The additional combinations – computing the number of objects in the k-neighborhood and computing the maximum distance within a fixed radius – were of little interest. This model function actually allows us to consider an interesting new combination: Computing the average distance within a fixed radius around an object could be a reasonable score, assuming that an outlier will likely have less close and more distant neighbors. This method is however not very useful in practice, since the radius parameter is particularly hard to choose, and the value becomes unstable when there are only few neighbors available. The combination by [AP02] with a fixed size neighborhood is much more reasonable.

So far, the algorithms were essentially identical to the application of the model function onto the context of an object. The first method known to use a more complex approach is the Local Outlier Factor [LOF, Bre+00]. Instead of just computing a local score on the object itself, it in fact computes a particular property – a density estimation – for each object; then again compares these values within the local neighborhood. For modeling LOF we need a total of four model functions, one of which we have seen before in Definition 6.2.10: what is called k-distance in LOF is essentially the $\operatorname{maxdist}_{i,d}$ function that the kNN-Outlier method by [RRS00] used. It serves a stabilizing role in LOF, and we will then discuss how it can be removed to obtain a "simplified LOF" method (which has actually been used – probably unintentionally – in many approaches that allegedly were based on the original LOF idea, see the case study in Section 6.2.4). The second model function of LOF computes a density model known as "local reachability density" (c.f. Equation 3.6) and is the key component of LOF:

Definition 6.2.14 (Local Reachability Density Model Function)**:**

$$\operatorname{lrd}_{i,j,d}(o, c(o), \mathfrak{D}) := 1/\operatorname{mean}_{n\in c(o)} \max\{D_j(n), d(D_i(o), D_i(n))\},$$

where mean denotes the arithmetic mean operator and D_j is the $\operatorname{maxdist}_{i,d}$ result obtained before, while d is the distance function applied to the objects in D_i.

The other two model functions required for the definition of LOF are very basic operations that we will however see in many of the following methods, the computation of a mean value over the neighborhood and a simple, context-free comparison step for simple numeric models by computing the fraction:

Definition 6.2.15 (Mean Model Function)**:**

$$\mathrm{mean}_i(o, c(o), \mathfrak{D}) := \mathrm{mean}_{n \in c(o)} D_i(n)$$

Definition 6.2.16 (Fraction Model Function)**:**

$$\mathrm{frac}_{i,j}(o, \_, \mathfrak{D}) := \frac{D_i(o)}{D_j(o)}$$

These four model functions (Definitions 6.2.10, 6.2.14, 6.2.15 and 6.2.16) can now be connected together to form the LOF model:

Definition 6.2.17 (Local Outlier Factor)**:**
LOF [Bre+00] is a linear generalized outlier detection model for distance function d and neighborhood size k with

$$\mathrm{LOF}(d, k) =[(\mathrm{maxdist}_{0,d}, k\mathrm{NN}_{d,k}), (\mathrm{lrd}_{0,1,d}, k\mathrm{NN}_{d,k}), (\mathrm{mean}_2, k\mathrm{NN}_{d,k}), (\mathrm{frac}_{3,2}, \emptyset)]$$

Note the chaining of operations given by the indices on the operators: the maximum distance is computed on the original data, the local reachability density uses the original data and this maximum distance, the final step puts the density models only into relation with each other. We will use this later to obtain a dependency graph representation of the models.

6.2.4 Case Study: Variants of LOF

For LOF, we have pointed out the often overlooked detail of the reachability distance. A variation commonly seen in LOF extensions is to drop the second model function of LOF and use a much simpler density estimation instead, resulting in the following base model of a density-quotient outlier model:

Definition 6.2.18 (Simplified Local Outlier Factor)**:**
Simplified-LOF is a linear generalized outlier detection model for distance function d and neighborhood size k with

$$\text{Simplified-LOF}(d, k) =[(1/\mathrm{maxdist}_{0,d}, k\mathrm{NN}_{d,k}), (\mathrm{mean}_1, k\mathrm{NN}_{d,k}), (\mathrm{frac}_{2,1}, \emptyset)].$$

In this baseline method, local density is estimated by the inverse of the k-distance, and the combination of mean and frac forms the core of most algorithms that reference LOF.

Given that LOF is clearly distance-based and the article introducing the Local Distance-Based Outlier Factor (LDOF) [ZHJ09] compares the algorithm to LOF, one would expect a strong overlap of these methods. Our representation shows that it actually is a variation of the Simplified-LOF: instead of taking the maximum distance, LDOF uses the $\mathrm{avgdist}$ model for estimating the local density, and instead of the mean density it uses pairwise distances for estimating a neighborhood density:

Definition 6.2.19 (Pairwise Distance Model Function)**:**

$$\operatorname{pairdist}_{i,d}(o, c(o), \mathfrak{D}) := \frac{1}{|c(o)| \cdot (|c(o)| - 1)} \sum_{n,m \in c(o), m \neq n} d(D_i(n), D_i(m))$$

We can now combine these to form LDOF:

Definition 6.2.20 (Local Distance-based Outlier Factor)**:**
LDOF [ZHJ09] is a linear generalized outlier detection model:

$$\operatorname{LDOF}(d, k) = \left[(\operatorname{avgdist}_{0,d}, k\mathrm{NN}'_{d,k}), (\operatorname{pairdist}_{0,d}, k\mathrm{NN}'_{d,k}), (\operatorname{frac}_{1,2}, \emptyset)\right]$$

A different kind of variation of Simplified-LOF is Influenced Outlierness (INFLO) [Jin+06], which diverges from Simplified-LOF by using a different context set for its second model function:

Definition 6.2.21 (Influenced Outlierness)**:**
INFLO [Jin+06] is a generalized outlier detection model for distance function d and neighborhood size k with

$$\operatorname{INFLO}(d, k) = [(1/\operatorname{maxdist}_{0,d}, k\mathrm{NN}_{d,k}), (\operatorname{mean}_1, k\mathrm{NN}_{d,k} \cap \mathrm{r}k\mathrm{NN}_{d,k}), (\operatorname{frac}_{2,1}, \emptyset)]$$

Another Simplified-LOF variation that is more interesting for our framework since it introduces a new kind of model function is the Local Outlier Probabilities model (LoOP) [Kri+09a], which includes an additional normalization step based on the assumption that the quotient scores are normally distributed (with a fixed mean). This step is interesting, because it involves the complete data set as context for estimating the distribution parameter. Alternative normalization functions that can directly replace this model function have been studied in detail in [Kri+11] and are discussed in Section 5.3.

Definition 6.2.22 (Error Function Normalization Model Function)**:**

$$\operatorname{erf}'_{i,\lambda}(o, c(o), \mathfrak{D}) := \max\left\{0, \operatorname{erf}\left(\frac{1}{\sqrt{2}} \frac{D_i(o)}{\lambda \cdot \sqrt{\operatorname{mean}_{n \in c(o)} D_i(n)^2}}\right)\right\}$$

Additionally, LoOP uses a different density estimation function, that assumes a half-Gaussian distribution of the local distances:

Definition 6.2.23 (Probability Density Model Function)**:**

$$\operatorname{pdist}_{i,d}(o, c(o), \mathfrak{D}) := \lambda \sqrt{\operatorname{mean}_{n \in c(o)} (d(D_i(o), D_i(n)))^2}$$

Definition 6.2.24 (Local Outlier Probabilities)**:**
LoOP [Kri+09a] (see also Section 5.1.1) is a linear generalized outlier detection model:

$$\operatorname{LoOP}(d, k, \lambda) = [(\operatorname{pdist}_{0,d}, k\mathrm{NN}_{d,k}), (\operatorname{mean}_1, k\mathrm{NN}_{d,k}), (\operatorname{frac}_{1,2} - 1, \emptyset), (\operatorname{erf}'_{2,\lambda}, \mathrm{global})]$$

Again, the second and third model functions are an easily recognizable pattern of Simplified-LOF. The first model function (the quadratic mean of the distance) and the fourth model function, which serves as global normalization step, are the key contributions of this method.

Another recent example for (unintentional?) use of Simplified-LOF instead of the actual LOF model is Projection-Indexed Nearest-Neighbors (PINN) [dCH10].

Thus, overall, this case study may demonstrate that a better understanding of the actually used outlier model and the adopted notion of locality in some algorithm may help to reveal the relationships, similarities, and differences between some approaches in the literature.

6.2.5 Case Study: Plot Models as Used by LOCI

A well known method that uses a more complex model – so far, all model functions in fact were mappings onto the real numbers – is the Local Correlation Integral (LOCI) [Pap+03]. Where LOF only used the $\mathrm{maxdist}$ function (Definition 6.2.10) with a parameter k, LOCI uses a plot consisting of radius/count pairs that give the number of neighbors within the given radius. We represent these plots as a map with the signature $r \mapsto v$, mapping a radius r to a value v. Again we present a slight generalization of LOCI that instead of producing a binary result for a given threshold (k_σ in LOCI) produces a score representing the threshold value of k_σ where the point would become an outlier. Some fine details such as the minimum radius $r_{\min}$ and minimum neighborhood size $\hat{n}_{\min}$ were also omitted for brevity, as was the computation of interesting values for r.

LOCI uses a – difficult to grasp – interplay of two radii, r and αr. In general, the radius of αr is used for density estimation, the radius of r is used as reference set. The first model function for LOCI is the density estimation using the modified radius. We deliberately assign it to the radius of r to simplify the whole LOCI model, reducing the use of α to this single occurrence.

Definition 6.2.25 (Density Plot Model Function)**:**

$$\mathrm{denplot}'_{d,\alpha}(o, c(o), \mathfrak{D}) := r \mapsto |\{n \in c(o) \wedge d(n,o) < \alpha r\}|$$

Similar to the mean function, these plots are averaged over their neighbor sets to obtain a type of mean count integral, taking only those neighbors into account that are within the given radius (the use of α now is hidden in D_i).

Definition 6.2.26 (Plot Mean Model Function)**:**

$$\mathrm{plotmean}_{i,d}(o, c(o), \mathfrak{D}) := r \mapsto \mathrm{mean}_{p \in c(o) \wedge d(p,o) < r}\, D_i(p)(r)$$

The quantity denoted as σ_{MDEF} in LOCI is the corresponding standard deviation, normalized additionally by the mean.

Definition 6.2.27 (MDEF Standard Deviation Model Function)**:**

$$\mathrm{sigmdef}_{i,j,d}(o, c(o), \mathfrak{D}) := r \mapsto \frac{\mathrm{stddev}_{p \in c(o) \wedge d(p,o) < r}\, D_i(p)(r)}{D_j(o)(r)}$$

As we will be able to see below, the normalization is not needed, at which point we just have the common standard deviation formula:

Definition 6.2.28 (Plot Standard Deviation Model Function)**:**

$$\mathrm{plotstddev}_{i,d}(o, c(o), \mathfrak{D}) := r \mapsto \mathrm{stddev}_{p \in c(o) \wedge d(p,o) < r}\, D_j(o)(r)$$

The comparison step is then another quotient function, resulting in the MDEF plot, by computing the quotient of the object count to the mean object count, where lower values than 1 indicate outlierness.

Definition 6.2.29 (MDEF Plot Model Function)**:**

$$\begin{aligned}\mathrm{plotmdef}_{i,j}(o, \_, \mathfrak{D}) :=& r \mapsto 1 - \frac{D_i(o)(r)}{D_j(o)(r)} \\ \equiv& r \mapsto \frac{D_j(o)(r) - D_i(o)(r)}{D_j(o)(r)}\end{aligned}$$

We carry out the equivalent simplification as we did in Definition 6.2.28:

Definition 6.2.30 (Plot Delta Model Function)**:**

$$\mathrm{plotdelta}_{i,j}(o, \_, \mathfrak{D}) := r \mapsto D_j(o)(r) - D_i(o)(r)$$

The value is finally normalized by taking the local MDEF standard deviation into account. Applying this function to the results of Definitions 6.2.27 and 6.2.29 is equivalent to applying it to the results of Definitions 6.2.28 and 6.2.30:

Definition 6.2.31 (Plot Fraction Model Function)**:**

$$\mathrm{plotfrac}_{i,j}(o, \_, \mathfrak{D}) := r \mapsto D_i(o)(r) / D_j(o)(r)$$

LOCI considers points to be outliers based on their highest MDEF score, which can be turned into a model function reducing the plot to a single score

Definition 6.2.32 (Plot Maximum Model Function)**:**

$$\mathrm{plotmax}_i(o, \_, \mathfrak{D}) := \max_r D_i(o)(r)$$

Definition 6.2.33 (Local Correlation Integral)**:**
LOCI [Pap+03] is a generalized outlier detection model:

$$\begin{aligned}\mathrm{LOCI}(d, r, \alpha) = &[(\mathrm{denplot}'_{d,\alpha}, \mathrm{range}_{d,\alpha r_{\max}}),\\ &(\mathrm{plotmean}_{1,d}, \mathrm{range}_{d,r_{\max}}), (\mathrm{sigmdef}_{1,2,d}, \mathrm{range}_{d,r_{\max}}),\\ &(\mathrm{plotmdef}_{1,2}, \emptyset), (\mathrm{plotfrac}_{4,3}, \emptyset), (\mathrm{plotmax}_5, \emptyset)]\end{aligned}$$

An equivalent definition can be given using the simplified formulas and standard functions:

Definition 6.2.34 (Local Correlation Integral (equiv.))**:**
LOCI [Pap+03] is a generalized outlier detection model:

$$\begin{aligned}\mathrm{LOCI}(d, r, \alpha) \equiv &[(\mathrm{denplot}'_{d,\alpha}, \mathrm{range}_{d,\alpha r_{\max}}),\\ &(\mathrm{plotmean}_{1,d}, \mathrm{range}_{d,r_{\max}}), (\mathrm{plotstddev}_{1,d}, \mathrm{range}_{d,r_{\max}}),\\ &(\mathrm{plotdelta}_{1,2}, \emptyset), (\mathrm{plotfrac}_{4,3}, \emptyset), (\mathrm{plotmax}_5, \emptyset)]\end{aligned}$$

At this point, we do not only have a clear understanding what LOCI actually computes which is much more difficult from the formulas in the original publication. We can also see that it does follow the common pattern of firstly computing a local feature – here the mass of the neighborhood for a radius αr, as per Definition 6.2.25 – and secondly comparing the deviation of this value from the value of its neighbors (this time with radius r) and normalizing it. Finally, it does all this for a variety of radius values, using the maximum score obtained.

6.2.6 Dependency Graph and Order of Locality

Analyzing the outlier models we have seen so far in this uniform framework leads to the obvious question on how to make use of this structural knowledge. We have seen that the model functions are shared among various algorithms, that we can do new recombinations such as Simplified-LOF, and that we can easily identify building blocks used in different methods (which is not always clear from the original publications). For example, kNN-Outlier detection is the first model function of LOF, so when we are computing LOF we implicitly also compute the kNN-Outlier score. Context sets such as the kNN context are used throughout the algorithms and can sensibly be precomputed. In fact, all model functions (with the sole exception of LOCI) that we have seen so far can then be computed in essentially linear time of the database size (more precisely, most algorithms are in $\mathcal{O}(|O| \cdot k)$ then), the expensive step in all of these algorithms is the computation of the context set: computing the kNN of an object has the complexity $\mathcal{O}(|O|^2)$ when done naïvely and $\mathcal{O}(|O| \log |O|)$ when an appropriate index structure is available.

In the model functions as written in our definitions, we also implicitly denoted a dependency graph that can help us analyzing the corresponding algorithms for efficient implementation.

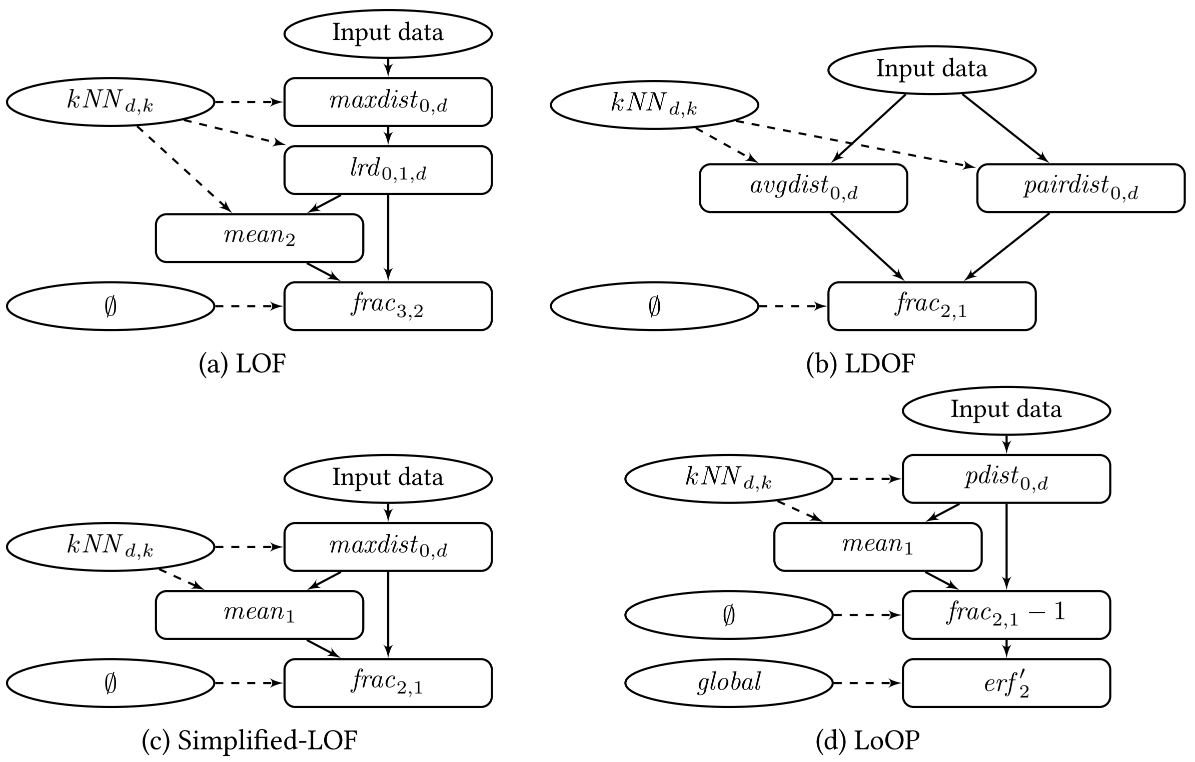


Figure 6.2: Graphical dependency graphs for LOF variations. The relationship of LOF, Simplified-LOF and LoOP are easy to recognize, while LDOF differs much more.

The functions often have integer indexes that reference previous results, which directly encodes the dependency graph. A dependency in this graph means that a model function f_i uses the intermediate data D_j produced by a model function $j < i$. In addition to the dependencies between model functions, we also include dependency edges to the input data and context functions. Figure 6.2 contains explicit dependency graphs of the methods LOF, LDOF, LoOP and Simplified-LOF. Model functions are represented as rounded boxes, while input data D_0 and context functions are represented as ellipses. Let us define the dependency graph formally.

Definition 6.2.35 (Dependency Graph)**:** Let G be the dependency graph of the algorithm defined by the relations:

$$(j \longleftarrow i) \in G \Leftrightarrow f_i \text{ depends on } D_j$$
$$(j \dashleftarrow c_j) \in G \Leftrightarrow f_i \text{ depends on } c_j$$

Steps that do not depend on each other can trivially be computed in parallel (an example for this situation can be found in Figure 6.2b for LDOF, where the $\mathrm{avgdist}$ and $\mathrm{pairdist}$ functions can be computed in parallel).

From the dependency graphs depicted in Figure 6.2 it can be clearly seen that LoOP extends Simplified-LOF with the normalization step, while the similarity of LDOF to LOF is mostly in using the $k\mathrm{NN}$ and the frac function (as elaborated in Section 6.2.4). In Simplified-LOF for example, the frac function depends on the results of both the first and second model function (which in turn depended on the first function). Since the fraction however has an empty set, when implementing this method we can compute the second and third model function in a single iteration over the data, without the need to store the intermediate result of the second model function. We can unfortunately not apply the same trick to the first model function: the second function needs its value for more than one object at a time. When we choose to combine the two functions into one, we will have to invoke the first function much more often (if the $k\mathrm{NN}$ contexts are precomputed, it is of course affordable to recompute the maximum function). This analysis shows us two different strategies for evaluating Simplified-LOF (and most similar methods): If we precompute the $k\mathrm{NN}$ contexts, we can essentially compute the LOF outlier score in a single pass over the data set. If we cannot afford the precomputation due to storage costs but are able to store the intermediate results of the model functions, we can still avoid extensive $k\mathrm{NN}$ computations. Only when we have extremely little memory available, we may need to compute the $k\mathrm{NN}$s of $k\mathrm{NN}$ by evaluating the whole outlier model for each point independently.

Another analysis that we can perform on this graph is to determine a notion of locality complexity of an outlier detection model:

Definition 6.2.36 (Order of Locality of a Path)**:**
Given a Generalized Outlier Detection Model $[(f_0, c_0), \ldots, (f_n, c_n)]$ and its dependency graph $G = \{(i \leftarrow j)\}$. Then the order of locality of a path $L(a_k \leftarrow a_{k-1} \leftarrow \ldots a_1)$ with $a_i \in \{1 \ldots n\}$ is defined as:

$$L(a_j) := \begin{cases} 0 & \text{iff } c_{a_j} \in \{\emptyset, \text{global}\} \\ 1 & \text{otherwise} \end{cases}$$
$$L(a_k \leftarrow a_{k-1} \leftarrow \ldots a_1) := L(a_k) + L(a_{k-1} \leftarrow \ldots a_1)$$

Definition 6.2.37 (Order of Locality of a Model)**:**
Given a Generalized Outlier Detection Model $[(f_0, c_0), \ldots, (f_n, c_n)]$ and its dependency graph $G = \{(i \leftarrow j)\}$, the **Order of Locality** of the model is defined as the maximum order of locality of all paths in the model.

Based on this, we can obtain the order of locality of the analyzed outlier models. We give some examples here:

Proposition 6.2.1 (Order of Locality of kNN-Outlier)**:**
The *order of locality* of the kNN-Outlier model [RRS00] is 1.

Proof. Prop. 6.2.1 follows directly from Def. 6.2.37 and Def. 6.2.11. □

Proposition 6.2.2 (Order of Locality of LDOF)**:**
The *order of locality* of the LDOF outlier model [ZHJ09] is 1.

Proof. Prop. 6.2.2 follows directly from Def. 6.2.37 and Def. 6.2.20. □

Proposition 6.2.3 (Order of Locality of Simplified-LOF)**:**
The *order of locality* of the Simplified-LOF outlier model (Def. 6.2.18) is 2.

Proof. Prop. 6.2.3 follows directly from Def. 6.2.37 and Def. 6.2.18. □

Proposition 6.2.4 (Order of Locality of LoOP)**:**
The *order of locality* of the LoOP outlier model [Kri+09a] is 2.

Proof. Prop. 6.2.4 follows directly from Def. 6.2.37 and Def. 6.2.24. □

Proposition 6.2.5 (Order of Locality of LOF)**:**
The *order of locality* of the LOF outlier model [Bre+00] is 3.

Proof. Prop. 6.2.5 follows directly from Def. 6.2.37 and Def. 6.2.17. □

These findings align with our intuition that LOF takes locality more into account than the other models, and reflects the simplification of Simplified-LOF. This bears repercussions for the order of locality of all outlier models that pretend to be a variant of LOF but actually are based on Simplified-LOF (as discussed in Section 6.2.4).

Let us note that these findings do not state any superiority of LOF. We do not imply that "more" locality is "better" in any way. But we state that methods of different orders of locality model outliers in truly different ways. This should be recognized and taken into account when using these models (be it for applications or as role models for adapted outlier detection models). In the following, we show application scenarios, where the notion of locality is adapted to make basic outlier models suitable for complex data.

6.3 Locality and Spatial Outliers

Spatial outlier detection has grown as a field of its own interest over several years [Ans95; SLZ03; LCK03; KLC06; SC04; CS06; LLC10; CLB10]. A key result of [Ans95] is the generalization of the global spatial association statistics Moran's I [Mor50] and Geary's C [Gea54] to individual contributions denoted as the local Moran I_i and local Geary C_i, then using a statistical test to identify strong contributions. Additionally the Moran scatterplot was introduced, which plots the locally standardized attribute value against the globally standardized attribute. In this plot, objects close to the regression line indicate consistency with the trend, whereas objects in the upper left and bottom right areas appear different on local and global scales. This concept of comparing local with global scores can be found in many newer methods in slight variations, others however just use the local scores proposed here directly or with only slight modifications.

The idea is to separate spatial attributes from other attributes, compute the neighborhood w.r.t. the spatial attributes solely but compare the non-spatial attributes only to derive a notion of outlierness. Most of these methods use a local neighborhood based on the spatial attributes solely in order to extract a score (the simplest type of model) for the object using the non-spatial attributes only. Many methods can just process a single non-spatial attribute, and there are rather few methods that use a model more complex than a preliminary score or a non-trivial comparison step, but we will highlight some examples in this section.

With our framework, this notion of spatial outliers can be implemented quite easily and straightforward. Actually, even some methods that at first appear to follow an entirely different approach can be rewritten to follow this schema, sometimes even significantly improving both the understanding and performance of the algorithm.

6.3.1 Data

For the experiments we used the US Census data[2], which are available at various granularities. For the population data, we used the Census County Subdivisions level, which often subdivides counties in particular in densely populated areas resulting in $36,351$ objects. The partitioning is based on administrative regions which occasionally do form a grid but also often follow geographical boundaries such as rivers. We analyzed the ethnic group attributes consisting of 5 dimensions representing ethnic groups in the population that approximately sum up to 1.0 (people may belong to multiple ethnic groups). There are many more attributes available, but these may require specialized algorithms and extra normalization; 5-dimensional normal vectors is a data type algorithms can be expected to work well with. Since some algorithms are not performing that well and may only be able to analyze single attributes, we additionally used a data set representing land use homogeneity at a county level containing $3,109$ records derived from satellite imaging.[3] For the third data set we use unemployment rates in Germany[4] again at a county level. The data set has a single dimension, 412 records ranging from 2.2 (full employment) to 18.3 with a median value of 7.0 and a skewed distribution.

6.3.2 Neighborhood in Spatial Outlier Detection

Most algorithms use the k nearest neighbors in the spatial attributes as neighborhood set, because of the uniform size. When polygon data are available, polygon intersection and shared borders (also known as the “meets” relation, in line with Allen’s Interval Algebra) are obvious choices, that may however result in objects such as islands not having any neighbors at all, or just a single neighbor when e.g. a city is contained within a single county polygon. Rarely, more complex notions of neighborhood are encountered such as Voronoi cells as proposed e.g.

[2]Available at the U.S. Census Bureau, United States Department of Commerce. `http://www.census.gov/`
[3]Available at the Arizona State University GeoDa Center, `http://geodacenter.asu.edu/`
[4]Available at Statistisches Bundesamt Deutschland, `http://www.destatis.de`

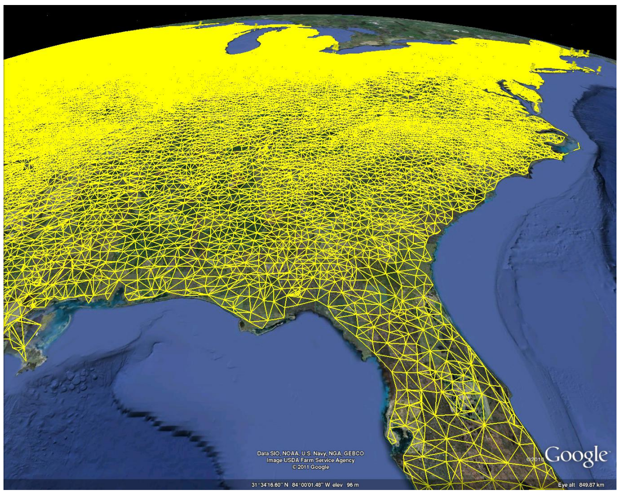


Figure 6.3: Neighborhood Graph in Google Earth.
Background © 2011 Google.

by [LLC10]. In the latter two cases, it can be desirable to expand the neighborhood to the t-fold transitive of the original relation to obtain a sensible neighborhood size. Figure 6.3 shows a 1-neighborhood graph for the high resolution census data set. Neighborhoods can for example be defined using length-limited paths in this graph. Occasionally, measures obtained during computation of the neighborhood (such as shared border length and distance) can be reused for weighting in the model generation phase.

Often, the definition of the actual neighborhood is part of a proposed method. It is, however, obvious that the choice of a concrete method for neighborhood computation will have a huge impact on the results of any spatial data mining method and hence should be considered independently from the algorithmic approach and the outlier model. In the following, we discuss the properties of models and comparison methods as well as the impact of using a local or a global context and reference irrespective of the definition of spatial neighborhood (we use the same definition of spatial neighborhood for all methods in the evaluation). For the US census data, neighbors are defined using polygon adjacency expanded to 5 steps for census county subdivisions (which often are very fine-grained) and 3 steps for county level. For the data set on Germany, we also used 3 steps at a county level.

Neighborhood computations, all compared methods, and our generalized and adapted methods are implemented in the ELKI framework [Ach+11; Ach+12].

6.3.3 Models in Spatial Outlier Detection

The common model used in spatial outlier detection is a deviation score, where the non-spatial attribute value(s) of an object are compared to the attributes of the neighbors. For example, [SLZ03; LCK03] propose various simple statistics such as the quotient of the attribute value and the neighbors' attribute mean, the difference of the attribute value to this mean, and the standardized deviation of the attribute value from the neighbors' attribute median. [KLC06] extend the model to the difference from a weighted mean normalized as z-score and the weighted mean difference (without normalization). However, the model is then (in some of the proposed methods) iteratively refined by replacing attribute values with their expected value, making the result of the algorithm much less predictable. As such, there is no formal definition of what actually constitutes an outlier for these methods, except the given algorithm.

In other cases, the model is less obvious: POD [KLD07] is described as a graph-based method. For each object, the k nearest neighbors are determined, which are used as edges of a graph. The weight of each edge is based on the similarity of the non-spatial attribute. The edges are managed in a global priority queue, and are successively removed until some objects become isolated, which are returned as outliers. The complexity of the algorithm is given as $\mathcal{O}(kn \log(kn))$ due to managing the priority queue of size kn.

However, this algorithm can be transformed into our framework easily. In essence, the edges are processed by their length, longest first. An object becomes an outlier, when the last edge is removed. The other edges essentially do not play any role, and do not need to be managed. Therefore, we chose to model each object by its shortest edge – a very simple local score – and obtain the same ranking. Instead of managing the complete priority queue, we can directly compute the length at which an object becomes an outlier. By comparing these lengths globally we obtain the identical ranking, and the top m results equal those of POD. The runtime however is reduced to $\mathcal{O}(n \log n)$, dominated by performing one kNN query for each object. Most of the outlier detection algorithms have this complexity, since the model generation and comparison steps usually are much faster than a kNN query.

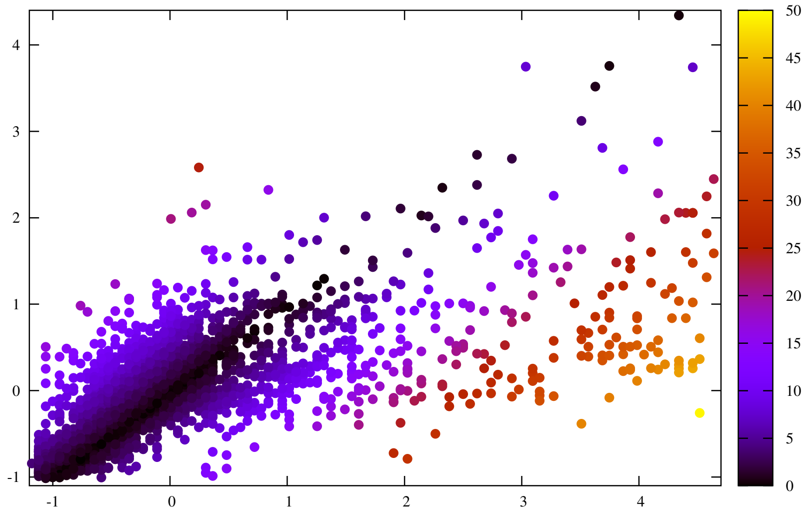
These examples show how using the framework allows for a deeper understanding of the algorithms' actual results (including a more formal definition of what constitutes an outlier) as well as a canonical way of computing the results that is usually cheaper than computing the neighborhoods.

In Table 6.4, we give a summarized overview of the models used by various spatial outlier detection methods proposed. We use a number of shorthand notions for brevity such as $\pi(p)$ for the non-spatial projection of an object, ω_n for a weight assigned to a neighbor, $E_{\mathcal{N}}(\ldots)$ for the expected value (usually the mean), and z-score, i.e., $z(x) := \frac{x-\mu_X}{\sigma_X}$, for the standard score normalization that assumes a normal distribution.

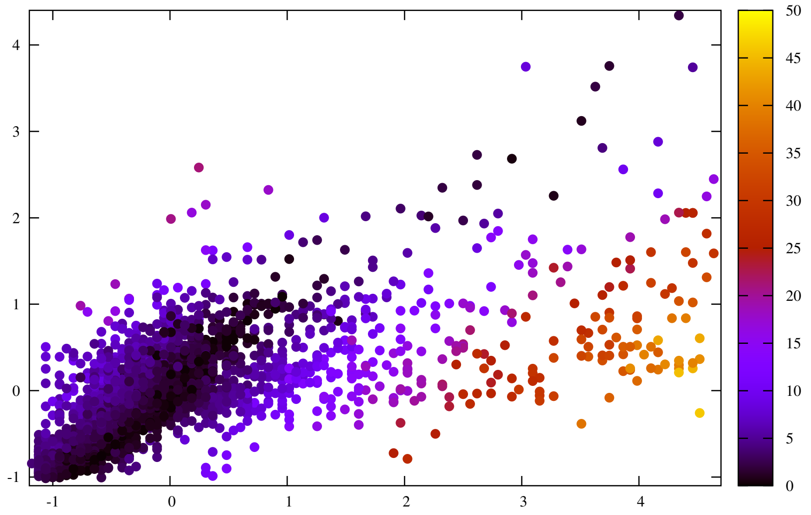
It is fairly obvious that most methods compute a simple local statistic such as the mean deviation, then try to normalize and compare these values on a global level out of necessity. However, they often lack motivation for their choices, and the methods in total offer little beyond the classic work of [Ans95]. The similarities between the methods have not been properly studied before and the users are essentially facing a variety of algorithms without indication of which is the most appropriate for them. The evaluation which just lists a number of example outliers detected is not very helpful here either, especially since many of these outliers may well be

Table 6.4: Locality and Models in Spatial Outlier Detection.

method	context,	model function,	notes
	reference	comparison function,	
		normalization (optional)	
Local Moran [Ans95]	Matrix	$\sum_{\mathcal{N}} \omega_n \pi(p)\pi(n)$	Spatial autocorrelation
	global	with $\sum \pi(p)\pi(n)$	
		z-score	
Local Geary [Ans95]	Matrix	$\sum_{\mathcal{N}} \omega_n (\pi(p) - \pi(n))^2$	Spatial autocorrelation
	global	with $\sum (\pi(p) - \pi(n))^2$	
		z-score	
z-statistic [SLZ03]	kNN	$\pi(p) - E_{\mathcal{N}}[\pi(n)]$	
	global	threshold	
		$\lvert z\text{-score}\rvert$	
iterative r [LCK03]	kNN	$\pi(p)/E_{\mathcal{N}}[\pi(n)]$ or inverse	iteratively updated:
	global	threshold	$\pi(p) = E(\pi(n))$
iterative z [LCK03]	kNN	$\pi(p) - E_{\mathcal{N}}[\pi(n)]$	iteratively updated:
	global	threshold	$\pi(p) = E(\pi(n))$
		$\lvert z\text{-score}\rvert$	
median [LCK03]	kNN	$\pi(p) - \text{med}_{\mathcal{N}}[\pi(n)]$	
	global	abs. descending	
		$\lvert z\text{-score}\rvert$	
weighted z [KLC06]	kNN	$\pi(p) - \sum_{\mathcal{N}} \omega_n \pi(n)$	
	global	abs. descending	
		$\lvert z\text{-score}\rvert$	
avg. diff [KLC06]	kNN	$\sum_{\mathcal{N}} \omega_n \lvert\pi(p) - \pi(n)\rvert$	
	global	descending	
POD [KLD07]	kNN	$\min_{\mathcal{N}}\{\lvert\pi(p) - \pi(n)\rvert\}$	inefficient graph-based alg.
	global	top-k	
		label	
SLOM [CS06]	spatial	trimmed mean distance $\hat{d}$	largest distance ignored
	spatial	$\frac{\hat{d}}{\text{mean}(\hat{d})+1} * \beta$	β oscillation measure
	global	descending	



(a) Z-statistic [Ans95; SLZ03]



(b) Median [LCK03]

Figure 6.4: Moran scatterplots with outlier scores for US Census land use data (basic methods).

global outliers as well (such as Soldier Summit, Utah, which is a ghost town and thus obviously has uncommon rent and population values already on a global level).

6.3.4 Experimental Comparison of Spatial Outlier Scores

In Figure 6.4, we visualize the results for z-statistic-based methods (comparing local and global z-scores) of [Ans95; SLZ03; LCK03] in relation to the raw z-scores in a Moran scatterplot. The x-axis is the global z-score, the y-axis the local z-score. Globally unusual objects are on the left and right side, locally unusual objects (by their z-score) are on the top and bottom. The color is determined by the outlier score. The results of both methods are closely related and are essentially the deviation from the diagonal. The imbalanced nature of the actual data distribution – which apparently is not normal but skewed – seems to affect the algorithm performance. Figure 6.5 contains some of the more advanced methods, including SLOM [SC04; CS06] but also, using our framework, the canonical adaptations of LOF [Bre+00] and LDOF [ZHJ09] to the spatial domain. The results differ much more from the z-score than the median-based method in Figure 6.4b, which produced mostly the same results as the original z-score. SLOM results here show noise in the range of low scores from 0 to 0.2, and no objects are assigned a particularly high score. The adaptations of the traditional outlier detection methods offer a much better contrast and seem to find more interesting outliers, such as the objects around $(0.4, -1)$ that are slightly above the global mean, but around one standard deviation below their neighbors mean. By definition, LOF and LDOF only detect outliers that are towards the bottom right area, but can trivially be adapted to detecting the type of outliers in the upper left or both by using the inverse ratio.

Figure 6.6 compares different algorithms on this data set, assessing the correlation between the resulting outlier scores. The relationship of LOF and LDOF is surprisingly linear, while z-statistic and LOF diverge much more. SLOM interestingly seems to differ mostly from z-statistic for low scores. These probably are the points in the upper left area of the Moran scatterplots that are not outliers by the LOF-style definition used.

In summary, we see here that the implementation of a spatial neighborhood does not make a method local in the strict sense of using locality not only for model building but also for model comparison. Making spatial outlier detection truly local remains as a possible improvement for a broad range of existing methods.

In Figure 6.7, we show some example analysis of the unemployment rates for 2009 in Germany on a county level. The classic z-statistic is not very convincing on this data set. It detects only two strong (both of which are global outliers) and some less strong outliers that cannot be easily explained. SLOM performs better and detects some additional outliers. However, as with z-statistic, only two of them are significant: the harbor city Bremerhafen next to Bremen, and the small city of Pirmasens close to France, where the ongoing demise of the shoe producing industries along with the closing down of a US military base have caused the unemployment rates to skyrocket. The LOF adaptation in our framework performs very well, detecting much more interesting additional outliers (and all of the outliers mentioned above). For example, it detects the city of Munich as an outlier. While the unemployment rate for Munich was just 6% – an excellent value for a big city, the median in this data set is 7% – this is twice as much as that of the surrounding counties. This is a prime example of a local outlier, with a very normal value on the global scale, but a strong divergence compared to the local neighbors. It is this kind of outliers that we expected all the algorithms to discover.

Figure 6.8 gives a detail view of the outliers detected for Bavaria. Most city regions here show up as outliers, which is correct since they usually have a significantly higher unemployment rate than the rural areas in Bavaria (largely because unemployed people tend to move to the cities where much more new job opportunities are created, while people with safe jobs tend to buy houses in the rural areas outside of the cities). But there also is an interesting case in which this rule of thumb does not hold: the city of Ingolstadt is with 4.8% rather close to the Bavarian average of 4.5%, and indeed it does not achieve a high outlier score. The region of Eichstätt – a much larger, rural area not far north of Ingolstadt – however has so called full employment with just 2.2% unemployment rate (and as many open positions as unemployed people). The reason for this excellent score however is located in Ingolstadt: the car manufacturer Audi is doing very well and hiring many people, which can afford to move to nice homes outside of the city in Eichstätt. So while this gives Ingolstadt a score typical for this area of Bavaria, it brings Eichstätt to this unusually low score. Note that none of the other methods managed to rank Eichstätt highly. The observed trend of cities showing up as outliers does not hold for all of Germany. In eastern Germany, Dresden and Chemnitz for example are well-aligned with the surrounding areas at around 12%. Hannover also is with 9.3% just slightly higher than the surroundings at around 8.5%.

Overall, we see again that the concept of locality (that does not necessarily come along with selecting spatial neighbors for model building) can considerably improve the results of outlier detection.

6.3.5 SLOM as a Special Case of Local Outlier

To return to our formal analysis, we also show the dependency graph for the spatial LOF adaptation in Figure 6.9. Compared to Figure 6.2a the only differences are that we are using spatial neighbors as reference set, and we added a normalization step from [Kri+11] using a Gamma normalization. A similar normalization was also added to the comparison method for visualization purposes. This adaptation of the classic LOF method to spatial data is leading to interesting observations on outlier rates, while the raw numbers just depicted the well known difference between Western and Eastern Germany and that the southern states are doing much better economically. The other spatial outlier detection methods primarily produced the counties with the highest unemployment rates, which are global outliers.

To study the impact of the use of locality in the reference set for model comparison, we design two base-line methods: For a first, simple method, we measure the deviation from the mean vector of the neighbors and compare this on a global level. For the advanced method, we add the second type of locality and subtract the mean deviation of the neighbors. The setup is also detailed in Table 6.5 and their dependency graphs are shown in Figure 6.10. As a reference, we compare these two straightforward formulations of the general framework for spatial data with SLOM [SC04; CS06], one of the more renowned approaches in this family which analyzes the spatial neighborhood beyond a simple statistic by computing the so-called oscillating parameter β, which grows when the local distance distribution is skewed and the mean is not central. Figure 6.11 is the dependency graph visualization of this method. It consists of a trimmed-mean average distance (the maximum distance is not included) denoted as $\hat{d}$ and the aforementioned stability parameter β. These functions are given as pseudo codes in the SLOM article, and for

Table 6.5: Spatial Outlier as Special Case of Local Outlier.

method	context	model
	reference	comparison
	normalization	
SLOM [CS06]	spatial	trimmed mean distance $\hat{d}$
	spatial	$\frac{\hat{d}}{\text{mean}(\hat{d})+1} * \beta$
	global	descending
simple spatial	spatial	dev. from mean vec.
	global	descending
	erf	
advanced spatial	spatial	dev. from mean vec.
	spatial	dev. from mean dev.
	erf	

some of the terms in the formulas little reason is given, except, for example, to avoid division by 0 by always adding $+1$ to the divisor.

We compare the results of these baseline methods and SLOM on the US Census ethnic groups data (using 5 attributes), as visualized in Figure 6.12 using a Google Earth overlay. While the strongest outliers are not affected much by using the locality in model comparison, it clearly stabilizes the results in the Mississippi delta area and improves contrast in general. The contrast of outlier scores in our method is also a lot better than in SLOM, where the highest achieved score is just 0.91, and we had to boost the contrast manually for the visualization.

The main outliers detected by all three methods are comparable, and not very surprising. They usually fall into one of three categories: Indian reservations, invalid or incomplete data (there are records with no inhabitants), and small towns. Larger cities are usually not considered outliers. Instead they are split into multiple Census subdivisions that in turn are similar. Only occasionally, one of these subdivisions is recognized as being dissimilar, e.g., when there is a large (and local) Asian community. While the Indian reservations can be seen as global outliers and are recognized easily on a global level, outliers such as the “Space Center CCD” in Florida are not atypical on a global scale. However it is a local spatial outlier, since it contains a wide mixture of ethnic groups while it is located in a predominantly Caucasian area of Florida.

To summarize: we basically reproduce the results of SLOM with a rather simple and efficient setting (advanced method) in terms of the identified outliers and we even improved the stability and contrast of the actual outlier scores. However, the model constructed is a simple two-stage “difference from mean” approach that is very straightforward and easily comprehensible compared to the SLOM pseudo code. Regarding the order of locality, we also see the impact of using second order locality where the advanced method produces much more stable and useful results than the simple (first order) method. As well, we see the impact of third order of locality (the spatial adaptation of LOF) over the second order of locality used in SLOM.

6.3.6 Univariate vs. Multivariate Outlier Analysis

In comparison with recent non-spatial outlier detection algorithms, what is noticeable about spatial outlier detection algorithms is that they are in most cases only able to perform a univariate, not a multivariate statistical analysis of outliers. Most approaches focus on data that consist, aside from the spatial coordinates, of a single non-spatial attribute. For many methods it is obvious or conceded that they are only capable of performing a univariate analysis [e.g. KLD07; SLZ03; LCK03; KLC06]. Furthermore, the univariate analysis used in most of these methods is rather simple and is covered by classic statistics as discussed in Section 1.2. Only some methods [SC04; CS06] are obviously applicable in the multivariate case though they do not care to demonstrate or even to state that they are.

Let us finally point out that the application of our general framework to the task of spatial outlier detection allows to transfer all the achievements of multivariate outlier detection given in traditional outlier detection research since it is easy to apply a traditional model to spatial data as a special case (as we demonstrated) by means of an adapted notion of locality.

6.4 Locality in Video Streams

The multimedia community analyzing video sequences is interested in a lot of different questions like, e.g., key-frame extraction and storyboard display [MA08; KK10], shot or scene change detection [HNB03; Lee+04], or detection of continuity errors [PZ09]. In this application example, we do not aim at covering such a wide range of goals but just to demonstrate the flexibility and usability of the general outlier detection framework to adapt to highly specialized tasks. To this end, we examine video sequences, defining for each video frame the previous 12 frames (0.5 seconds) both as local context and as reference set as you would do in a streaming context. Frame similarity is measured using color histograms in hue, saturation and brightness (HSB) space with quadratic form distance to capture color similarity (texture and edge features are not used for this experiment).

From the local context, the root mean square distance (RMSD) to the previous frames is computed:

Definition 6.4.1 (Root-Mean-Squared-Deviation Model Function)**:**

$$\mathrm{RMSD}_{i,d}(o, c(o), \mathfrak{D}) := \sqrt{\mathrm{mean}_{n \in c(o)}\, d(D_i(o), D_i(n))^2}$$

This model function is meant to capture "image instability" and is visualized in the first row of Figure 6.13.

Our intended video outlier definition is based on a sudden increase of instability. This can now be modeled using two simple additional model functions: the first computes the maximum over the context set, the second one the increase over this maximum. These two can trivially be combined into one model function, we chose however to split them to show reusability of components. Finally, we also add a normalization using erf.

Definition 6.4.2 (Maximum Model Function)**:**

$$\max_i(o, c(o), \mathfrak{D}) := \max_{n \in c(o)} D_i(n)$$

Definition 6.4.3 (Increase Model Function)**:**

$$\mathrm{inc}_{i,j}(o, c(o), \mathfrak{D}) := \max\{0, D_i(n) - D_j(n)\}$$

Definition 6.4.4 (Video Outlier Detection)**:**
Video outlier detection is a generalized outlier detection model for distance function d and neighborhood size k with

$$\mathrm{VideoOutlier}(d, k) = [(\mathrm{RMSD}_{0,d}, \mathrm{prev}_k), (\max_1, \mathrm{prev}_k), (\mathrm{inc}_{1,2}, \emptyset), (\mathrm{erf}_3, \mathrm{global})].$$

The advanced method is adaptive to very different situations. The results were not refined and could further be improved by locating local maxima only. However, this naïve approach was surprisingly effective, in particular since the only parameter is the size of the time window used.

Figure 6.13 consists of three rows. The first row shows the local RMSD, the second row shows the derived outlier scores. The third row contains human-made annotations to the video that indicate single-frame events (crosses) and multi-frame transitions (boxes) with varying severity. The news clip is from a public television news summary, consisting of various short scenes. Most of the annotated events capture switches between different news items, sometimes with transitions or the occasional fast camera pan. Let us discuss some of these events in detail. The region marked as A is actually a still image (a black and white archive picture of the movie director Bernd Eichinger who deceased January 24th 2011). When zoomed in closely, there is a periodic signal to be seen here, which is caused by the keyframes of the video stream compressions. Many of the non-annotated spikes for example at the locations marked with K can be attributed to image quality changes due to keyframes. The first K is in a press conference setting. At the last two marked locations, the keyframe event is clearly less significant than the true event shortly after. We marked some camera pans with a P. While they show a strong variance in the upper plot, they are not recognized as outliers in the second plot. These camera pans start slowly; therefore the increase in variance is only gradual. The three adjacent boxes labeled as B form a complex three-part transition. First the heading is removed; then the scene transition happens; then the new heading is added. Two frames of the main transition are shown in Figure 6.14. C is a typical scene change within the news item, causing an abrupt change with only a “ghost image” in the first frame, a typical example of the top outliers detected. Two consecutive frames are shown in Figure 6.14. A fast camera pan event with light changes can be seen at D. Additionally the high level of detail blurs with the video compression, and again keyframes show up as outliers within this high-activity region. The simpler variance detection shows the high overall variance, the second row shows that the improved method actually recognizes individual key frames. Two frames less than a second apart are shown in Figure 6.14. A true outlier we only annotated in the second pass is found at E. We had first only skipped through the clip to mark scene changes. But when investigating this outlier, there is a clear reason to acknowledge it: the scene is a press conference, and the detected outlier frame is caused by a photographers flash. There is a clear color change, but it is still less severe than in a scene change. Again, two consecutive frames are shown in Figure 6.14.

6.5 Locality in Network Outliers

As another application example, we refer to the ideas and modeling of community outliers as presented by [Gao+10]. For their community outlier detection algorithm (CODA), they used EM to both learn community assignments and outliers in the data set at the same time, whereas we focus on detecting the outliers directly within their neighborhood context. In the data set based on a selection of conferences and journals listed in DBLP[5] data, the goal is to identify outlying conferences and authors.

For the selected conferences and journals, we extracted all publication titles, applied the tokenizer and word stemmer from the text search engine software Xapian[6] and produced term frequency vectors. Orthogonally, we extracted all the participating authors, considering every author a term directly. Since DBLP data are normalized, we do not have to take care of different spellings ourselves. We used the common TF-IDF normalization to weight down common words and ubiquitous authors as well as cosine similarity to compute the similarities.

With the small data selection of 20 conferences (DBLP 20) as described by [Gao+10], we were able to retrieve the same results as CODA: the conferences CVPR and CIKM were identified as the top outliers by a simple LOF-based approach. We use the same components as LOF, but this time only modify the context functions (using r_1 = titles, r_2 = authors):

Definition 6.5.1 (Bipartite Local Outlier Factor)**:**
Bipartite local outlier factor is a generalized outlier detection model for distance function d and neighborhood size k and two relations r_1 and r_2 with

$$\text{BipartiteLOF} = [(\text{maxdist}_{0,d}, r_1 k\text{NN}_{d,k}), (\text{lrd}_{0,1,d}, r_1 k\text{NN}_{d,k}), (\text{mean}_2, r_2 k\text{NN}_{d,k}), (\text{frac}_3, 2, \emptyset)].$$

The same outliers were already analyzed by [Gao+10] as being community-outliers since both conferences bring together different research communities: CVPR draws on computer vision, artificial intelligence, and machine learning, while CIKM attracts data mining, information retrieval, and database people. Unlike the other conferences, it is hard to pin them down to one

[5]Digital Bibliography & Library Project, `http://www.dblp.org/db/`
[6]Xapian search engine library, `http://xapian.org/`, GPL

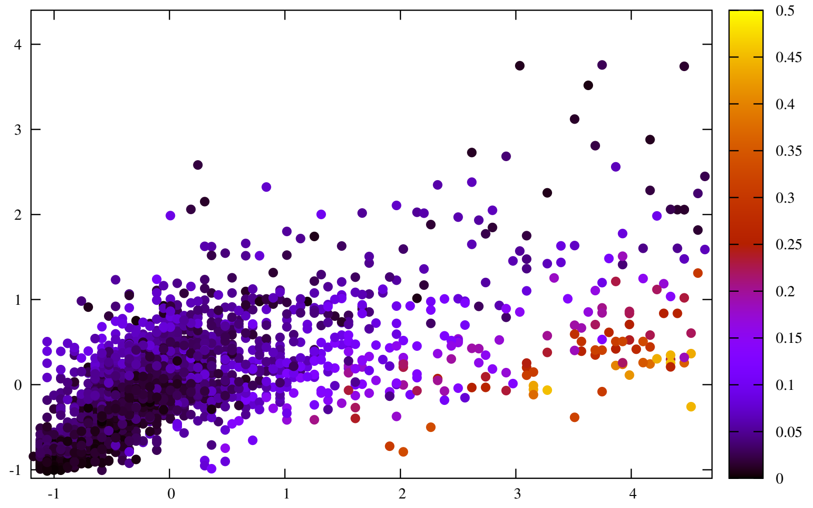
Table 6.6: Top 10 outliers in DBLP 50 data set.

Conference	Score	Notes
EXTREME	0.970	small XML conference
DAGSTUHL	0.968	multi-topic, multi-language
APWEB	0.965	Asia-Pacific community
BTW	0.903	German-language database community
WIDM	0.902	web data mining
SSDBM	0.891	sub-community
JBCB	0.875	bioinformatics journal
TCBB	0.863	bioinformatics journal
SADM	0.843	data mining and statistics community
GEOINF.	0.818	geo-informatics

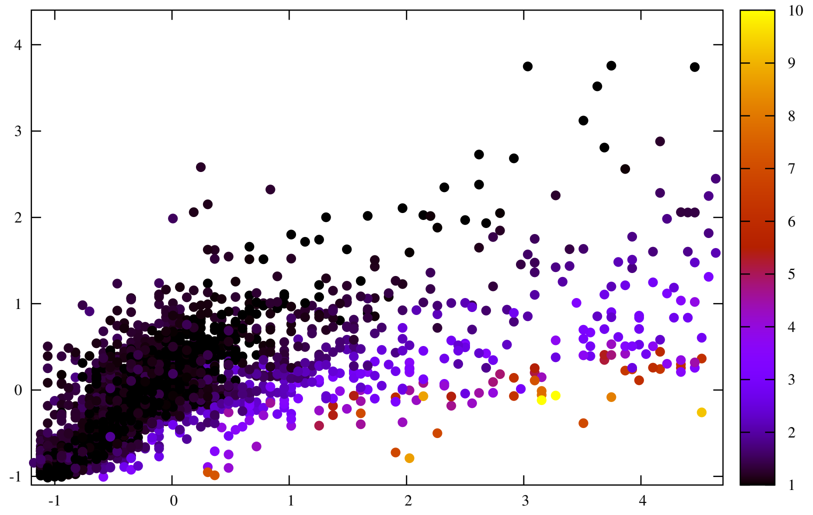
research community or the other. Figure 6.15 and Figure 6.16 give the similarity matrices on authors and titles, respectively. The similarity matrices support the interpretation that CIKM is connected to multiple communities both in titles and authors. CVPR is loosely connected on the titles similarity, but is even a global outlier when it comes to the authors involved.

Interestingly, CODA is not a local approach but learns globally c classes (using EM-clustering), where c is an important parameter. The effect of EM is that certain local structures are learned. This may be the reason why the results of CODA can be also retrieved with an approach that uses locality in the first place.

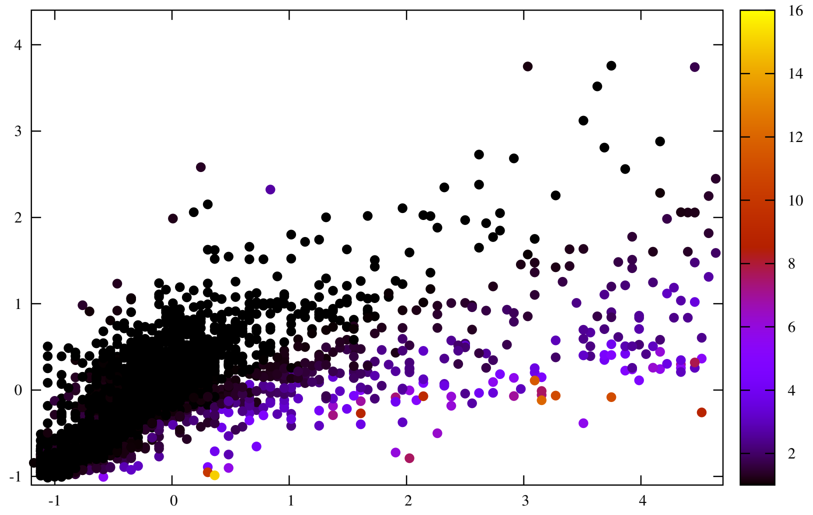
We also performed experiments on a broader data selection using 50 conferences and journals from DBLP (DBLP 50). The corresponding similarity matrices are depicted in Figure 6.17 and 6.18. The difference between both ways to assess the similarity can be highlighted by BTW, a German conference that attracts German database researchers (hence showing a certain similarity level to international database conferences ICDE, SIGMOD, VLDB, EDBT in terms of common authors) but comprises many papers written in German (hence the low overall similarity in terms of title words). The interesting aspect of community outliers as discussed by [Gao+10] is that both similarities are not used in isolation but complementary. Correspondingly, with our settings, BTW is not the utmost outlier although still a prominent one. The top-ten community outliers based on a generalized local outlier approach using building blocks from LoOP [Kri+09a] (largely for the normalization benefits) are listed in Table 6.6. Interestingly, in this larger data set, CIKM is not a prominent outlier any more. Including web-conferences and other special topics like bioinformatics and geo-spatial as well as a broader selection of IR conferences leads to more cross-community-links overall (as, e.g., the prominent outliers DAGSTUHL and SADM).



(a) SLOM [SC04; CS06]

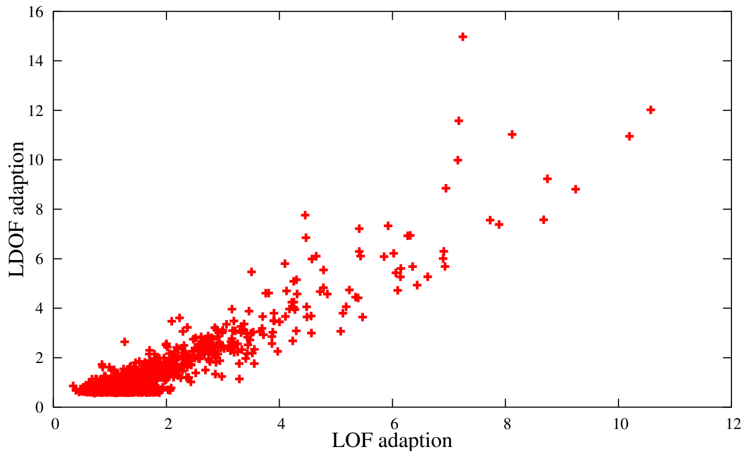


(b) LOF [Bre+00] adaptation

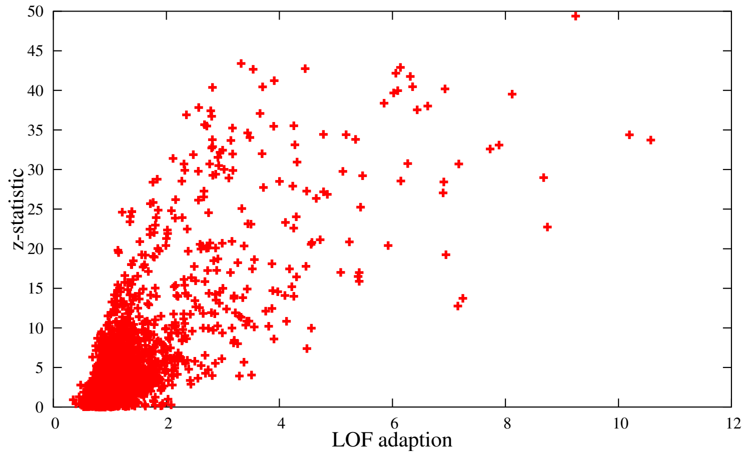


(c) LDOF [ZHJ09] adaptation

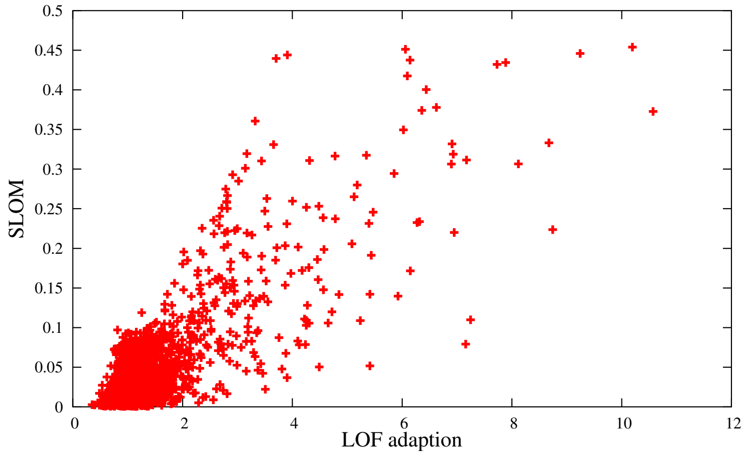
Figure 6.5: Moran scatterplots with outlier scores for US Census land use data (advanced methods).



(a) LOF vs. LDOF

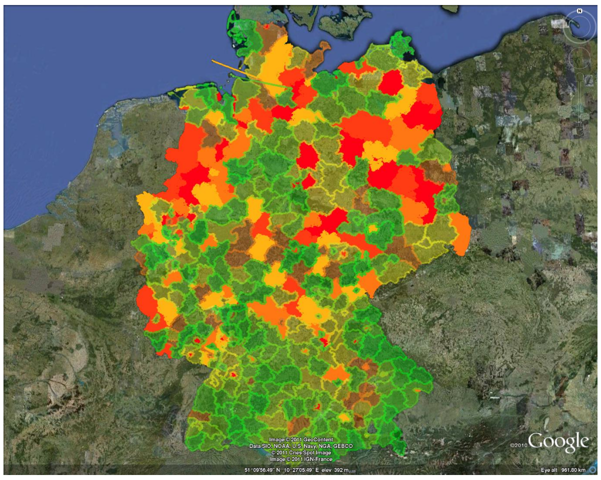


(b) LOF vs. Z-statistic

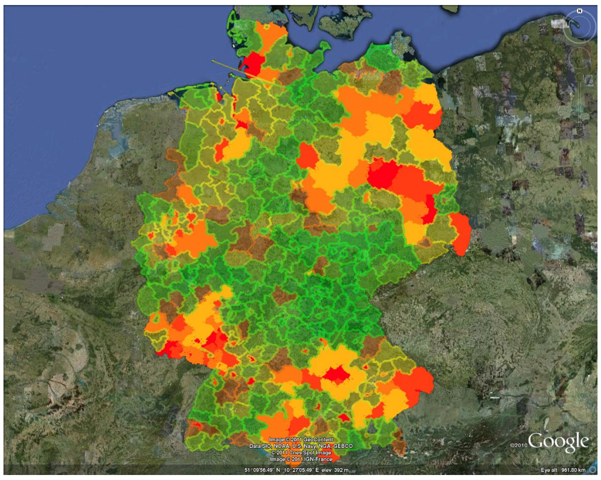


(c) LOF vs. SLOM

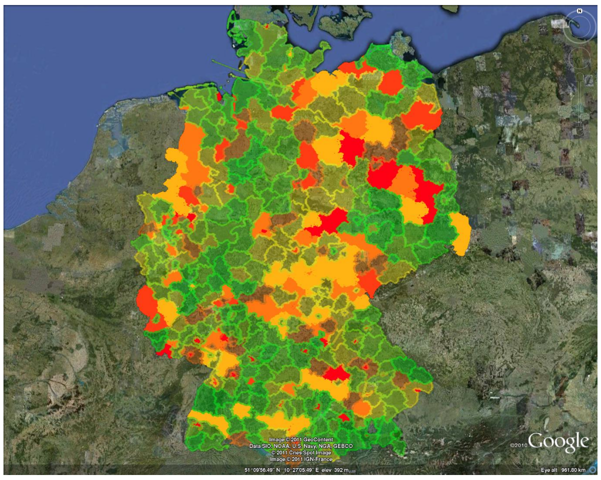
Figure 6.6: Comparing different methods.



(a) z-statistic



(b) SLOM



(c) LOF adaptation

Figure 6.7: Outlier scores in unemployment rates in Germany.
Background © 2012 Cnes/Spot Image, TerraMetrics, GeoContent, Google

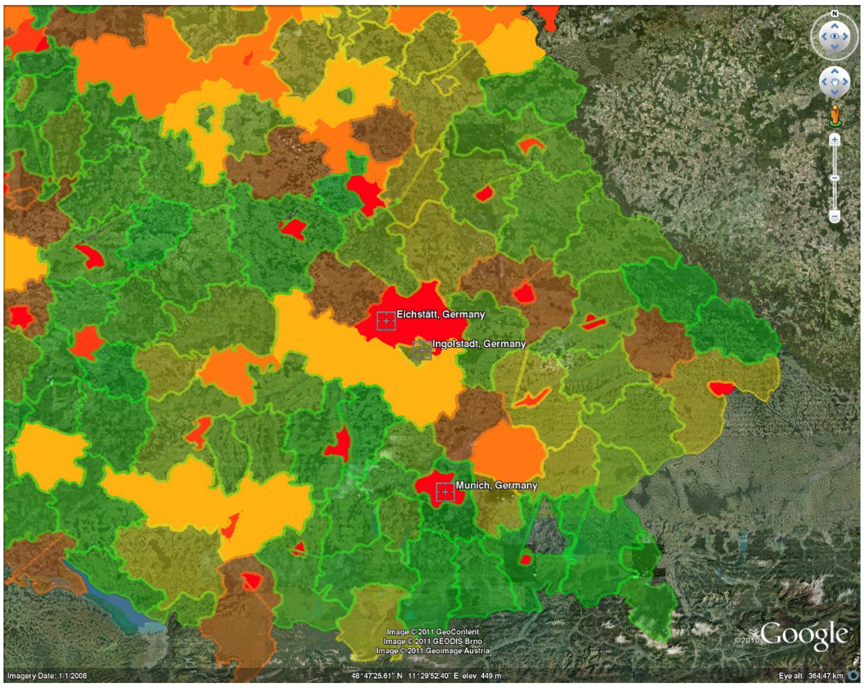


Figure 6.8: Bavaria detail for LOF adaptation.
Background © 2012 GEODIS Brno, GeoContent, TerraMetrics, Geoimage Austria, Google

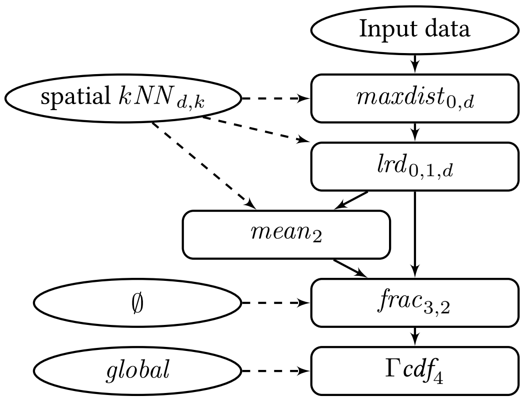
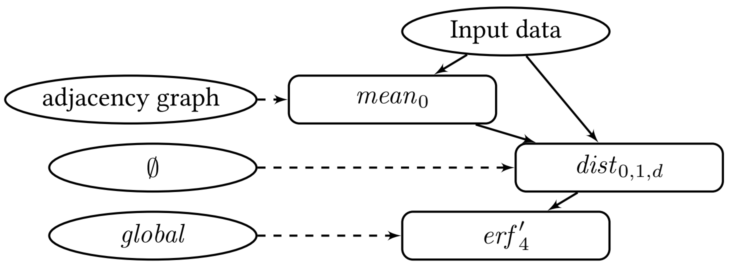
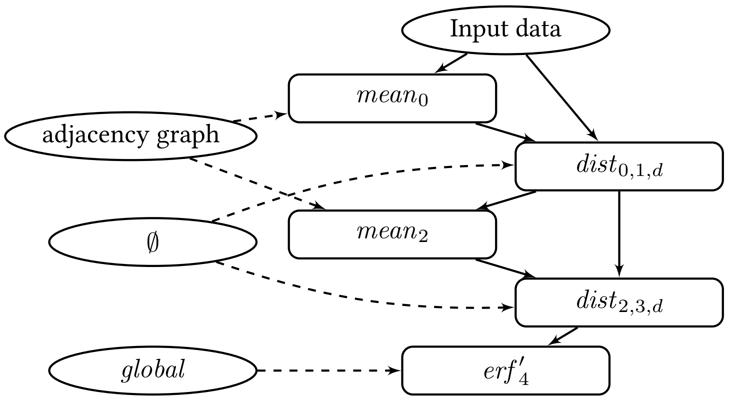


Figure 6.9: Spatial adaptation of LOF, including normalization.



(a) First order deviation outlier



(b) Second order deviation outlier

Figure 6.10: Dependency graphs for baseline spatial outlier detection methods.

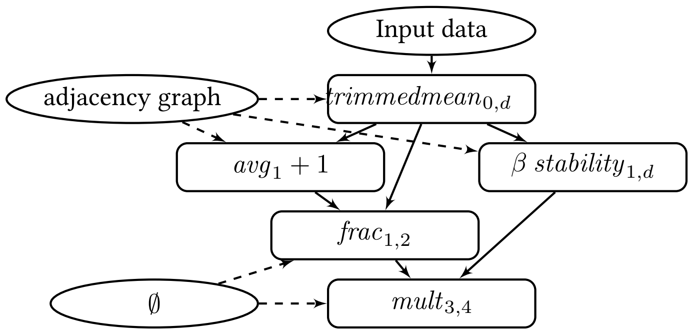
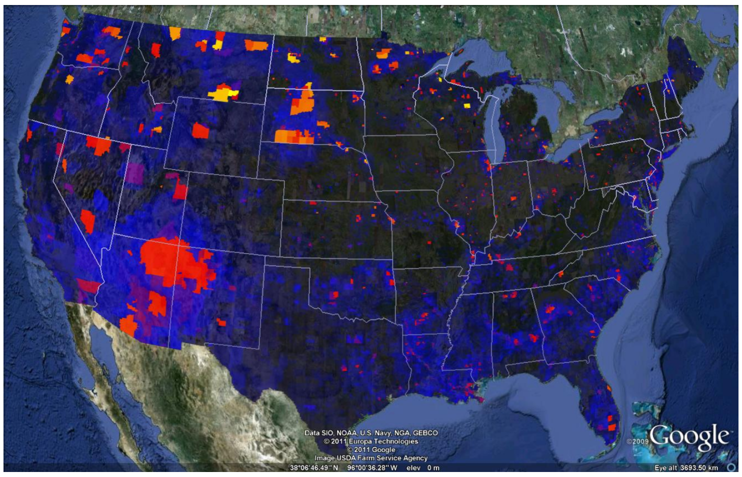
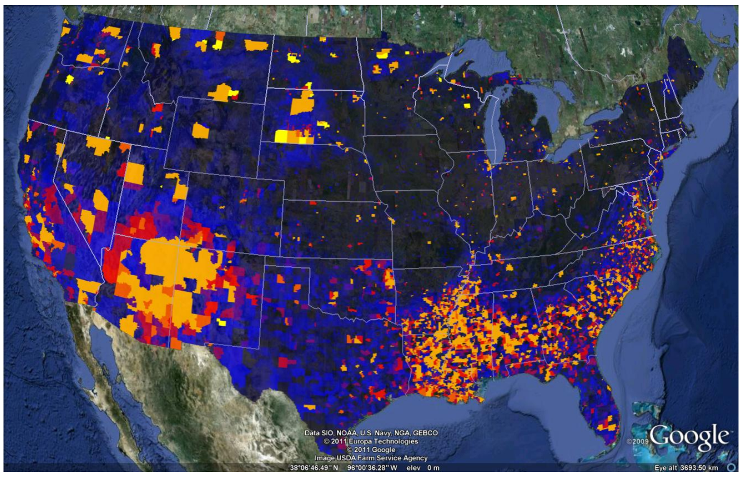


Figure 6.11: Dependency graph for SLOM.


(a) SLOM [SC04; CS06]


(b) Simple method

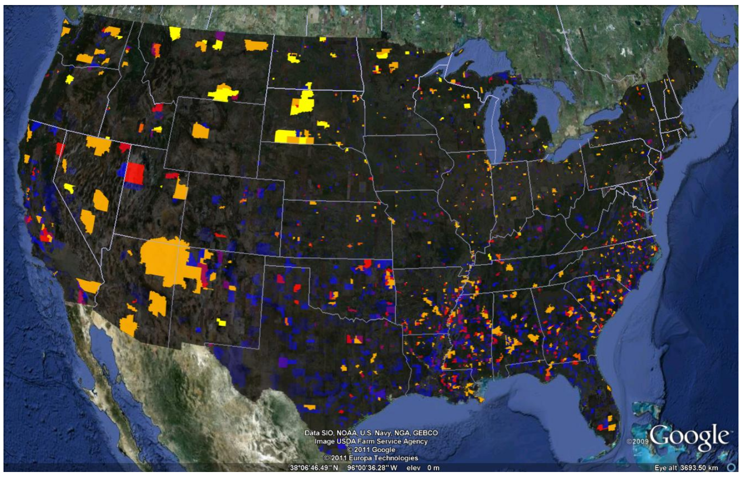

(c) Advanced method

Figure 6.12: Spatial outliers on US Census data on 5-dimensional ethnic groups. Bright colors indicate outliers, dark colors inliers.
Background © 2011 Google, Europa Technologies.

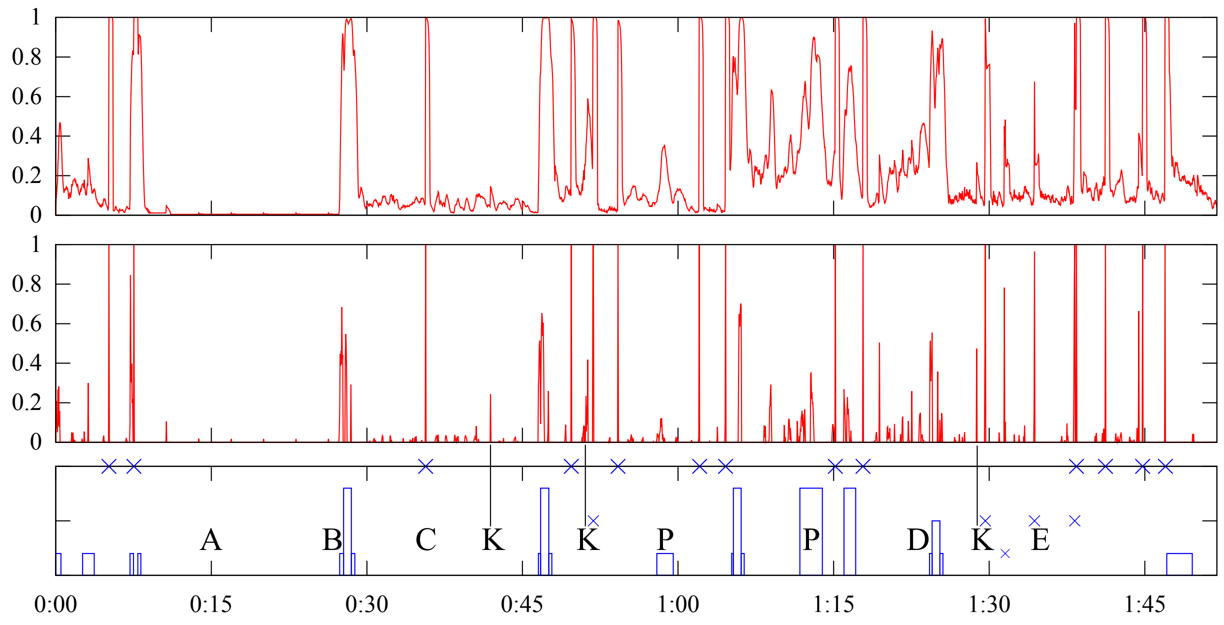


Figure 6.13: Outlier scores in a news clip. The first plot uses the RMS distance directly, the second plot uses the increase in RMS. In the third plot, crosses indicate interesting events, while boxes indicate transitions of varying significance.

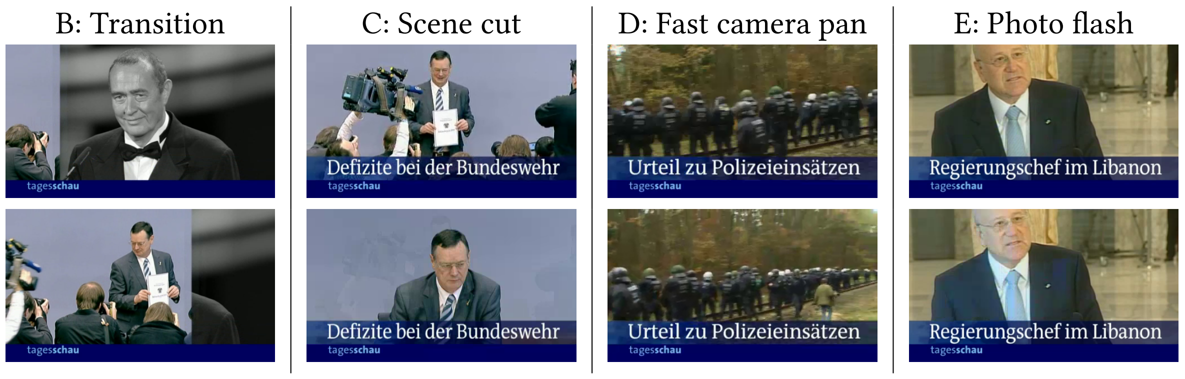

B: Transition
C: Scene cut
D: Fast camera pan
E: Photo flash
Defizite bei der Bundeswehr
Urteil zu Polizeieinsätzen
Regierungschef im Libanon
tagesschau

Figure 6.14: Frames captured from the news video clip at outlier locations B, C, D, and E. © 2011 ARD

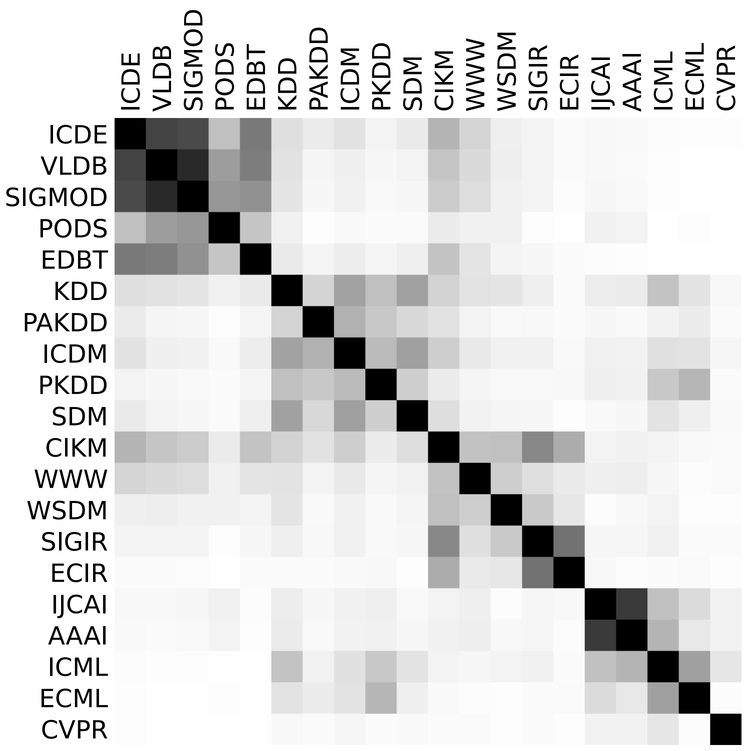


Figure 6.15: Author similarity matrices for DBLP 20 data set.

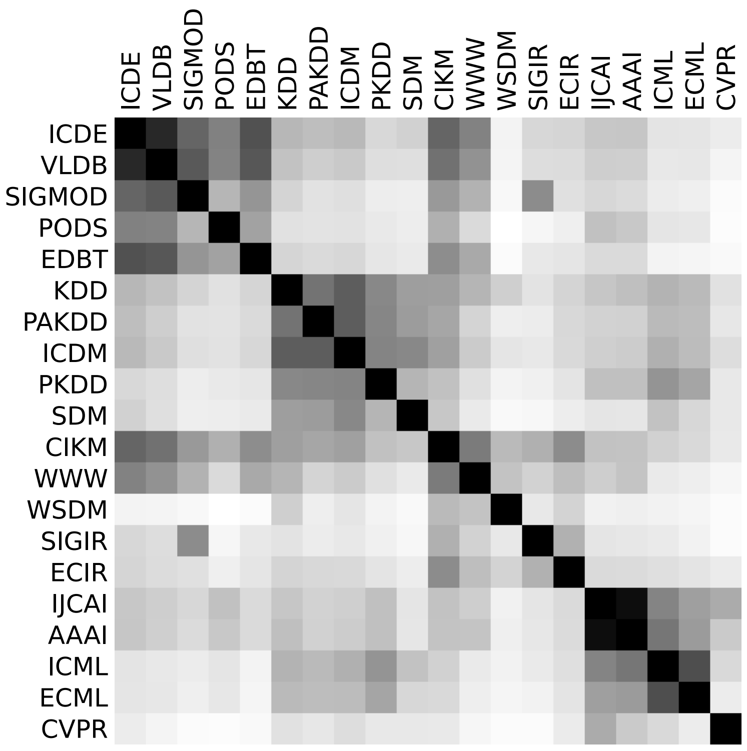


Figure 6.16: Title similarity matrices for DBLP 20 data set.

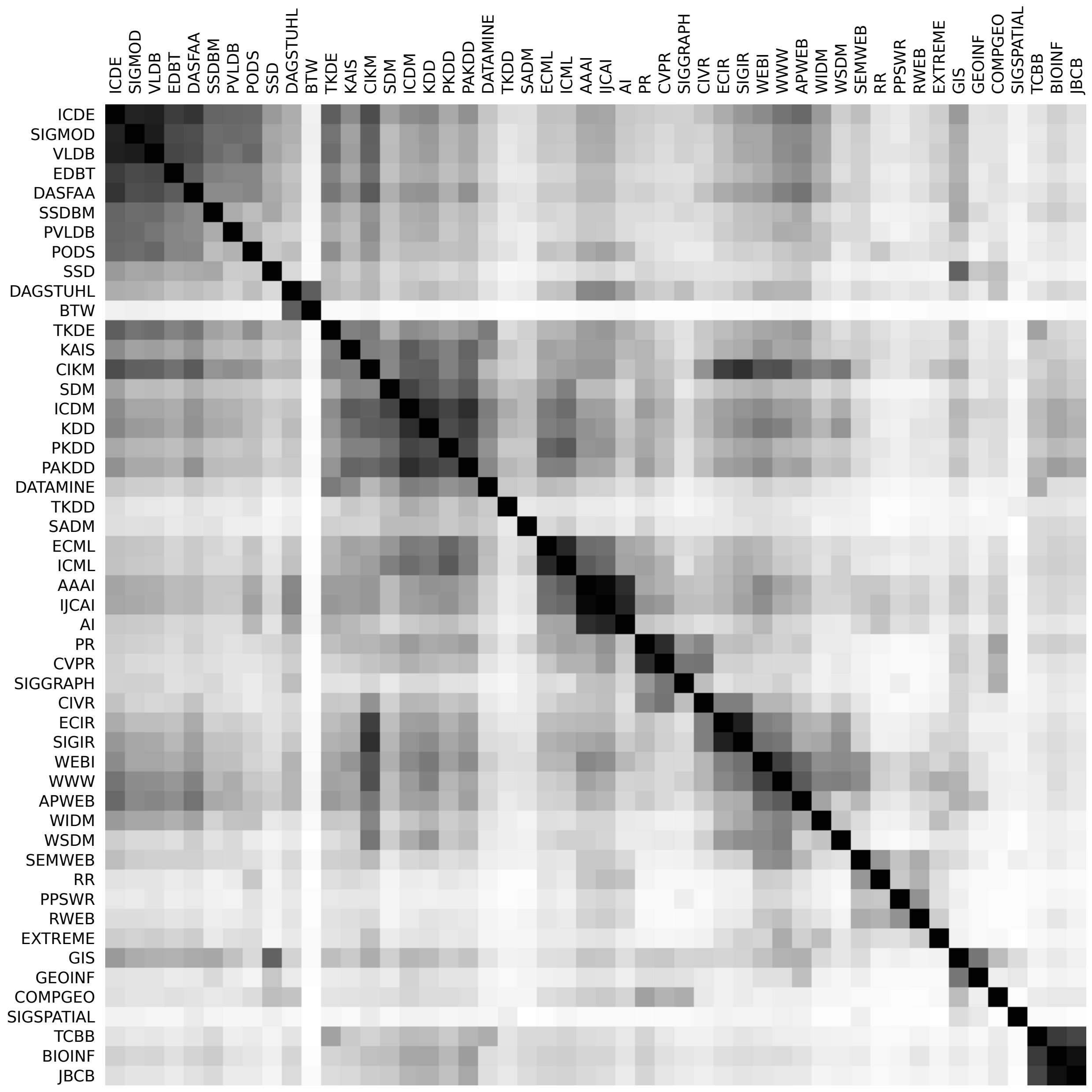


Figure 6.17: Title Similarity for DBLP 50 data set.

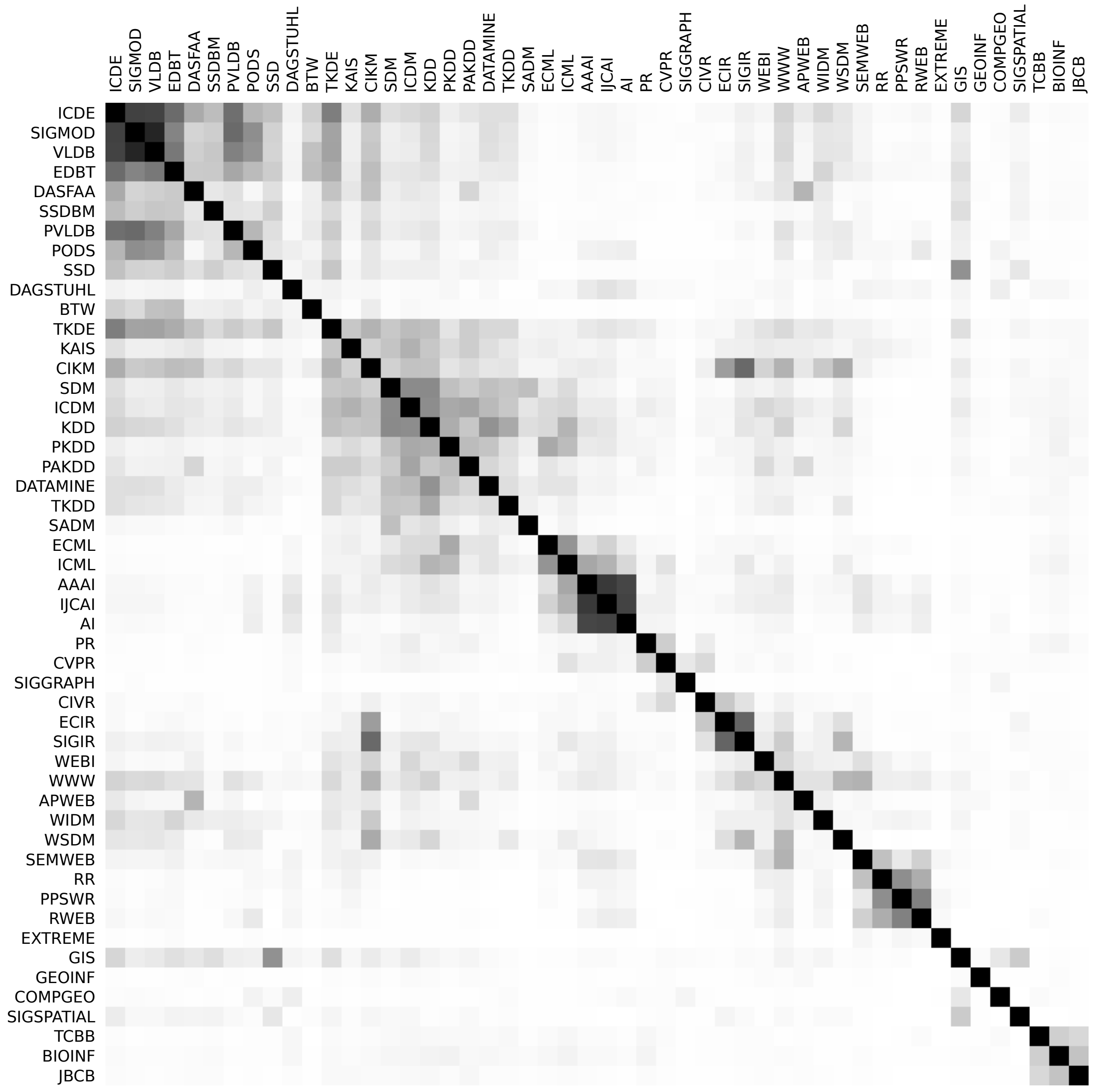


Figure 6.18: Author Similarity for DBLP 50 data set.

7 Ensemble Methods

This section is an enhanced version of the approaches published in:

> H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Interpreting and Unifying Outlier Scores". In: *Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.* 2011, pp. 13–24
>
> E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. "On Evaluation of Outlier Rankings and Outlier Scores". In: *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA.* 2012, pp. 1047–1058

7.1 Background

In classification, building ensembles of single classifiers to gain an improved effectiveness has a rich tradition and a sound theoretical background [Die00; VM02]. Also in clustering, building ensemble clustering methods has found much interest over the years [GA11]. In the area of outlier detection, though much effort has been invested in design and implementation of advanced outlier detection algorithms, only some attempts to design superior combinations can be found in the literature [LK05; GT06; NAG10; Kri+11; Sch+12; Zim+13]. Since then, this domain was called an "emerging area" and "a fruitful research direction for improving the quality of outlier detection algorithms" [Agg12]. Sometimes, ensemble methods are also referred to as meta methods [Agg13], as they are usually built such that they process the output of other methods, and are agnostic to the underlying methods.

Let us reconsider the fundamental lessons learned w.r.t. ensemble methods in classification. The two basic conditions for an ensemble to improve over the contained base-classifiers are that the base classifiers themselves are (i) *accurate* (i.e., at least better than random) and (ii) *diverse* (i.e., making different errors on new instances). These two conditions are necessary and sufficient. If several individual classifiers were not diverse, then all of them will be wrong whenever one of them is wrong. This way, nothing is gained by combining them. On the other hand, if the errors made by the classifiers were uncorrelated, more individual classifiers may be correct while some individual classifiers are wrong. Therefore, a majority vote by an ensemble of these classifiers may be also correct. It is clear that each ensemble member should be at least somehow meaningful in order to get meaningful results out of their combination. Hence, a key for building good ensembles is to use ensemble members that make uncorrelated errors (if any). Feature bagging, a common procedure in ensemble classification or ensemble clustering, was used in [LK05] to induce diversity to ensembles, but the actually achieved decorrelation of detectors has not been evaluated. Overall, this requirement has not yet found theoretical attention in the few attempts to design outlier ensembles, and actually the means to do so have not been around so far. Instead, the four approaches known so far concentrated on methods for meaningfully combining the scores. They addressed the problem that scores delivered by different methods (or in different subspaces, where the scores are usually based on distances) usually vary strongly. For combination of such different scores, [LK05] proposed, first, a normalization by ranking (breadth-first traversal through the outlier rankings to combine), and, second, the cumulative sum of the different scores. The second decisively relies on the comparability of the retrieved scores. To enhance this comparability was the aim of the subsequently proposed approaches. Calibration approaches (sigmoid functions or mixture modeling) to fit outlier scores provided by different detectors into probability values have been used in [GT06]. They induce diversity by using different values for k of the kNN-distance as an outlier score. In [NAG10], the scores provided by a specific algorithm are centered around their mean and scaled by their standard deviation. Thus, all scores are of roughly the same magnitude, even if rather different algorithms are used for combination. Nevertheless, to induce diversity among different detectors, [NAG10] primarily follow the feature bagging approach of [LK05]. Statistical reasoning was used in [Kri+11] to translate scores of different outlier detection methods into sort of outlier probabilities. There, the possibility of enhancement by combining different methods has been demonstrated, yet no measure of actual diversity or correlation between the used base algorithms has been applied. An improved version of this rescaling is introduced in Section 5.3.

7.2 Components of an Ensemble Method

As discussed in [Agg12], some methods can be considered an ensemble approach in the wider sense: LOF [Bre+00] for example suggested to try different values of k and use the maximum observed, and LOCI [Pap+03] suggested flagging outliers if the score reaches a threshold for any radius tested. Last but not least, kNN-Weight [AP02; AP05] can be seen as ensemble of kNN-Outlier detectors. However, this notion of ensembles may be a bit too broad. One may then as well consider every distance-based algorithm to be an ensemble method, since the Manhattan distance is an “ensemble” of one-dimensional absolute differences.

For a method to be titled “meta-algorithm” (as used e.g. in [Agg12; Agg13]), it *should* be agnostic to the underlying input algorithm(s), and not just happen to use some kind of aggregation of multiple values. We propose to only call those algorithms ensemble methods that (i) are compatible with *different* underlying algorithms and consider the normalization required to make the output of different input algorithms comparable, (ii) explicitely consider sources of *diversity*. Advanced ensembles will also (iii) evaluate which ensemble members to keep and which to discard.

7.2.1 Score Normalization

When combining scores in an ensemble, it is essential to first make them comparable. As discussed in Section 5.3, in particular the scores of different algorithms will follow different distributions and scales. And even within a single method, the scales may vary due to e.g. different distance functions and parameters.

In [LK05], the authors experimented with the naïve sum of outlier scores as well as using a variant of the maximum rank ("breadth first"). The reason why the first approach worked reasonably is due to LOF scores having roughly comparable results even in different subspaces (at least when these are normalized the same way, and all other parameters of LOF are also kept fixed). The second approach, returning objects by their maximum rank, discardes the actual scores and this way avoids the problem of score normalization. The probability normalization proposed in [GT06] is actually too aggressive: the contrast of the results is maximized, such that intermediate values in the range $[0.1; 0.9]$ become rare or even nonexistant. On such scores, the proposed multiplication of outlier probabilities then reduces to performing set intersection; the alternate proposal of using the sum of the scores reduces to counting how often an object was nominated. HeDES [NAG10] is the first to combine scores of different algorithms (hence the name "heterogeneous detector ensemble on random subspaces") while paying attention to the problem of scale. To reduce the effects of scale, they standardize each score to zero mean and unit variance (which bears an implicit assumption of the scores being normal distributed, which does not hold in practice, as seen in Section 5.3.1). Additionally, each detector is assigned a weight based on supervised training for computing a weighted sum; in practice data sets will likely be too different for such a training approach to be reliable.

7.2.2 Score Combination

The two most common ways of combining two scores (not including score normalization and transformation) are to use the maximum score (e.g. in LOF and LOCI the authors propose to use the maximum of multiple scores, or denoted as "breadth-first" when using the maximum ranks in [LK05]) or the average score (or, equivalently, the cumulative sum [LK05; NAG10], kNN-Weight [AP02; AP05]). Often, no particular reason is given why one method is preferred over another. We want to propose some additional variations and discuss their prerequisites and theoretical background.

The first group of score combination ("voting") methods are those based on different averages (see Section 2.1 for an overview). This includes the average, but also the minimum, median and maximum are extreme types of means. Averages (including the minimum and maximum) are more robust towards errors of low magnitude, whereas large deviations will have more influence on the outcome.

The second group, on which we will go into more detail, are those motivated from probability theory. Assuming that the scores $S := \{s_i\}$ are the probabilities of the object being an outlier, and assuming independence of the individual methods, we can compute the joint probability that the method is an outlier in at most / at least one of them (see [GT06], "parallel" and "serial" configurations) as:

$$\text{Product}(S) := \prod_i s_i \tag{7.1}$$

$$\text{Inverse-Product}(S) := 1 - \prod_i (1 - s_i). \tag{7.2}$$

Alternatively, we can also employ Bayesian odds (see Equation 5.12):

$$\underbrace{\frac{P(O|S)}{P(I|S)}}_{\text{Posterior odds}} = \underbrace{\frac{P(S|O)}{P(S|I)}}_{\text{Bayes Factor}} \cdot \underbrace{\frac{P(O)}{P(I)}}_{\text{Prior odds}}$$

we can now combine individual Bayes factors obtained (as per Equation 5.13) to obtain a single factor. If we assume independence, this is computed as:

$$\frac{P(S|O)}{P(S|I)} = \prod_i \frac{P(S|O_i)}{P(S|I_i)}$$

if our input data are not Bayes factors, but instead posterior outlier probabilities (as e.g. per Equation 5.15), then we first need to scale them back into posterior odds scale using Equation 5.10 and then remove the (redundant) prior on each:

$$\frac{P(S|O)}{P(S|I)} = \prod_i \left(\frac{p_i}{1 - p_i} \frac{1 - \varphi}{\varphi} \right)$$

resulting in:

$$\text{Posterior-Odds}(S) = \left(\prod_i \frac{p_i}{1 - p_i} \cdot \frac{1 - \varphi}{\varphi} \right) \cdot \frac{\varphi}{1 - \varphi}. \tag{7.3}$$

instead of assuming independence, we may also compute the multiplicative mean of the Bayes Factors instead:

$$\text{Posterior-Odds}(S) = \left(\prod_i \frac{p_i}{1 - p_i} \cdot \frac{1 - \varphi}{\varphi} \right)^{1/|I|} \cdot \frac{\varphi}{1 - \varphi} = \prod_i \frac{p_i}{1 - p_i}. \tag{7.4}$$

where φ is the prior outlier probability. Both the Bayes factor and the posterior odds are not probabilities, but likelihood ratios, which are sometimes considered to be more informative than a p-value [KR95; Goo08]. Similar to the scaling we used in Section 5.3.2.6, we can now map this back to a probability using Equation 5.11:

$$P(O|S) := \frac{\text{Posterior-Odds}(S)}{1 + \text{Posterior-Odds}(S)}.$$

Given the experience from Naïve Bayes classification, it can be desired to clip the values of s_i to a restricted range of e.g. $[0.01; 0.99]$ to avoid single scores of 0 or 1 to determine the result (which would yield a division by 0, if an object is scored 1). Obviously, using a probability-based ensemble voting functions have high requirements on the normalization of the outlier scores. Furthermore, we need to specify the parameter φ which not only influences the score interpretation, but also the results. For increased numerical precision, it may furthermore be desirable to perform all of these computations in logspace.

In Figure 7.1 we visualize the effect of combining two scores using average (Figure 7.1a) and maximum (Figure 7.1b), product (Figure 7.1c) and inverse product (Figure 7.1d), and Bayesian combinators with different values of φ ($\varphi = .5$ in Figure 7.1e, $\varphi = .1$ in Figure 7.1f, and $\varphi = .01$ in Figure 7.1g). We also include the approach using Bayesian multiplicative mean scaling (Equation 7.4) in Figure 7.1h. Contour lines indicate $.2$, $.4$, $.6$ and $.8$ thresholds; red color indicates scores close to 0 while green color indicates combined scores close to 1. For small values of φ (e.g. Figure 7.1g), the Bayesian approach at first seems to work rather bad: the majority of combinations will receive a high combined score. However, it assumes a non-uniform distribution of the scores; the majority of scores are expected to be close to 0. With this prior assumption of having only a few larger scores, it becomes important to retain this contrast, and we actually expect it to work best if the majority of scores is close to 0. This emphasizes that we need to take prior assumptions of the scores into account.

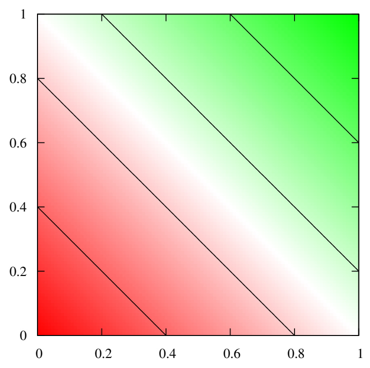
7.2.3 Diversity Sources

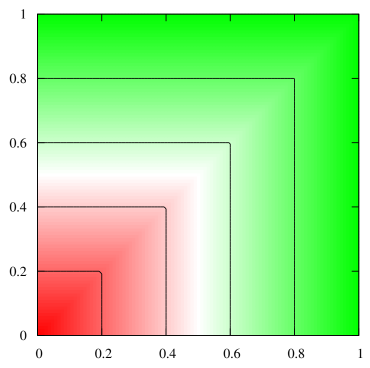
Diversity in an outlier detection ensemble can originate from a number of choices. An obvious – but also the most challenging – choice is to choose different base algorithms. Other sources for diversity are parameterization (in particular parameters such as the number of neighbors and the distance function), data preprocessing and projection (in particular, feature bagging [Bre96], i.e. random selection of subspaces, also used in random forests [Bre01]) or the use of subsampling of the data set [Zim+13].

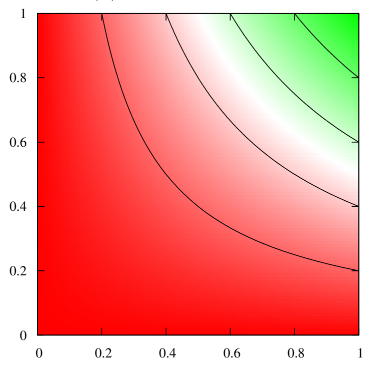
7.2.3.1 Diversity from Algorithms

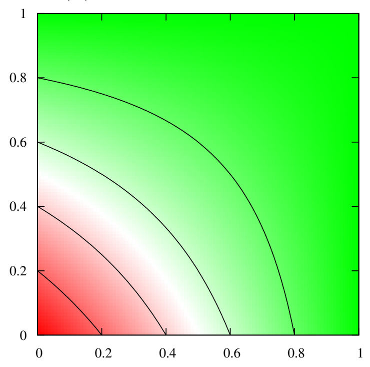
Building an ensemble from different base algorithms appears to be the most appealing choice: different algorithms can use different outlier models, and combining these results into a single outlier detection result should be able to find multiple kinds of outliers.

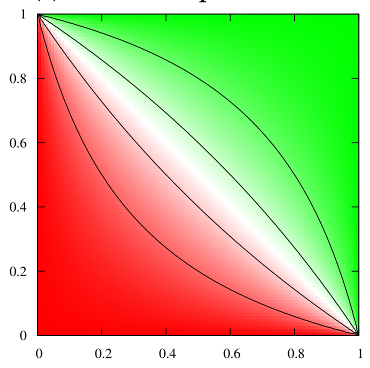
It turns out that it is non-trivial to combine the results of different algorithms, as already discussed in the previous Section 7.2.1. The early score normalization approaches in [GT06; NAG10] are based on weak heuristics such as z-standardization for combining different results. The first successful approach was published in [Sch+12], and the improved score rescaling discussed in Section 5.3 further improves these results.

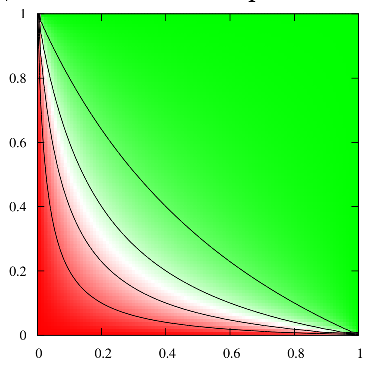

(a) Mean Score

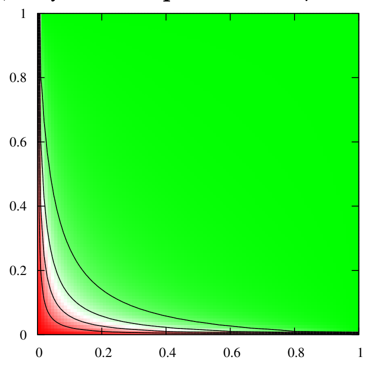

(b) Maximum Score


(c) Product, Equation 7.1


(d) Inverse-Product, Equation 7.2


(e) Bayesian, Equation 7.3, $\varphi = .5$


(f) Bayesian, Equation 7.3, $\varphi = .1$


(g) Bayesian, Equation 7.3, $\varphi = .01$

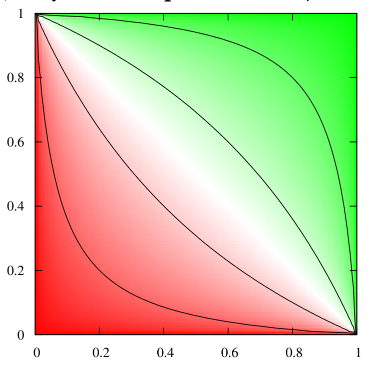

(h) Bayesian Mean, Equation 7.4

Figure 7.1: Visualization of score combination rules

7.2.3.2 Diversity from Parameters

An alternate source of diversity is to vary the parameters of an algorithm. For some parameters, such as k in kNN-based algorithms, one can expect that the score distributions remain similar, and thus simple variance-based rescalings such as HeDES [NAG10] are sufficient. Often, they do not improve much over the baseline results, in particular when the scores follow different distributions: HeDES rescaling can only make variance comparable, but not e.g. skewness.

Varying the distance function will provide some additional diversity; at least when distances based on different paradigms are chosen. Merging these results will require careful score normalization, as the distances returned by different measures will have different distributions. Furthermore (as seen before in Figure 5.21 and repeated in the experimental section in Figure 7.4), many distance functions belong to highly correlated families, and there often exists a strong correlation even between different families. Due to the limited number of distance functions based on different paradigms, we can construct only small ensembles with a high diversity: in our experiments we found that 20 popular distance functions can empirically be grouped into 4-5 families that are highly correlated (see Section 7.4.1.2 and the block structure visible in Figure 7.4). Some approaches that could further increase diversity of distance functions – such as using a randomized weighting – are equivalent to projecting the input data differently. When using binary weights only, we obtain a standard technique to get diversity for ensemble methods known as feature bagging, which we will discuss next.

7.2.3.3 Diversity from Feature Bagging

An easy to use and powerful source of diversity is feature bagging [Bre96]. The benefits of this approach are twofold: leaving away part of the data for each ensemble member yields rather uncorrelated ensemble members (unless the dimensions are highly correlated), but it also helps with some of the problems with mining high-dimensional data (as discussed in Chapter 4): in particular, the distance functions in the lower-dimensional space usually offer better contrast, and ensemble members have a chance of skipping irrelevant dimensions. While some ensemble members may consist mostly of irrelevant attributes, they will likely also not find strong outliers then, and will not find much agreement in the ensemble. Other ensemble members will however consist of mostly relevant attributes and provide very clear results and be more likely to reach consensus. The optimal parameters for feature bagging will obviously depend on the amount of redundant and noisy dimensions in the data set.

It is an interesting observation that *leaving away data* is a good source for diversity in an ensemble, because it contradicts the intuition for building individual methods. However, in an ensemble context, we redundantly use the data multiple times, so usually, no data will remain unused in the end: if we construct an ensemble of 25 detectors, each using $\approx 75\%$ of the data, all data are used 18.75 times on average.

7.2.3.4 Diversity from Subsampling

If we performed “bagging” on the transposed matrix, instead of selecting some dimensions at random, we select some instances of the data; a process known as subsampling and closely related to bootstrap aggregating [Bre96]. The use of subsampling for outlier detection ensemble construction was proposed in [Zim+13] as a source of diversity. Our own experiments with subsampling for outlier detection date back to 2011, and the corresponding code was included in ELKI 0.5.0 [Ach+12] and used by [Zim+13]. However, we then did not use it in an ensemble approach, but only used it as a baseline for evaluation approximate nearest neighbor search. The important key contributions of [Zim+13] are the ensemble use case and an analysis of the theoretical properties. The main concern that stopped us from further pursuing this direction was the observation that this emphasizes parameterization problems. When analyzing the performance of subsampling approaches, algorithm parameters that depend on the data set size should be adjusted to prevent bias from the results coming from parameterization: as we have seen before, most methods are rather sensitive to the choice of the neighborhood size k. But when a method worked best with $k = 5$ on a data set, when subsampling it to $s = 10\%$, what would be the comparable value of $k\prime$ to use on the subsample? Choosing $k\prime = s \cdot k = 0.5$ is obviously not possible. Using the same k on the subsampled data set, however, may yield a result comparable to using k/s (i.e. $k\prime = 50$ in the toy example) instead. In practice, for some methods such as kNN Outlier, we found small values such as $k = 1$ to work best. Since such parameters implicitly depend on the data set size, subsampling the data may have unpredictable influence on the results, as seen in the experiments.

Similar problems can also arise in machine learning, which is why bootstrap aggregation does not subsample the data set, but uses sampling with replacement to produce a data set of the same size as the original data (and approximately the same class distribution). However, for outlier detection this is not beneficial, as we have highly unbalanced classes and need each single object to be evaluated. For small classes, such as outliers, the errors introduced by bootstrap aggregation will be much larger.

Some of the theoretical observations of [Zim+13] (based on earlier considerations on data bubbles in [Bre+01]) may only be useful for large values of k: For small values of k, the deviation of the true kNN distance from the expected value may become too large. For the 1-dimensional uniform distribution $U[0;1]$, the kth smallest value (also known as rank order statistic) is known to be Beta distributed: $\mathcal{U}_{(k)} \sim \mathrm{Beta}(k, n+1-k)$. The mean of this Beta distribution is $\frac{k}{k+n+1-k} = \frac{k}{n+1}$. For the d-dimensional case the mean thus is $\left(\frac{k}{n+1}\right)^{1/p}$.[1]

In [Zim+13] the authors investigate the expected relative error of the distance to the k-nearest neighbor when subsampling. Let $E[d_k, n]$ be the expected distance to the kth nearest neighbor

[1]The formula for the mean as given in [Bre+01; Zim+13] is incorrect, but for large n it makes little difference.

in n objects. Then $E[d_k, sn]/E[d_k, n]$ is the relative change when subsampling sn objects:[2]

$$\frac{E[d_k, sn]}{E[d_k, n]} = \frac{\left(\frac{k}{sn+1}\right)^{1/d}}{\left(\frac{k}{n+1}\right)^{1/d}} = \left(\frac{n+1}{sn+1}\right)^{1/d} \qquad \left(\approx_{n\to\infty} s^{-1/d}\right). \tag{7.5}$$

Roughly said, this proves that the k-distances are expected to grow by a factor of $s^{-1/d}$ when subsampling. Of larger interest, however, is the variance of this term, as the variance translates into the diversity of the resulting ensemble. Rescaling the data set by a constant yields a similar *average* change, but with 0 variance and thus does not introduce diversity at all.

For the 1-dimensional case, the variance of this Beta distribution is known to be

$$\mathrm{Var}[\mathrm{Beta}(k, sn+1-k)] = \frac{k\cdot(sn+1-k)}{(k+sn+1-k)^2(k+sn+1-k+1)} = \frac{k\cdot(sn+1-k)}{(sn+1)^2(sn+2)}.$$

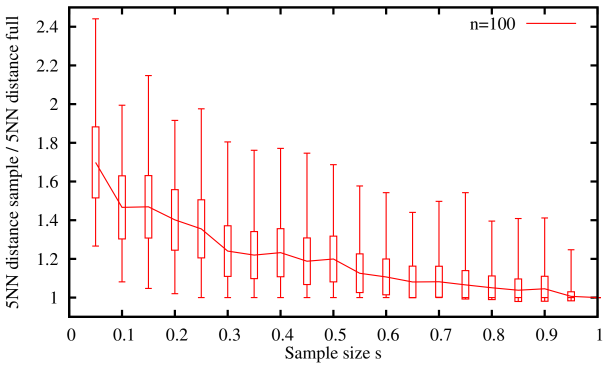
If we now compute the relative variance (which is a squared measure, therefore we need to relate it to the squared estimate), we obtain (for 1-dimensional data)

$$\begin{aligned}\frac{\mathrm{Var}[\mathrm{Beta}(k, sn+1-k)]}{E[d_k, n]^2} &= \frac{k\cdot(sn+1-k)}{(sn+1)^2(sn+2)} / \left(\frac{k}{n+1}\right)^2 \\ &= \frac{(sn+1-k)}{(sn+1)^2(sn+2)}\cdot\frac{(n+1)^2}{k} \\ &\approx_{n\to\infty} \frac{sn^3}{s^3n^3k} = \frac{1}{s^2k},\end{aligned} \tag{7.6}$$

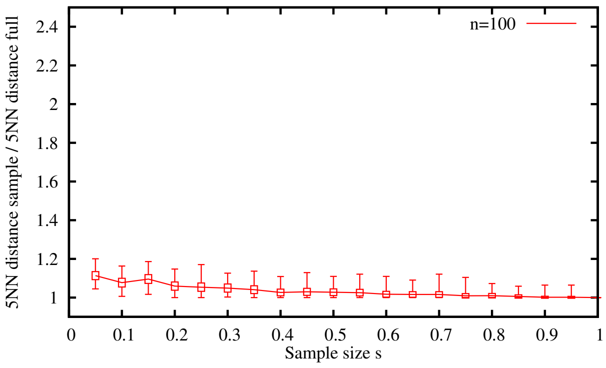
which translates to a standard deviation of $1/s\sqrt{k}$ for large n and 1-dimensional data. Unfortunately, this equation does not as easily translate to multi-dimensional data as the mean. Assuming a sampling rate of $s = 0.1$ and 1-dimensional data, the kNN distance is thus expected to increase by a factor of 10. For $n = 100$, the expected standard deviation is approximately 2.90, for $n = 1000$ it is ≈ 4.30 with a limit for $n \to \infty$ of $10/\sqrt{k} \approx 4.47$. The high variance of this distribution prevents the subsampling approach from working well when used in a single-run approach, for small sample sizes $s\cdot n$ combined with small values of k. On the other hand, however, this variance is also beneficial to the ensemble use case discussed in [Zim+13]. The errors introduced by different subsamples will be largely independent; but a low variance would yield highly similar results; a high variance is likely a prerequisite for obtaining diversity. This may explain why sampling rates of 10% were more beneficial in their experiments than higher sampling rates: doubling the sampling rate halves the variance.

Note that Equation 7.6 is not for the multi-dimensional case; unfortunately we cannot provide a closed formula for the d dimensional case. Experiments indicate that the formula will have a power term _$^{1/d}$ similar to the mean, meaning that subsampling will be much less effective for introducing diversity in high-dimensional data, due to the concentration of distances. Furthermore, the formal analysis assumes drawing new samples from the same uniform distribution

[2]This is a simpler, but equivalent, formulation to Equation 6 in [Zim+13].



(a) Dimensionality $d = 5$



(b) Dimensionality $d = 27$

Figure 7.2: Relative error of 5-nearest-neighbor distance, $n = 100$, $s = .05 \ldots 1.0$.

instead of actually sampling the existing data (this is the reason why the expected variance for $s = 1$ and $k = 1$ is not 0). Also in practice, sampling will not affect every object with exactly the factor s. Only on average, $s \cdot n$ of the original nearest neighbors will remain in the sample. For small k, this will also contribute to diversity. Nevertheless, these equations show the general behaviour: diversity increases with lower sampling rates, but less so with high dimensional data. Figure 7.2a visualizes the quotient Equation 7.5 for $k = 5$ when sampling from $n = 100$ instances distributed uniformly in a 5-dimensional unit ball. We performed 1000 iterations and visualize the mean, standard deviation, minimum and maximum using whisker plots (compare to Figure 1 in [Zim+13]). On 27-dimensional data (we chose 27, because the data set we will be using in the experimental section has 27 dimensions), the variance obtained this way is substantially lower, as seen in Figure 7.2b.

In conclusion, there are multiple (overlapping and sometimes conflicting) effects happening when using subsampling that make evaluation difficult. On one hand, the performance will (most obviously for kNN-Outlier) be more similar to that of $k = k\prime/s$, which can be beneficial or problematic. On one hand, it improves scalability for large data sets, but on the other hand, for data sets where kNN-Outlier with $k = 1$ works best, subsampling will usually deteriorate. In our experiments, LOF usually worked best with $k \approx 10$, which explains why in [Zim+13] they had best results with a very low value of $k = 2$. The other effect is that of variance, which allows ensemble approaches to work.

7.2.4 Ensemble Construction and Pruning

Most published ensemble approaches do not consider pruning or weighting ensemble members. HeDES proposes to assign each detector a weight based on supervised evaluation on training data sets. However, we found that the performance of most algorithms varies too much with parameterization and data characteristics to allow for estimating such weights in a reliable manner from other data sets. Furthermore, the HeDES approach would still only weight or prune ensemble members (by assigning a weight of 0) on a global level, but it cannot make this decision for each individual data set.

In [Sch+12] we suggested a greedy approach based on the similarity measure discussed for evaluation in Section 5.4. Using this similarity measure, however, requires some heuristic choices with respect to the weights: outlier detection is an imbalanced problem, and as such requires weighting. In the experiments, we obtained reasonable results with using the union of all top k ranked objects as initial outlier set: even if we have some inliers in this set, these inliers will likely be the borderline objects, and it is not immediately harmful if these objects are also given extra weight by the similarity measure.

Experiments showed that unpruned ensembles based on a large number of detectors yield only performance inbetween of the average and the best ensemble member. For this reason, we should not use all outlier detectors, but we need to find a way to remove those algorithm results that are too highly correlated with each other, and keep those that might introduce a different point of view (and thus may detect different outliers and improve the result).

7.3 Greedy Ensemble Construction

The key idea of the greedy ensemble is to maximize diversity while keeping the ensemble small. Starting from an initial estimation, ensemble members are processed by decreasing similarity (i.e. diversity) to the already chosen methods; methods that appear to improve the result are added to the ensemble, while methods that merely cause a loss of decisiveness of the ensemble are dropped. A greedy approach (decisions are final, and will never be revised) will keep the complexity low.[3] In practice, we found this approach to yield small ensembles that perform reasonably well. We experimented with other heuristics to update the weight- and target vectors, but the simple greedy approach worked best; but likely future research will yield better ensemble construction approaches.

Based on similarity analysis of outlier score rankings, we propose an unsupervised greedy ensemble construction approach, optimizing diversity: For a set I of individual outlier detectors, n the data set size and the user-specified parameter φ (the expected / desired outlier rate), select the union K of the top k data points of each instance in I, resulting in $\varphi n \leq |K|$ different data points. Under the preliminary assumption that these were the true outliers of the data set, we build a target vector t that has a 1 when the object is in these K objects and 0 otherwise. We also compute the weight vector ω for our distance measure based on this working assumption using $\frac{1}{2|K|}$ and $\frac{1}{2(n-|K|)}$ as weights.

We initialize the ensemble E with the detector $i \in I$ that has the highest similarity to the target vector t. Then we sort all remaining detectors $I \setminus i$ by the lowest similarity to the prediction result of the current ensemble E, because we want to maximize diversity. For each candidate i

[3]Usually, computing the individual algorithms will be much more expensive due to the cost of computing the nearest neighbors, as seen in Chapter 8.

Algorithm 2: Greedy Ensemble Construction Algorithm.

```
Data: $I$ individual outlier detection results
Data: $n$ data set size
Data: $\varphi$ expected / desired outlier rate
/* Compute preliminary outlier set:                                  */
$K := \bigcup_{i \in I}$ top-$k(i)$ with $k$ minimal such that $|K| \geq \varphi n$;
/* Compute target and weight vectors:                                */
$t := \{t_x := 1$ iff $x \in K\}$;
$\omega := \{\omega_x := \frac{1}{2|K|}$ iff $x \in K$ otherwise $\frac{1}{2(n-|K|)}\}$;
/* Initial ensemble - choose most similar result:                    */
$E := \emptyset$;
for $i \in I$ do
    if $dist_\omega(i, t) < dist_\omega(E, t)$ then $E := \{i\}$ ;
end
$I := I \setminus E$;
/* Greedy ensemble construction:                                     */
$p_E :=$ current prediction of $E$;
sort $I$ by decreasing diversity to $p$;
while $I \neq \emptyset$ do
    $i :=$ remove first from $I$;
    $p_i :=$ prediction of $E \cup \{i\}$;
    if $dist_\omega(p_i, t) < dist_\omega(p_E, t)$ then
        $E := E \cup \{i\}$;
        $p_E := p_i$;
        /* Optional: update $t$, $K$                                 */
        sort $I$ by decreasing diversity to $p$;
    end
end
return $E$
```

in this list, we test if the ensemble improves when the new detector is added to it, i.e. whether the similarity of the ensemble with the target vector increases. If so, we update our ensemble with the additional method and reorder the list, otherwise we discard the candidate. At each iteration, one method gets either included or discarded, and these decisions are not revised, making this a greedy algorithm. In Algorithm 2 we give a pseudocode for the method.

The motivation is that we have two factors to decide by: diversity and increase of similarity to the target vector (i.e., improved accuracy). An increase in similarity to the preliminary target vector is the better hard factor – we do not want to include detectors that decrease accuracy – while diversity is more useful as a preference criterion. Also note that this process is entirely unsupervised (except for the initial choice of available detectors, obviously): The ensemble is greedily built from unsupervised detectors by just estimating their performance. Though this is a simple approach to merely demonstrate the applicability of these similarity measures in ensemble construction, the results are convincing and very competitive. Note that this greedy strategy is not possible with a classic *precision*$@k$ or ROC evaluation, that cannot compare two methods for their diversity.

Clearly, using results with uncorrelated errors for combination can only be the second criterion since a certain level of accuracy in the ensemble members is essential – and the better the accuracy, the stronger the overall correlation. These two, however, are not as tightly connected as in classification or clustering, since we do not assess the correlation w.r.t. binary decisions but w.r.t. the resulting rankings, where quite different rankings can relate to equal performance in binary decision. This is exactly the gap in information that is not considered when using ROC analysis only. We demonstrate in the experiments how using this second criterion can improve the overall performance of an ensemble considerably.

7.4 Experiments

In the experiments, we use a squared Euclidean weighted Pearson correlation distance as defined in Section 1 for assessment of the similarity of outlier scores. The reason for this is that we do not normalize the combined predictions of the ensembles; and as noted in Section 5.4, the weighted Pearson correlation measure works better when scores have not been normalized back to $[0;1]$ range.

To numerically measure the gain in outlier detection performance of a combination (ensemble) of outlier detectors as compared to their individual performance, we use the relative improvement towards the target AUC score of 1 over the best of the detectors considered:

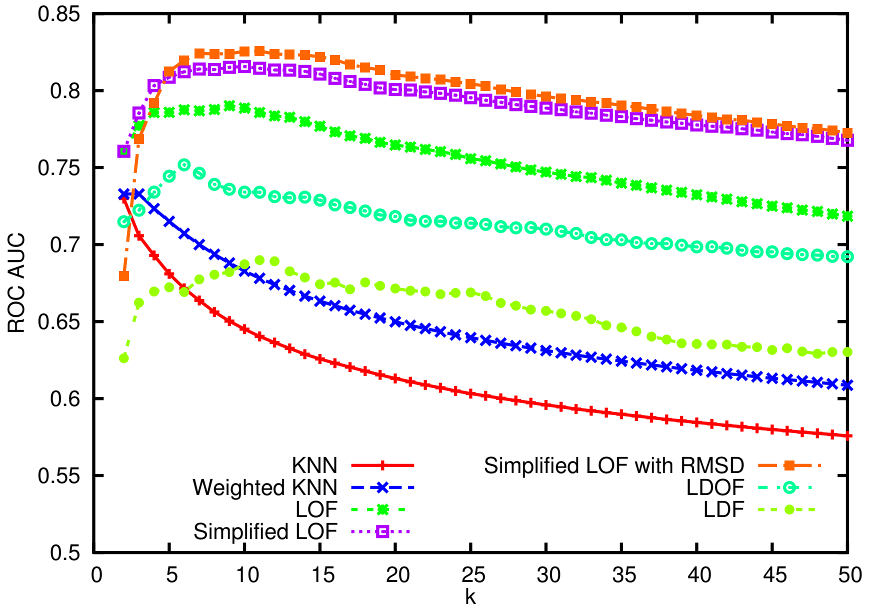
$$\text{gain}(M_1, M_2) := 1 - \frac{1 - AUC(\text{Combination of all } M_i)}{1 - \max_i \left(AUC(M_i)\right)} \tag{7.7}$$

A gain of 0 means that the method performs as good as the better of the input methods. An improvement over this will yield a positive gain, with 1 meaning an improvement to the perfect result. Note that the gain function is relative to the input methods and should merely be read as an indicator of which combinations are beneficial or detrimental. We will be using green and red colors to indicate this trend. For obvious reasons, improving over an already good result is harder than improving over an average outcome, and the largest gain does therefore not necessarily indicate the best result.

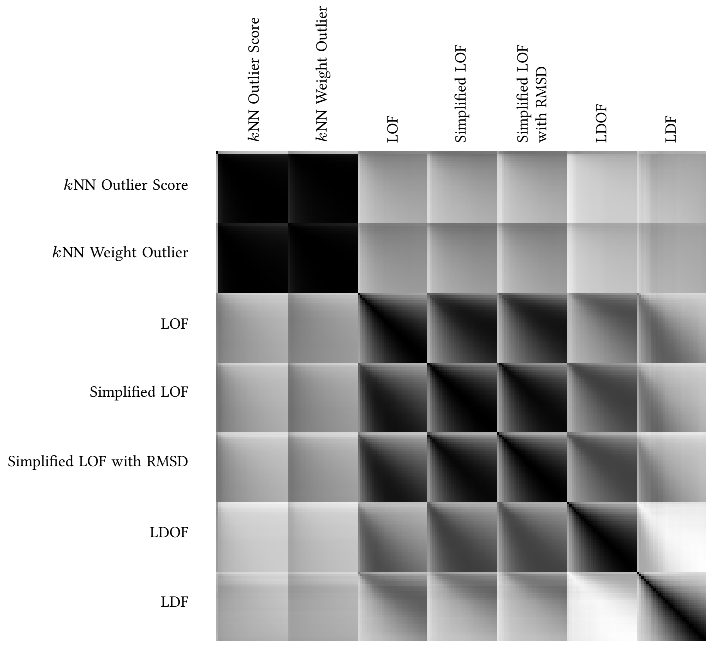
7.4.1 Combinations of Different Algorithms and Parameters

7.4.1.1 Varying the Neighborhood Size k

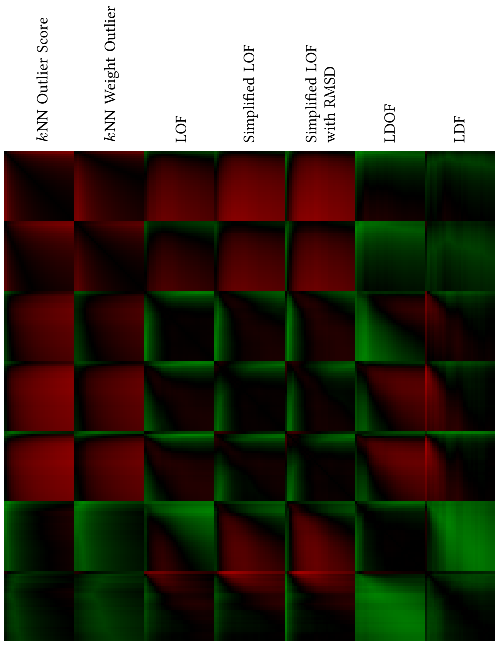
In Figure 7.3, we inspect the relationship between accuracy of the algorithms (Figure 7.3a), similarity of their results (Figure 7.3b), and accuracy gain in combining two algorithms (Fig-



(a) Accuracy (ROC AUC) with increasing k



(b) Similarity



(c) Gain using Bayesian scaling

Figure 7.3: ALOI: Similarity of methods and gain in combining pairs.

ure 7.3c), here on the ALOI data using L_1 distance and varying k for all algorithms from 2 to 20 (Figure 7.3a: $k = 2 \ldots 50$). We see essentially that combining correlated algorithms does not yield a better result than either of them (black) or, depending of the individual perfor-

Table 7.1: Ensemble performance with varying k (LOF, L_1 metric)

Scaling	Combination	Pruning	Reference	ROC AUC	Gain/naïve	Gain/best
Mixture Model	Parallel		[GT06]	(.2057)	(unstable)	
Mixture Model	Serial		[GT06]	.4994	-1.509	-1.396
Average ensemble member				.7718	-0.144	-0.092
Rank	Max			.7871	-0.067	-0.019
Identity	Max		[Bre+00]	.7895	-0.055	-0.008
Best individual method ($k = 9$)				.7911	-0.047	-
z-score	Mean		[NAG10]	.7982	-0.012	+0.034
Sigmoid	Parallel		[GT06]	.8003	-0.001	+0.044
Sigmoid	Serial		[GT06]	.8005	0.000	+0.045
Identity	Mean			.8005	-	+0.045
CDF+COP	Mult.		[Kri+12]	.8058	+0.027	+0.070
CDF	Bayes	Greedy	(new)	.8121	+0.058	+0.101

mance, even deteriorate (red). The kNN-Outlier and kNN-Weight models perform well with rather small k. In this region, combination with, e.g., LOF [Bre+00] shows good improvements (green). While the accuracy of kNN-Outlier [RRS00] and kNN-Weight [AP02; AP05] is decreasing with bigger values for k, also the combination with other methods does usually deteriorate the ensemble performance (red). An interesting exception are LDOF [ZHJ09] and LDF [LLP07], which apparently are different enough from the kNN methods to yield an improvement. Here, the combination of two not extremely well performing methods results in an intriguing gain of performance (albeit at a low level).

The best combination results are obtained when one algorithm is given a small value of k, while the other is given a large value, which indicates that a high diversity is beneficial.

Table 7.1 gives the results for different ensembles built using LOF, L_1 metric and $k = 2 \dots 30$. The mixture model fitting did not work well with LOF scores, and the results are not meaningful. All other methods in this setting improved over the average ensemble member. Taking the maximum LOF value, as suggested in the original LOF publication, performs almost as good as choosing the optimal $k = 9$. However, the average LOF score resulted in even slightly better results. Neither rescaling using sigmoids as suggested in [GT06] nor accounting for variance by using the z-score as suggested in [NAG10] led to a substantial improvement. The cdf based rescaling with COP score transformation improved the results, but not as much as the new method which is also cdf based, but also uses Bayesian score combination and ensemble pruning. The pruned ensemble consisted of 5 members: $k = \{10, 2, 4, 9, 18\}$ (chosen in this order). This again is a very good result. $k = 9$ and $k = 10$ are the two best individual results; and as we have seen before it increases diversity if we combine this with substantially lower or higher scores. In the low score range ($k = 2$ vs. $k = 4$) the results differ somewhat, whereas the results of $k = 18$ and even $k = 30$ do not differ much anymore.

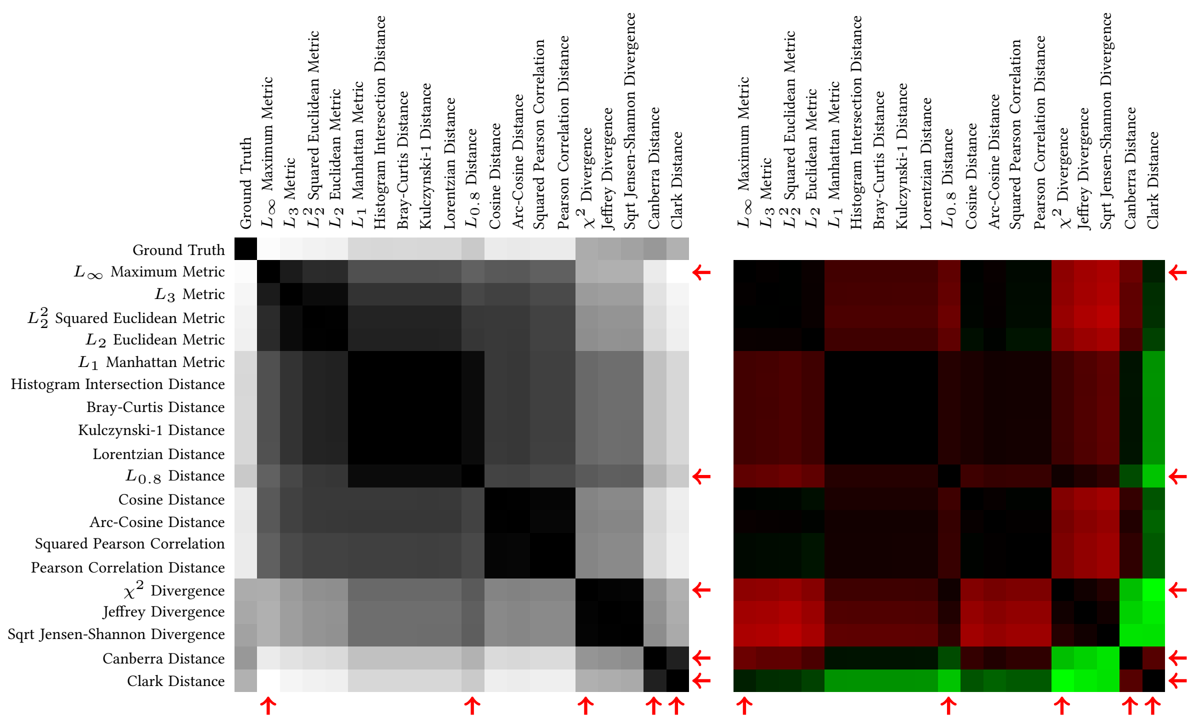


Figure 7.4: Similarity and gain of LOF ($k = 10$) results based on different distance functions. Red arrows indicate ensemble members of Greedy ensemble.

7.4.1.2 Varying the Distance Function

In Figure 7.4 we compare results of running LOF with $k = 10$ and different distance functions. On the left you can see that many distance functions yield highly similar results. Unsurprisingly, the majority of the combinations of two such results are not beneficial (the red color indicates the combination performs worse than the better individual method). However, the combination of Canberra distance together with one of the divergence based measures yields a substantial improvement; despite the two individual methods already performing rather well.

Nevertheless, the overall result is as expected: varying the distance function parameter does not yield a diversity that is very useful for ensemble construction: the individual results are too highly correlated.

Table 7.2 gives the results when constructing an ensemble based on different distance functions. Mixture model fitting [GT06] was too unstable to produce meaningful results. The sigmoid based approach from the same publication worked much better and outperformed both the Manhattan (L_1) distance result (which would be a reasonable default on this histogram data set due to the correspondence to histogram intersection distance) and the average ensemble member performance. This performance was on the same level as taking the naïve mean (which is capable of combining scores from a LOF-only ensemble), but not higher than that of taking the maximum of multiple LOF runs (as suggested in LOF [Bre+00]). z-score based normalization

Table 7.2: Ensemble performance with mixed distance functions (LOF, $k = 10$)

Scaling	Combination	Pruning	Reference	ROC AUC	Gain/naïve	Gain/best
Mixture Model	Parallel		[GT06]	(.1618)	(unstable)	
Mixture Model	Serial		[GT06]	.4988	-2.079	-3.149
L_1 Metric (uninformed choice)				.7895	-0.293	-0.743
Average ensemble member				.7930	-0.271	-0.714
Sigmoid	Parallel		[GT06]	.8345	-0.017	-0.370
Identity	Mean			.8372	-	-0.348
Sigmoid	Serial		[GT06]	.8391	+0.012	-0.332
Identity	Max		[Bre+00]	.8397	+0.015	-0.327
CDF+COP	Inv.-Mult.		[Kri+12]	.8459	+0.053	-0.276
z-score	Mean		[NAG10]	.8467	+0.058	-0.269
Rank	Max			.8531	+0.098	-0.216
Best individual method (Canberra distance)				.8792	+0.258	-
CDF	Bayes	Greedy	(new)	.8917	+0.335	+0.103

as used in HeDES [NAG10] and cdf normalization with COP [Kri+12] rescaling further improved. However, and this is the downside, none of them were able to achieve the performance of the best individual ensemble member, using Canberra distance. Combining the cdf scores using the Bayesian rule and pruning the ensemble using the greedy strategy however, yields a gain over the single method. The ensemble consists of 5 methods, indicated by red arrows in Figure 7.4. Except for the maximum norm – which works not very well on this data set – this is probably what a human would have chosen after seeing the performance on the ground truth. The ensemble contains both the best performing method (using Canberra distance), the method that seems to be best for combinations (Clack distance), one of the highly similar divergence measures (χ^2 divergence) and with $L_{0.8}$ and L_∞ the two most different methods from the Minkowski family.

7.4.1.3 Combining Different Algorithms

Probably the most interesting and challenging combination is that of different algorithms. For this setup, we limited k to $k \in \{2, 10, 30\}$, use L_1 distance, but include algorithms kNN-Outlier, kNN-Weight, LOF, LDF, LDOF, Simplified-LOF and LoOP, resulting in 21 combinations. We deliberately chose a low, medium and high k as the different algorithms have different optimal parameters (as seen in Figure 7.3a). Yet, we also want to keep the candidate set small for this experiment.

The results are given in Table 7.3. This time, the mixture model approaches both failed, and one of the sigmoid fitting approaches also. Unsurprisingly, without rescaling, the maximum score yields rather bad results (dominated by the kNN-Weight results for large k). Using a rank normalization – roughly what is called “breadth-first” in [LK05] – slighly improves over the average ensemble member. One of the sigmoid fitting [GT06] approaches worked, but was slightly worse than just using the unscaled average outlier scores. The first method to give a measureable improvement over the naïve ensemble is the HeDES-like approach [NAG10], but it still performs worse than the best ensemble member (LoOP, with $k = 10$). The cdf based approaches again work best, with the COP scaling being slightly worse than the approach based on Bayesian score combination and ensemble pruning.

The greedily constructed ensemble consists of LoOP $k = 30$, kNN-Outlier $k = 2$, LOF $k = 2$, LOF $k = 30$ and LoOP $k = 2$. The approach clearly was able to recognize that the best combinations involve a small and a large value of k, and that kNN-Outlier performance does not improve when increasing k on this data set.

For the first experiment, we kept the number of candidates low (21 candidates). If we increase the candidate set size using 7 algorithms, 8 different values of k and 6 different metrics yielding a total of $7 \cdot 8 \cdot 6 = 336$ instances, the benefits of ensemble pruning become more apparent. Largely due to the more powerful distance functions, the overall results increased, as it can be seen when comparing Table 7.4 to the previous setting, Table 7.3. This is most evident by the best performing single method improving from $.8255$ to $.8978$. The heuristic approach using z-standardization of the outlier scores (as used by HeDES [NAG10]) however continues to perform unexpectedly well, and improves substantially over the median and average performance. The greedy ensemble (retaining a diverse subset of just 9 methods out of the 336 candidates) however yields a new best result for this data set. As control, we also compute the average performance of ensembles consisting of 9 randomly chosen methods. The ensemble constructed using the greedy strategy is 1.259 standard deviations better than the average random ensemble of the same size. The methods retained by the greedy ensemble are: LoOP $k = 30$ with Manhattan distance; LOF $k = 30$, Simplified-LOF $k = 5$ and LDF $k = 25$ with Jensen-Shannon divergence; LOF $k = 2, 5, 15, 25$ and LDF $k = 8$ with Canberra metric. The ensemble shows a good diversity, while at the same time preferring the more powerful distance functions Canberra and Jensen-Shannon.

7.4.2 Different Ensemble Combination Rules

In Figure 7.6 we evaluate different methods of normalizing and combining scores. The baseline method of using the average (or sum; e.g. Weighted-kNN) can be found in Figure 7.5a, and the popular alternative of using the maximum (e.g. LOF when using multiple values of minPts) is in Figure 7.5b. Unsurprisingly, using the unscaled maximum only works when using algorithms with approximately the same scale (e.g. LOF). The simplest normalization – linear scaling to $[0; 1]$ – does not work very well with either normalization: When combined using the average (Figure 7.5a), most combinations yield 0 gain. For the maximum (Figure 7.5b), results are even worse. Another popular method of normalizing data, known as z-score standardization, was used for normalization in HeDES [NAG10], can be seen in Figure 7.5e. This approach is already more robust for using different methods: for example the combination of LDF with a large k and LOF with a low k yields a gain for the first time. (For column naming, refer to

Table 7.3: Ensemble performance with different algorithms (L_1 metric, $k = \{2, 10, 30\}$)

Scaling	Combination	Pruning	Reference	ROC AUC	Gain/naïve	Gain/best
Mixture Model	Serial		[GT06]	(.5000)	(unstable)	
Sigmoid	Serial		[GT06]	(.5296)	(unstable)	
Mixture Model	Parallel		[GT06]	(.5599)	(unstable)	
Identity	Max		[Bre+00]	.6903	-0.555	-0.775
Average ensemble member				.7192	-0.410	-0.609
Rank	Max			.7782	-0.113	-0.271
Sigmoid	Parallel		[GT06]	.7988	-0.010	-0.153
Identity	Mean			.8008	-	-0.142
z-score	Mean		[NAG10]	.8209	+0.101	-0.026
Best individual method (LoOP, $k = 10$)				.8255	+0.124	-
CDF+COP	Mult.		[Kri+12]	.8313	+0.153	+0.033
CDF	Bayes	Greedy	(new)	.8400	+0.197	+0.083

Table 7.4: Large ensemble (336 methods):
Distances: Manhattan, Euclidean, Canberra and Clark distances [DD06], Cosine similarity, Jensen-Shannon divergence [ES03]
Algorithms: LOF, Simplified LOF, LoOP, LDOF, LDF, kNN-Outlier, kNN-Weight
$k = \{2, 5, 8, 10, 15, 20, 25, 30\}$

Scaling	Combination	Pruning	Reference	ROC AUC	Gain/naïve	Gain/best
Mixture Model	Serial		[GT06]	(.5000)	(unstable)	
Mixture Model	Parallel		[GT06]	(.5000)	(unstable)	
Sigmoid	Serial		[GT06]	(.5001)	(unstable)	
Sigmoid	Parallel		[GT06]	.5346	-1.741	-3.554
Identity	Max		[Bre+00]	.7036	-0.746	-1.900
Median ensemble member				.7430	-0.514	-1.515
Average ensemble member				.7463	-0.494	-1.482
CDF+COP	Inv. Mult.		[Kri+12]	.7805	-0.293	-1.148
Identity	Mean			.8302	-	-0.661
CDF	Bayes	Random-9	(control)	.8437	+0.080	-0.529
Rank	Max			.8513	+0.124	-0.455
z-score	Mean		[NAG10]	.8925	+0.367	-0.052
Best individual (LoOP, $k = 10$, Jensen-Shannon)				.8978	+0.398	-
CDF	Bayes	Greedy	(new)	.9229	+0.546	+0.246

Figure 7.3. The most advanced method – using cdf normalization (Section 5.3.2.6) and Bayesian scaling Equation 7.3 – is presented in Figure 7.5f. The improvements over the z-score are hard to see in this representation, but usually are a more pronounced than before. This is probably best to see in the LOF and LDOF combination (row 3 and column 6), but also numerically.

In [GT06], the authors proposed an approach based on a similar paradigm. However, in our experiments it worked much worse. The two normalization approaches based on sigmoid fitting and mixture modeling both turned out to be unstable, resulting in the artifacts visible in Figure 7.6a, Figure 7.6b, Figure 7.6c and Figure 7.6d. The sigmoid fitting in combination with using the "parallel" configuration (i.e. the multiplication of the outlier scores) worked for some cases, but was unable to combine different methods well enough for complex ensembles. There is an interesting contrast to combining cdf scaling with the COP score normalization Equation 5.9: here, the results were next to identical for both the multiplicative and the inverse multiplicative configuration.

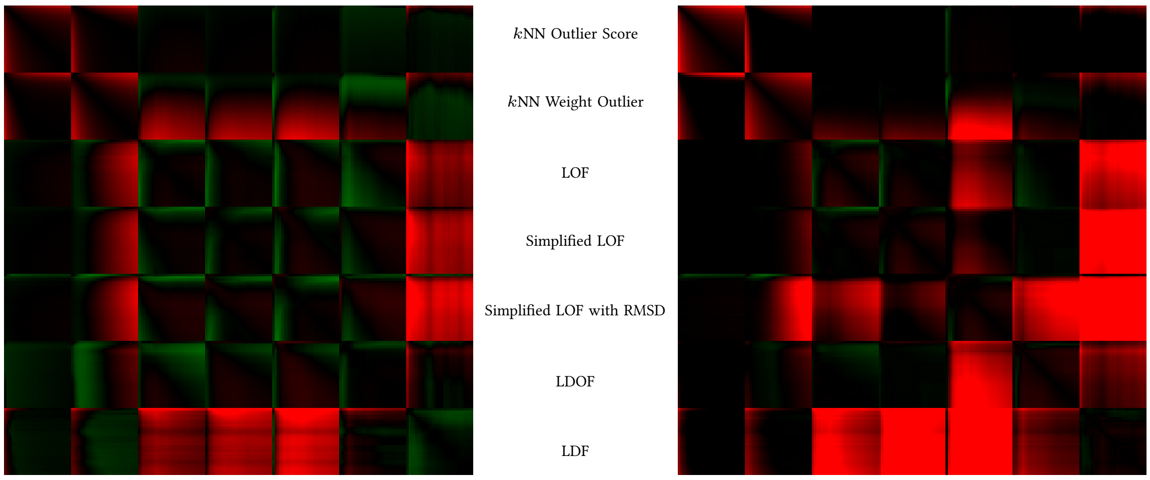
7.4.3 Subsampling Ensembles

Subsampling ensembles are non-trivial to analyze, as discussed before, due to incomparable parameterization. On the ALOI data set, as we have seen in Figure 7.3a, rather low values of k are appropriate. We subsampled this data set with $s = 0.2$ such that we can expect a good performance with a low k of about 2 (as seen in [Zim+13], extreme low values of k often performed best).

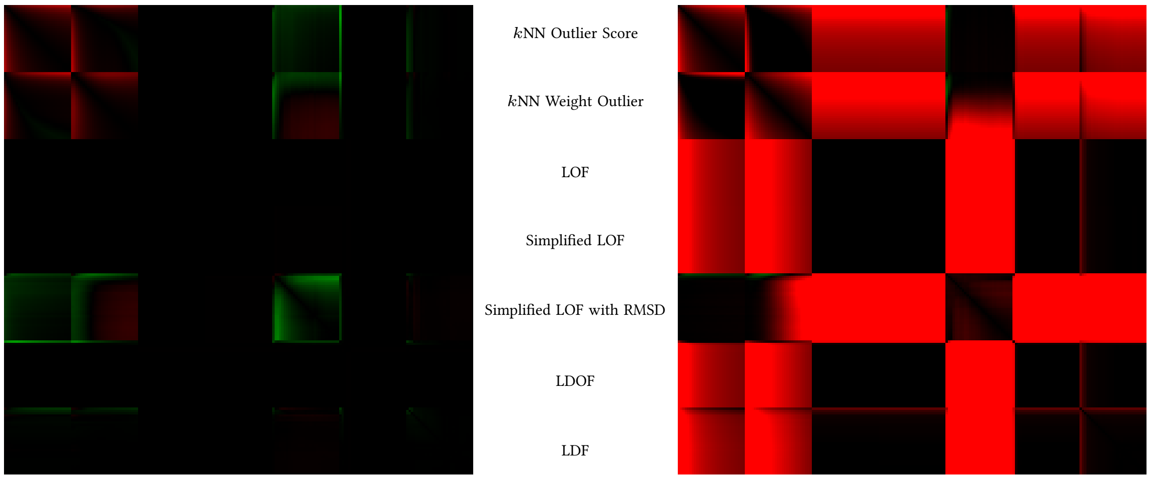
Figure 7.7 visualizes the results for 30 samples (with $s = .2$) along with the result on the full data set, and the performance of the ensemble using the mean of all samples. While the ensemble clearly outperforms the individual members – confirming the results of [Zim+13] that sampling is amenable for ensemble construction on a different data set – it performs worse than the method using the complete ALOI data set. We have seen in the analysis that increasing k reduces variance, and thus the ensemble becomes more similar to the individual results then. Also, the greedy ensemble performance drops with increasing k, which can also be attributed to the decreasing variance.

In this plot we can clearly see the aforementioned parameterization problem. LOF of the complete data performs best with $k = 9$, while the samples perform best with $k = 3 \ldots 5$. Afterwards, the performance drops rather quickly. The benefit from using the ensemble is largest with $k = 2$ because of the larger variance. Nevertheless, the results on the complete data clearly outperform even the ensemble over all samples, except for $k = 2$. Both the naïve and the greedy ensembles over $k = 2 \ldots 30$ however yield better results (see also Table 7.1).

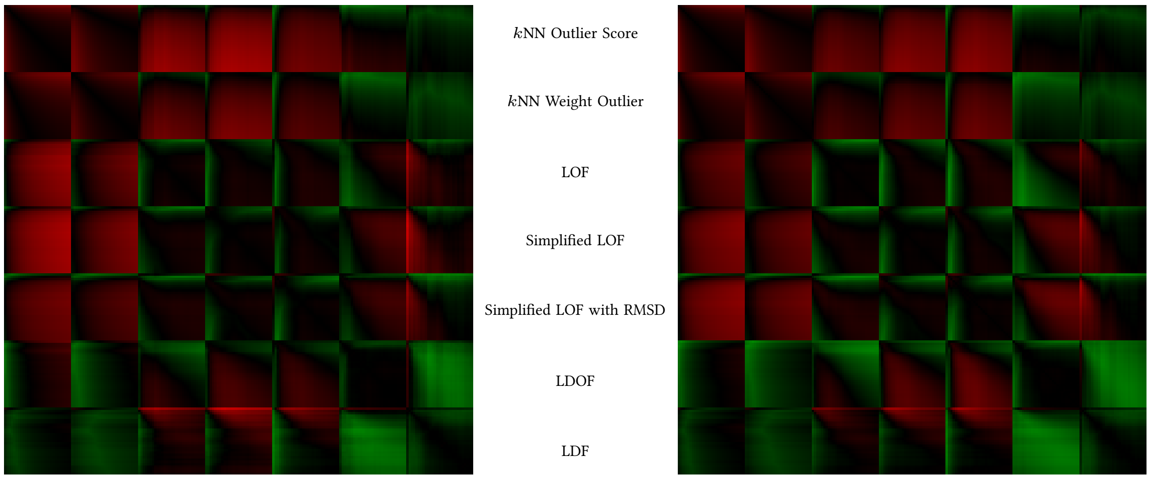
In Figure 7.8 we visualize the similarity of the full data results with $k = 2 \ldots 30$ to (i) the last ensemble member, for $k = 1 \ldots 30$, and (ii) all ensemble members at $k = 2$. The highest similarity to the full data results can be found for $k = 30$ on the full data set, and the sample $k \approx 5$; the similarity of the $k = 2$ ensembles is largest in the $k = 5 \ldots 15$ range of the base method.



(a) Unscaled, mean (b) Unscaled, maximum

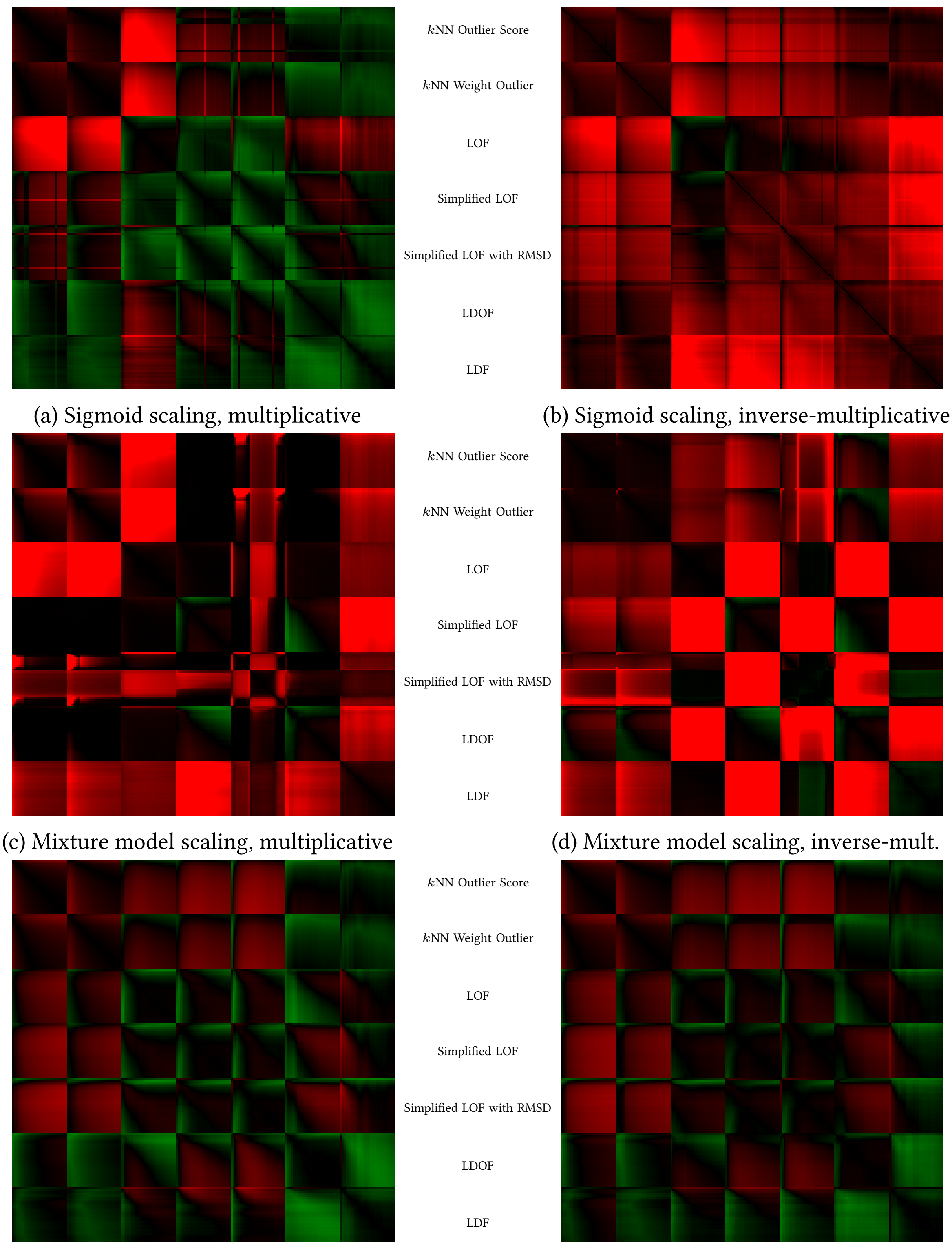


(c) Linear normalied, mean (d) Linear normalized, maximum



(e) z-score normalized, mean [NAG10] (f) cdf normalized, Bayesian combination

Figure 7.5: ALOI: Gain with different combination functions.



(a) Sigmoid scaling, multiplicative

(b) Sigmoid scaling, inverse-multiplicative

(c) Mixture model scaling, multiplicative

(d) Mixture model scaling, inverse-mult.

(e) COP scaling, multiplicative

(f) COP scaling, inverse-multiplicative

Figure 7.6: ALOI: Gain with different combination functions.

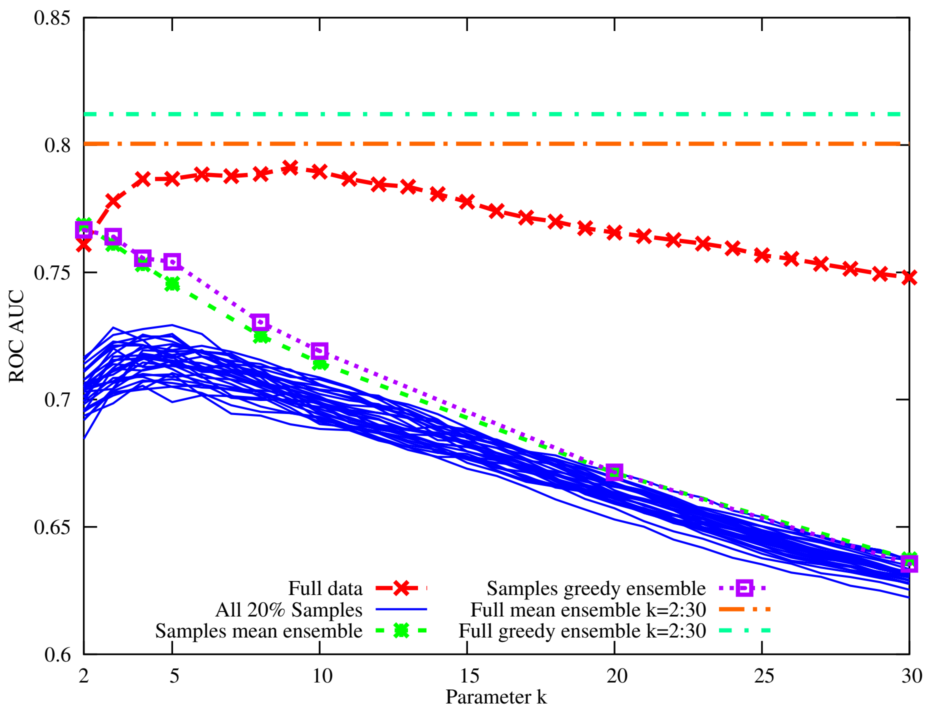


Figure 7.7: Performance of LOF on full ALOI data set, $s = .2$.

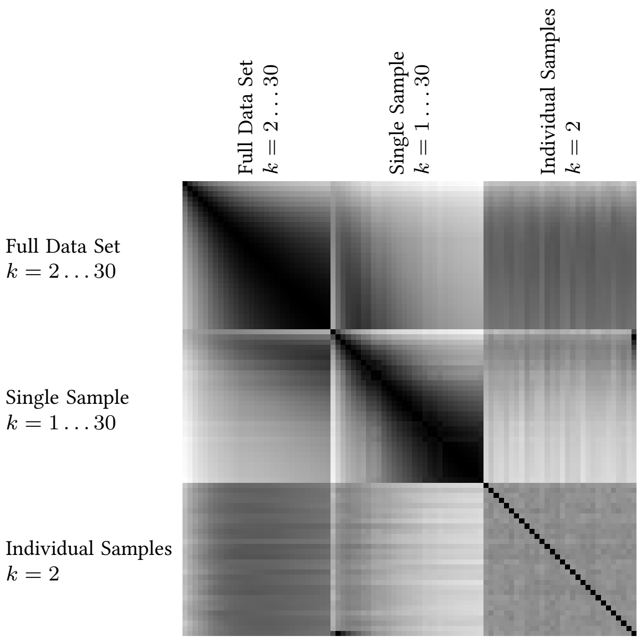


Figure 7.8: Similarity between LOF performance on complete ALOI data set compared to subsampling with $s = .2$ (single sample, all samples with $k = 2$)

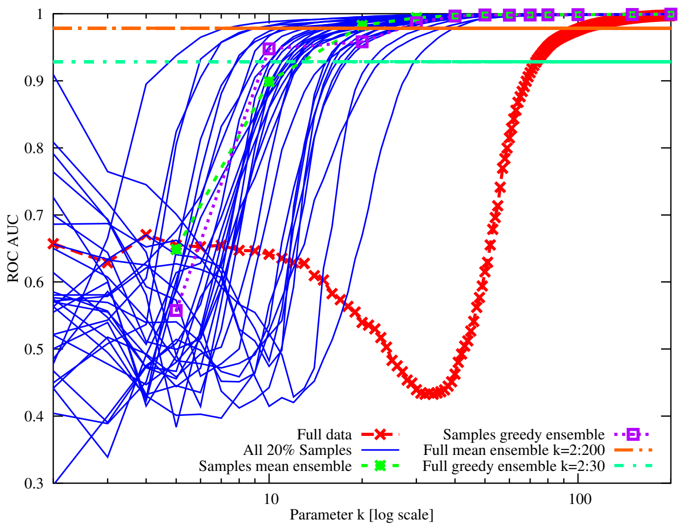


Figure 7.9: Performance of LOF on full SAT-2 data set, $s = 0.2$

On the Satimage data set (as used in [Zim+13]), subsampling at first appears to work very good: for $k = 50$, the performance both on the individual samples as well as the ensemble is much better than the full data set. However, when performing an extensive analysis on the data set, it shows that this observation is somewhat misleading. As seen in Figure 7.9, LOF on the complete data set can perform very well, when k is chosen large enough. For $k < 50$, it does not work very well, but for $k > 150$ its performance becomes close to perfect. This is likely due to a high number of duplicates or near-duplicates in the data set. The individual ensemble members show similar artifacts, but due to subsampling at $k < 20$. Thus, when comparing at a fixed $k = 50$, the subsamples appear to work much better than LOF on the complete data set. On this data set, ensembles over all $k = [2; 200]$ did not work very well, likely due to the near-random performance of a large number of k values. The greedy ensemble, which tries to maximize diversity and prunes redundant ensemble members, suffers from this much more than the naïve ensemble.

The benefits of using subsampling may, therefore, be a bit different than expected. From these experiments, we conclude that:

- Subsampling is amenable to ensemble construction when the sampling rate s is low enough to induce variance.
- Parameterization is not comparable to the complete data set. On the sample, a smaller $k' \approx s \cdot k$ should be used for comparable results.
- Subsampling can improve runtime performance due to both (i) the sampled data sets being smaller, and (ii) the parameter k' being smaller.
- Subsampling works well with naïve ensembles. Since the scores of the individual samples usually have comparable distributions, there is less need for scaling and greedy pruning will often not yield further improvements.

8 Scalability

“Although computer memory is no longer expensive, there’s always a finite size buffer somewhere. When a big piece of news arrives, everybody sends a message to everybody else, and the buffer fills. — *Benoît Mandelbrot* in [Bro04]”

In the previous chapters, we have investigated the mathematical background of outlier detection methods, improved their usability and result quality as well as their flexibility by making them applicable to a much broader range of data. The remaining key step is acceleration: although computers get faster and have more memory, our data sets grow even faster than our processing power. The last few years, companies have been collecting vast amounts of data and are now in the need of methods to analyze it. The challenges arising from this data and the technologies developed to manage such data coined the term of “big data”. While there are multiple aspects to this term, the most widely known is their volume: problems that scale quadratic with the data set size will become too expensive to compute at some point.

At the same time, what is the value of the exact computation, when our data will always only be an approximation of reality? With our data only being a sample, chances are that the exact computation is not substantially closer to the truth than an approximation. In fact, on noisy data, the rather crude statistic of the median (using only 1 or 2 exact values, and the others just for sorting) has proven to be more reliable than the mean, which is influenced by all data. And in outlier detection, we *must* accept that our data will be noisy and impure.

“Data analysis must progress by approximate answers, at best, since its knowledge of what the problem really is will at best be approximate. It would be a mistake not to face up to this fact, for by denying it, we would deny ourselves the use of a great body of approximate knowledge, as well as failing to maintain alertness to the possible importance in each particular instance of particular ways in which our knowledge is incomplete. — John W. Tukey [Tuk62]”

Mathematics is often said to be the only exact science. But this only holds as long as we do not get real data and observations into the picture. While based on mathematics and trying to live up to these standards, neither statistics nor data mining can ever be considered to be “exact”.

“If all the angels united, they would still be able to produce only an approximation, because in historical knowledge an approximation is the only certainty – but also too little on which to build an eternal happiness.
— *Søren A. Kierkegaard* in
Concluding Unscientific Postscript to the Philosophical Fragments (1846)”

8.1 Runtime Cost of Outlier Detection

When aiming at accelerating outlier detection, the first question to answer is where the performance bottlenecks of the algorithm are. While there are bottlenecks in theory, they may be different in practical use. A good example of this are HeapSort and QuickSort. In theory, HeapSort is asymptotically optimal. However in practice, QuickSort is the most important sorting algorithm, and most implementations will even fall back to InsertionSort for small arrays. The reason why InsertionSort is best for small arrays is because of our hardware: linear memory access is well optimized by CPUs, and when sorting happens in nearby memory (as with insertion sorting small parts) the CPU can make best use of caching and parallel execution of instructions. QuickSort, when implemented with good heuristics on the other hand exploits the nature of data, which is often not totally random, but may already be sorted to a large extend. Where HeapSort will always first completely rearrange the data in a backwards heap, then deconstruct this heap into a sorted array, QuickSort can largely just skip over already sorted parts of the data.

So in the end, we will always need to look at both theoretical complexity analysis as well as careful benchmarking, and code optimization. Over the development history of ELKI [AKZ08; Ach+09; Ach+10; Ach+11; Ach+12; Ach+13] implementations of the same algorithm became 5 times faster, by optimizing implementation details, but not by changing the actual algorithm; alternate implementations were sometimes found to be two orders of magnitude slower, while indexes can yield substantial speed ups for large data sets.[1] So in the end, performance is a matter of multiple factors: theoretical complexity, hardware reality, and implementation details. With benchmarks we will never be able to rule out the last factor; and the second factor may change with CPU generations. Yet, if we have a clear result on which parts to optimize, we have a good chance to identify real bottlenecks.

The runtime bottleneck we identified for outlier detection, fortunately, is supported both by theory and practical benchmarking. Table 8.1 gives experimental results for running LOF on the ALOI color histograms and DBpedia data sets, along with commonly given theoretical complexity results. From this data it becomes clear that theoretical results can be far from the reality due to the constant factors dropped during analysis. Performing all n^2 distance computations (with $n = 75000$) is by no means 75000 times slower than loading the data from disk, and bulk-loading of an index is much faster than querying it for each object (although both have asymptotic complexity of $\mathcal{O}(n \log n)$). Nevertheless, also from a theoretical point of view, finding the kNN of each object in the database is the most expensive part of LOF and similar algorithms. This not only involves distance computations, but also e.g. managing a priority queue for the search process and for keeping track of the current top k neighbors. On the ALOI data set, likely due to the dimensionality of 27 dimensions and the “high” value of $k = 100$,[2] the k-d tree fails to accelerate the queries, and the R*-tree also only yields a small performance gain over

[1] Benchmark details can be found at: `http://elki.dbs.ifi.lmu.de/wiki/Benchmarking`

[2] Actually, we use $k = 101$, and then remove the query point from the result.

Table 8.1: Runtime breakdown of LOF (with $k = 100$) in ELKI 0.6.0

Data set	ALOI 75k 27d						Theoretical
Index	linear scan		k-d tree		R*-tree		complexity
Load Ascii data	2238 ms	0.96%	2232 ms	0.50%	2231 ms	1.27%	$\mathcal{O}(n)$
Bulk-load index	0		624 ms	0.14%	996 ms	0.56%	$\mathcal{O}(n \log n)$
kNN search	230030 ms	98.84%	446653 ms	99.28%	172791 ms	97.99%	$\mathcal{O}(n^2)$ resp. $\mathcal{O}(n \log n)$
LOF	468 ms	0.20%	372 ms	0.08%	321 ms	0.18%	$\mathcal{O}(nk)$
Data set	DBpedia 475k, 2d						Theoretical
Index	linear scan		k-d tree		R*-tree		complexity
Load Ascii data	990 ms	0.04%	1057 ms	4.82%	1035 ms	5.99%	$\mathcal{O}(n)$
Bulk-load index	0		829 ms	3.78%	768 ms	4.44%	$\mathcal{O}(n \log n)$
kNN search	2672128 ms	99.74%	15740 ms	71.72%	11379 ms	65.85%	$\mathcal{O}(n^2)$ resp. $\mathcal{O}(n \log n)$
LOF	5879 ms	0.22%	4319 ms	19.68%	4099 ms	23.72%	$\mathcal{O}(nk)$

the linear scan. On the larger, but only 2-dimensional data set with latitude and longitude coordinates from DBpedia – using Euclidean distance, not geodetic distance, as we are not using the actual results; and the k-d tree in ELKI only supports L_p-norms – both the k-d tree and the R*-tree offer substantial acceleration.

Looking at this breakdown, it becomes obvious that the cost is dominated by the nearest neighbor search, which is known to be a hard problem and has been subject to decades of research in the database community. The k-d tree [Ben75] and R*-tree [Gut84; Bec+90] are two solutions proposed for this, but which work best for low-dimensional data. On large data sets, finding the k nearest neighbors of each object is a fairly expensive operation, and responsible for the major share of the runtime (over 99% for large data sets without indexing support). The actual density estimation and outlier analysis then is only a minor operation.

In traditional database literature, exact nearest neighbor search has received the most attention. This is unsurprising, because in the early use cases of databases, incomplete and incorrect answers would have been unacceptable: accounting has been a key use case for early databases, and how useful is "approximately correct" accounting data? In data mining however, and in outlier detection in particular, we cannot assume our data to be complete or correct. Instead, we are actually searching for data that might not be correct. Most of the data sets that we analyze are only a snapshot and excerpt of reality, with various kinds of errors in it. Therefore, approximate nearest neighbor search is a viable option, if it allows us analyzing data sets that we could no longer process otherwise.

> “ Each piece, or part, of the whole nature is always an approximation to the complete truth, or the complete truth so far as we know it. In fact, everything we know is only some kind of approximation, because we know that we do not know all the laws as yet. — *Richard P. Feynman* in [FLS63] ”

8.2 Approximate Nearest Neighbors in Dense High-Dimensional Data

We will first discuss the most common techniques for approximate nearest neighbor search in dense high-dimensional data, along with theoretical foundations and established methods. Note that for sparse data, where most attribute values are 0, most of these approaches will not work without modifications. And for very high dimensionalities, when distance functions no longer offer meaningful contrast, approximate indexes will also not offer much relief [Kab11] as they are still approximating the low-contrast distance function.

An important result for approximate nearest neighbor search is the Johnson–Lindenstrauss lemma [JL84], which proved the existence and bounds of a projection of n objects into a lower dimensional space of dimensionality $\mathcal{O}(\log n/\varepsilon^2)$, such that the distances are preserved within a factor of $1+\varepsilon$. Matoušek [Mat08] further improves these error bounds. The most interesting and surprising property is that the reduced dimensionality depends only logarithmically on the number of objects and on the error bound, but *not* on the original dimensionality d. Different ways of obtaining such a projection have been proposed for common norms such as Manhattan and Euclidean distance. A popular choice are the "database-friendly" random projections [Ach01; Ach03], where $2/3$ of the terms are 0 and the others ± 1 (along with a global scaling factor of $\sqrt{3}$), which can be computed more efficiently than the previously used matrices. Another popular choice are projections based on s-stable distributions [Dat+04], where the Cauchy distribution is known to be 1-stable and the Gaussian distribution to be 2-stable [Zol86] (i.e. they preserve L_1 and L_2 norms well). See [VW11] for an overview and empirical study on different variations of the Johnson-Lindenstrauss transform, indicating that a reduced dimensionality of $k = 2 \cdot \log n/\varepsilon^2$ will usually maintain the pairwise distances within the expected quality.

8.2.1 Dimensionality Reduction by Feature Selection

The simplest way to reduce dimensionality is to skip some dimensions. When this is done consciously, e.g. by measuring the discriminative power of features it is called "feature selection". For an overview of feature selection approaches, see e.g. [LM98]. When random features are chosen, it is the base for the ensemble method known as "feature bagging" [Bre96], and was used in Chapter 7 as diversity source. Randomized feature selection is trivial to compute and clearly the fastest approach, but it neither gives strong guarantees on the quality of the approximation, nor does it give a high speedup. The reduced dimensionality may, however, make traditional indexing methods more effective again.

For outlier detection, feature bagging was proposed in [LK05], who construct an ensemble based on detectors who each use a random subset consisting of $[\lfloor d/2 \rfloor; d-1]$ dimensions. For indexing and fast nearest neighbor search, it is desireable to fix the number of dimensions to a smaller value that aligns well with the page size of the on-disk representation.

8.2.2 Dimensionality Reduction by Random Projections

Mathematically, dimensionality reduction by random projections is a matrix multiplication:

$$v' = M \cdot v$$

where the matrix M has the size $d \times d'$ with $d' < d$. The challenge is to construct a matrix M such that the distances are optimally preserved. Learning an optimal projection matrix (e.g. using multidimensional scaling or linear discriminant analysis [Fis36]) is rather expensive and may not be able to accommodate changing data patterns over time, so the more common approaches construct such a matrix based on randomization. Some well known (with known error bounds) approaches are:

- "Database-friendly" random projections [Ach01; Ach03]:

$$M_{\text{Achlioptas-2}\,ij} = \begin{cases} +1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases}$$
$$M_{\text{Achlioptas-3}\,ij} = \sqrt{3} \begin{cases} +1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -1 & \text{with probability } \frac{1}{6} \end{cases}$$

  where in particular the second one is considered "database friendly" in the sense that the resulting matrix is sparse and contains only $\{-1, 0, +1\}$, meaning it can be computed using additions only (and a single multiplication).
- s-stable random projections [Dat+04]:

$$M_{\text{Cauchy}\,ij} \sim \text{Cauchy}(0; 1)$$
$$M_{\text{Gaussian}\,ij} \sim \mathcal{N}(0; 1)$$

Note that in each of these cases, the individual cells are independently drawn from the same distribution (i.i.d.). There exist variations which normalize projection vectors to unit length (see [VW11] for further information).

The first family of projections are for example used in the method projection-indexed nearest-neighbours (PINN) [dCH10; dCH12], while the second type is commonly used with locality sensitive hashing (LSH) [IM98; GIM99; Dat+04].

8.2.2.1 Indexing with Grid Hashing

Locality sensitive hashing (LSH) [IM98; GIM99; Dat+04] can (on dense vector data with L_p-norms) be seen as a grid based indexing approach, combining multiple established strategies:

1. Dimensionality reduction by s-stable random projections to k dimensions.
2. Grid based data binning into $\mathbb{N}^d$ bins of width w.
3. Reduction of grid size by hashing to a finite number of bins.
4. Similarity search in the bin of the query point only.
5. Ensemble of l such projections.

Individual parts of this approach can be substituted to accommodate different data types and similarity functions. For example, instead of computing the hash code on a regular grid, it can be based on the bit string of the raw data, or a bit string obtained by splitting the data space using random hyperplanes. MinHash [Bro97] is a hashing scheme designed for subset similarity search, which can be generalized to the winner-take-all (WTA) hashes for indexing multimedia data [Yag+11], which have recently been used to speed up large-scale object detection and recognition tasks [Dea+13]. LSH is an approximate nearest neighbor search algorithm both due to the use of random projections (which only approximately preserve distances) but also due to searching within the same bin as the query point only. The use of hash tables makes it easy to parallelize and distribute on a cluster.

Probably the largest difficulty in using LSH is choosing the three parameters k, w and l. In particular w is difficult to use in the context of outlier detection. Intuitively, the cell width w is the distance threshold when objects are likely to be in the same grid cell. For distances larger than w, they are likely in different grid cells. If this parameter is chosen too large, cells will contain too many points – in the worst case, all objects are in the same grid cell. If the parameter is chosen too small, in particular objects in sparse areas will have few true nearest neighbors in their grid and instead have random false matches due to the hashing, or even fewer neighbors than desired.

In [WPT11], the authors propose a LSH based outlier detection framework. However – in contrast to what the authors state – it cannot be used for "any distance-based outlier detection mechanism", but it will only be useful for global methods such as kNN Outlier: the key idea of this method is to use LSH to identify low-density regions, and refine the objects in these regions first, as they are more likely to be in the top-n global outliers. For local outlier detection methods, however, there may be iteresting outliers within a globally dense region; and the pruning rules this method relies on will not be applicable.

8.2.2.2 Indexing with Projected Tree Indexes

Projection-indexed nearest-neighbours (PINN) [dCH10; dCH12] shares the idea of using a random projection to reduce dimensionality. On the reduced dimensionality, a classic spatial indexing scheme is then employed to find neighbor candidates:

1. Dimensionality reduction using "database friendly" random projections to d' dimensions.
2. Build a spatial index (R*-tree, k-d tree) on the projected data.
3. Retrieve the $c \cdot k$ nearest neighbors in the projection ($c > 1$).
4. Refine candidates to k nearest neighbors in original data space.

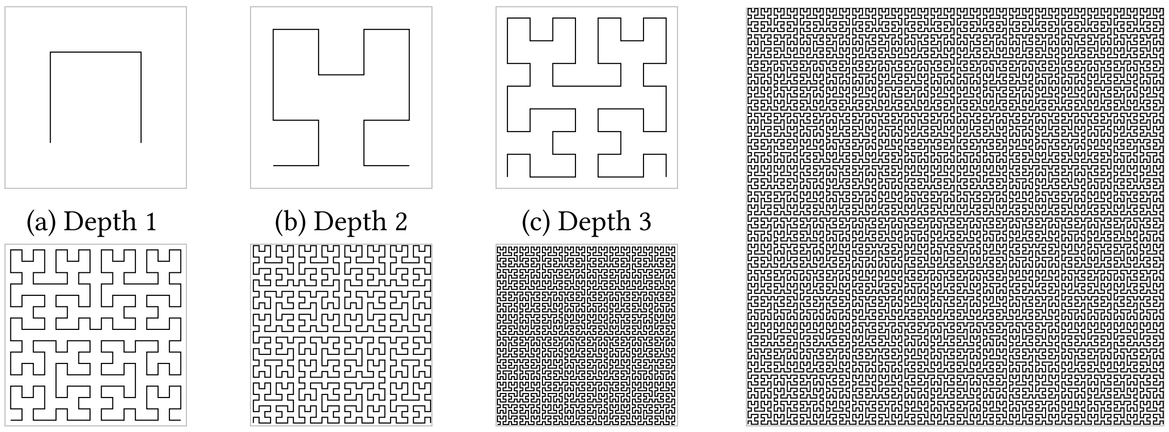
Due to the use of random projections this method may also not return the true k nearest neighbors, but it has a high probability (see [dCH10; dCH12] for detailed error bounds) of retrieving the correct neighbors. In contrast to LSH, it is also guaranteed to return the desired number of neighbors and thus always provide enough data for density estimation and reference sets to be used in outlier detection. When a true nearest neighbors is not found, the false positive will still be spatially close to the query point, whereas with LSH they could be any data.

8.2.2.3 Limitations of Random Projections

The type of random projections discussed in this chapter are not a general purpose technique: the Johnson–Lindenstrauss lemma only gives the existence of a random projection that preserves the distances, but we may need to choose different projections for different distance functions. The projections discussed here were for unweighted L_p-norm distances. Constant weights can be accommodated by instead scaling the data with the inverse weights, but for dynamic weights the results will degrade quickly. Furthermore, it should be noted, as pointed out in [Kab11], that random projection methods are not suitable to defy the concentration of distances (see Chapter 4 for a discussion of the problems of high-dimensional data) aspect of the curse of dimensionality: since according to the the Johnson–Lindenstrauss lemma distances are preserved approximately, these projections will also preserve the concentration effect.

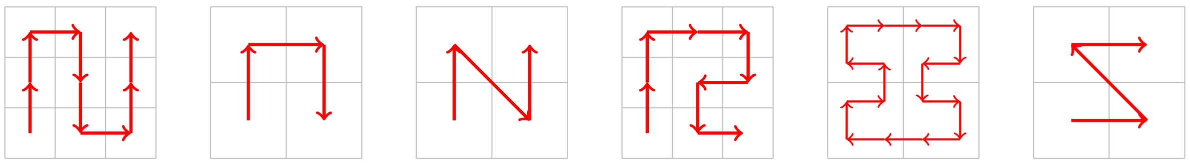
8.2.3 Dimensionality Reduction by Space-Filling Curves

Space-filling curves is a classic mathematical method for dimensionality reduction dating back to 19th century [Pea90; Hil91]. In contrast to random projections as discussed in the previous section, when using space-filling curves the data is always reduced to a single dimension. In fact, the earliest proposed space-filling curves such as the Peano curve [Pea90] and Hilbert curve [Hil91] were defined originally for the two dimensional plane, and have only later been generalized to higher dimensionality. A space-filling curve is a fractal line in a bounded d dimensional space (usually $[0;1]^d$) with a Hausdorff dimensionality of d that will actually pass through every point of the space. When drawn completely, the curve would therefore cover the complete square and be infinitely folded. For illustration purposes, it is common to draw approximations of the curve instead; since these curves are usually defined recursively, the simplest way of defining approximations is by limiting the recursion depth. Figure 8.1 visualizes the Hilbert curve this way at different levels. The self-similar and fractal nature of the curve becomes apparent when comparing the curves of different depth at different scales. At each level, the curve visually consists of four (rotated and scaled) copies of the previous level. Yet,



(a) Depth 1 (b) Depth 2 (c) Depth 3 (d) Depth 4 (e) Depth 5 (f) Depth 6 (g) Depth 7

Figure 8.1: Hilbert curve approximations at different recursion depth.



(a) Peano [Pea90] (b) Hilbert [Hil91] (c) Z-curve [Mor66] (d) Meandering Peano (e) Closed Hilbert (f) Rotated Z-curve

Figure 8.2: Primitives and variants of different popular space-filling curves.

these four copies are actually connected to each other, forming a single line that never crosses itself. Figure 8.2 shows the primitive constructs of the three most widely known space-filling curves, the Peano, Hilbert and Z-curves. For some curves there exists variants; for example the “meandering” Peano curve (Figure 8.2d), the closed loop Hilbert curve (Figure 8.2e, which actually uses the same primitive element as the Hilbert curve, only the first level arrangement is different) and the rotated Z-curve (Figure 8.2f).

The first curve used for databases was the Z-order used by Morton [Mor66] for indexing multi-dimensional data for range searching. This curve is also known as Morton code and Lebesgue curve. This curve can be obtained by interleaving the bits of two bit strings x_i and y_i into a new bit string: $x_1y_1x_2y_2x_3y_3x_4y_4\ldots$. The first mathematically analyzed space-filling curve was the Peano curve [Pea90], closely followed by the Hilbert curve [Hil91] which is considered to have the best mathematical properties. The Peano curve has not received much attention from the data indexing community because it splits the data space into thirds, which makes the computations rather complex. The Hilbert curve, while rather tricky in high-dimensional data due to the different rotations of the primitives, can be implemented efficiently with bit operations [But71], and has for example been used for bulk-loading the R*-tree [KF94] and image retrieval [Ngu+12].

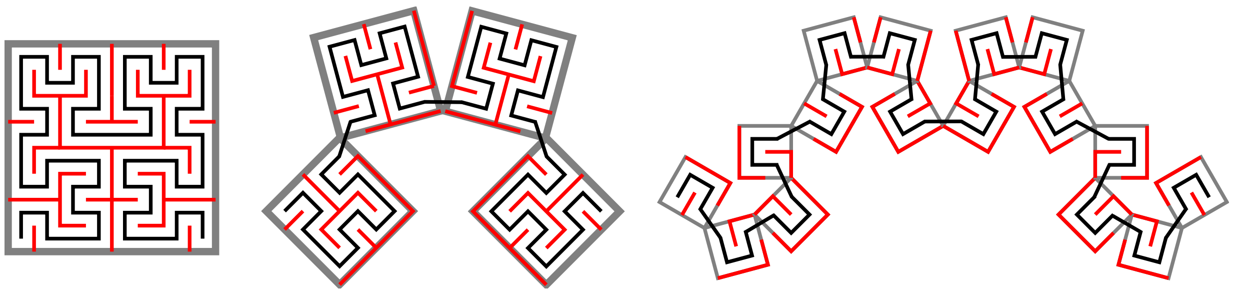


Figure 8.3: Visualization of Hilbert curve unfolding. Red edges are cut at curve iteration 3.

8.2.3.1 Indexing Data Using Space-Filling Curves

Indexing data with space-filling curves as suggested by Morton [Mor66] is straightforward: the data is first projected to the 1-dimensional coordinate, then indexed using a B-tree or similar data structure. However, *querying* such data is the main challenge: while this kind of index can answer exact matches well, and also rectangular window queries by computing the intersections of the query rectangle with the space filling curve and then retrieving the corresponding intervals from the index, finding the exakt k nearest neighbors is nontrivial.

For outlier detection, however, we will need query windows of different size in order to find the k nearest neighbors. A basic version of this method was published in [SZM98], which used a large number of “randomly rotated and shifted” curves for image retrieval. Essentially the same approach was also published in [LLL01], but using multiple systematically shifted – not rotated – copies of Hilbert curves and giving an error bound based on [Cha98]. For retrieving the k-nearest neighbors, both methods look at the preceding and succeeding k objects in each curve, and refine this set of candidates. Furthermore, the HilOut [AP02] algorithm uses the same curve layout as [LLL01], but is the first to use this indexing for outlier detection. It differs from the nearest-neighbor search algorithms in the way that it, after each scan of a curve, reduces the set of candidates for the kNN-Weight outlier model, and refines only these candidates when processing subsequent curves.

As the original motivation of space-filling curves was to provide a complete ordering of (2-dimensional) vectors, space-filling curves are an extreme dimensionality reduction technique: they always reduce to a 1-dimensional space, a linear order of all points. Intuitively, they can be interpreted as repeatedly cutting and opening the data space along some edge in the process of linearization. With an increasing number of dimensions, the number of such cuts – where neighborhoods are not preserved – increases. Figure 8.3 is a visualization of the 3rd order Hilbert curve, with the actual curve in black. Red lines indicate neighborhoods that are not preserved well: when the curve is unfolded, the red edges are no longer adjacent along the curve. The second and third images visualize how the curve unfolds if one would pull both ends of the curve apart. It can be seen that space filling curves rely on an intricate (actually infinite and fractal) pattern of cutting the data space into fragments that are then ordered by the position on the curve.

8.2.3.2 Approximation Guarantees of Space-Filling Curves

In [Cha98], approximation guarantees are given for grid-based indexes based on shifting the data diagonally by $1/(d+1)$ times the data extend on each axis. Roughly summarized, the proof shows that a data point must be at least $1/(2d+2) \cdot 2^{-l}$ away from the nearest border of the surrounding cell of size 2^{-l} in at least one of these curves due to the pigeonhole principle. Within at least one grid cell, all neighborhoods within a radius of $1/(2d+2) \cdot 2^{-l}$ therefore are in the same grid cell, i.e. nearby on the same curve. By looking at the length of the shared bit string prefix, we can easily determine the l which we have fully explored, and then stop as desired. Approximate k-nearest neighbor search on such curves – by looking at the k predecessors and successors on each of the $d+1$ curves only – is shown in [LLL01] to return approximate k nearest neighbors which are at most $\mathcal{O}(d^{1+1/p})$ farther than the exact k nearest neighbors, for any L_p-norm. For the 1-nearest neighbor, the error factor is shown to be at most $d^{1/p}(4d+4)+1$.

HilOut [AP02] takes also the bit strings into account to compute minimum and maximum distances. For the simple $k\mathrm{NN}$-Weight outlier model, it can this way identify true negatives, and instead refine the remaining outlier candidates more precisely. The focus of HilOut is to find the top-n outliers exactly, and skip over objects that cannot be in this set early. However, HilOut cannot as easily be adapted to local outlier detection models such as LOF, where a pruning would be needed that is based on local densities.

In our proposed approach, we divert from using the systematic diagonal shifting proposed for which these error bounds are proven. Details of our *randomized* approach will be given in the next section. It can be expected that the errors obtained by randomized projections are on a similar scale on *average*, but we cannot guarantee such bounds for the worst case anymore. We do however achieve better scalability due to the lower dimensionality of our projections, we gain the ability to use other space filling curves and are not restricted to using $d+1$ curves. Similar to how diversity improved outlier ensembles in Chapter 7, we can expect diverse random subspaces to improve the detection result in practice, even when we can no longer give precise theoretical error bounds.

8.2.3.3 Limitations of Space-Filling Curves

Space filling curves are easy to use in low dimensional space, but will not trivially scale up to high dimensionality due to the combinatorial explosion as discussed in Chapter 4 (just as any other grid based approach). They work on recursive subdivisioning of the data space, into 2^d (3^d for the Peano curve) cells, a number which grows exponentially with the dimensionality d. In most cases, the ordering of points will then be determined by binary splits on the first few dimensions only. HilOut suffers both from this aspect of the curse of dimensionality, and from the distance concentration which reduces its capability to prune outlier candidates: since all distances are increasingly similar, the set of outlier candidates does not shrink much with each iteration of HilOut. For this top-n method to perform well, it must quickly be able to shrink the set of candidates to a minimum, so that it can analyze a wider window of neighbors for them.

8.2.4 Fast Approximate kNNSearch with Space-Filling Curves

The proposed method to search for nearest neighbors is closely inspired by the methods discussed before, such as HilOut. However, it is designed with parallelism and distributed computation in mind: where HilOut uses a nested loops approach to refine the current top candidates for a single outlier detection model, we focus on an easy to parallelize method to compute the k nearest neighbors of all objects (often called as kNN-self-join), so that we can then use *any* of the kNN based outlier detection methods discussed in this thesis.

The principle of the proposed method is:

1. Generate m space-filling curves, but using
   a) different curve families (Hilbert, Peano, Z-curve),
   b) different random projections and/or subspaces,
   c) different shift offsets to decorrelate discontinuities.
2. Project the data to each space-filling-curve.
3. Sort data on each space-filling-curve.
4. Using a sliding window of width $w \cdot k$ generate neighbor candidates for each point.
5. Merge the candidates across all curves and remove duplicates.
6. Compute the distance to each neighbor candidate, and keep the k nearest neighbors.

All of these steps can easily be parallelized in a distributed computation framework such as Hadoop and MapReduce. The only step that needs additional coordination is the sorting step; however sorting large data sets on Hadoop is a solved problem, for example by first gathering quantile information, then rearranging the data into partitions and sorting these locally (see the TeraSort benchmark [Ras+11] with the 2013 record being the sort of 1.4 TB in 1 minute).

Algorithm 3 gives the full outlier detection algorithm for the generalized model of Section 6.1, but obviously one can also run other kNN, SNN and reverse-kNN based algorithms on the precomputed neighborhoods. The reverse-kNN are computed by simple list inversion to optimize data communication: this makes it easy to transmit an objects density estimate to the neighbor objects for comparison.

8.2.4.1 Bias in Approximation by Space-Filling Curves

There exists an interesting bias in the approximation using space-filling curves (SFCs), which makes them particularly useful for outlier detection. The error introduced by SFCs scales with the density of the data: if the bit strings of two vectors agree on the first $d \cdot i$ bits, the vectors are approximately[3] within a cube of edge length 2^{-i} times the original data space. For query points in a dense region of the data, the explored neighbors will be closely nearby, whereas for objects in less dense areas – i.e. outliers – the error introduced this way will be much larger on

[3]This does not exactly hold e.g. for the Z-curve which has discontinuities, and the Peano curve is not represented using bit strings, but the intuition nevertheless remains valid; furthermore we will be using random projections.

Algorithm 3: Parallel Approximate Outlier Detection

```
// Project data locally
on every node do
    for each block do
        for each curve do
            project data block to curve
            store projected data
            send quantiles to coordination node
        end
    end
endon
// Estimate distribution for sorting
on coordination node do
    for each curve do
        Merge received quantiles
        Estimate global data distribution
        send global quantiles to every node
    end
endon
// Rearrange data in cluster
on every node do
    for each curve do
        for each projected block do
            split according to global quantiles
        end
    end
endon
shuffle to new blocks
// Process sliding windows
on every node do
    for each curve do
        for each projected, shuffled block do
            Sort block
            for each object (using sliding windows)
            do
                emit (object, neighbors)
            end
        end
    end
endon
shuffle to (object, neighbor list)
// compute $k$NN and build R$k$NN
on every node do
    for each (object, neighbor list) do
        Remove duplicates from neighbor list
        Compute distances
        store $k$ nearest neighbors of object
        emit (neighbor, object) for each neighbor
    end
endon
shuffle to (object, $k$NN, R$k$NN)
// Compute models
on every node do
    for each (object, $k$NN, R$k$NN) do
        Compute model for object
        emit (object, (object, model))
        emit (reverse neighbor, (object, model))
    end
endon
shuffle to (object, model list)
// Compare models
on every node do
    for each (object, (neighbor, model)) do
        Compare model to neighbor models
        Store outlier score for model
        Collect outlier score statistics
    end
    send statistics to coordination node
endon
// Normalize Outlier Scores
on coordination node do
    Merge outlier score statistics
    send statistics to every node
endon
on every node do
    for each (object, score) do
        Normalize Outlier Score
        if score above threshold then
            emit (outlier, normalized score)
        end
    end
endon
```

average. Furthermore, for an object central to a cluster, “wrong” nearest neighbors tend to still be members of the same cluster, and will just be slightly farther away. For an outlier however, missing one of the true nearest neighbors – which may be another outlier with low density – and instead taking an even farther object actually increases the chance that we end up using a cluster member of a nearby cluster for comparison. So while the approximation will likely not affect inlier scores much, we can expect it to emphasize outliers!

This effect is similar to an observation for subsampling ensembles in [Zim+13] and Section 7.2.3.4: When subsampling a relative share of s objects from a uniform distributed sphere, it can be shown that the kNN-distances are expected to increase by a relative factor of $(1 - s^{1/d})/s^{1/d}$. Since for outliers this distance is expected to be higher, the expected increase will also be larger, and thus the outlier will become more pronounced with respect to this measure.

8.2.5 Different Kinds of Approximations

While all approaches discussed before can be used to find the approximate nearest neighbors efficiently, they are based on subtly different foundations of approximation.

Random projections are designed to approximately *preserve distances*, while reducing dimensionality. Using an exact index on the projected data, as done in PINN, will therefore find the k nearest neighbors with respect to the approximate distance. The index based on locality sensitive hashing (LSH) in contrary is lossy: it is designed to have a high chance of *preserving regions of a fixed size* w, where the size w is a critical input parameter: the smaller the size that needs to be preserved, the faster the index; when the parameter is chosen too high, all objects will be hashed into the same bin, and the index will degenerate to a linear scan. Space-filling curves on the contrary neither aim at directly preserving distances, nor do they try to preserve regions of a given radius. Instead, space-filling curves try to *preserve closeness*: the nearest neighbors of an object will often be nearby on the curve, while far neighbors will be far away. For the purpose of density based outlier detection, this yields an important effect: the index based on space-filling curves is better at adapting to different density in the data set than the other two indexes, which makes it more appropriate for local density based outlier detection methods.

As a rule of thumb, the following can be used:

- If the exact distances are of importance, PINN is expected to work best.
- If neighborhoods for a known, small radius w are needed, LSH is expected to work best.
- If k-nearest neighborhoods are needed, SFC is expected to work best.

8.2.5.1 Indexing Using 1-Dimensional Random Projections

As an additional baseline method, we also evaluate the following method for approximate nearest neighbor search: similar to PINN and our SFC method, the data is projected using a random projection; however we only use a 1-dimensional output space. Since the projected data is already 1-dimensional, it can then be sorted on this axis without the need of using a space-filling curve. The query process we use however is the same as proposed for SFC: the neighbors within a sliding window of $w \cdot k$ are added to the candidate set, and exact distances are only computed on this candidate set. This baseline method can therefore be seen as an the extreme case of combining the space-filling curve approach with a 1-dimensional random projection.

8.2.6 Evaluation

The method proved to be effective for outlier detection. Each space-filling curve will return some of the true nearest neighbors, and by using enough curves and a sufficiently large window size, a strong set of candidates can be found. At the same time, searching in the sorted space-filling curves is a linear process that does not involve distance computations. Intuitively, w should be chosen greater than 1, but when using many curves, even low values such as $1/\sqrt{m}$ can yield reasonable performance (but in the worst case, may yield less than k candidates).

Figure 8.5 visualizes the results for running the LOF algorithm on the 27-dimensional variant of the ALOI data set with $k = 20$ on a single CPU core. To make the results readable – we evaluated over 4000 different index variations total – we visualize only the skyline results in Figure 8.4a. The skyline are all objects where no other result is both faster and has a higher score at the same time (upper skyline) or where no other result is both slower and scores less at the same time (lower skyline). The upper skyline is primarily useful for judging the *potential* of a method, when all parameters were chosen optimal, whereas the lower skyline indicates the worst case. Figure 8.4a visualizes the skyline of outlier detection quality vs. runtime, whereas Figure 8.4b is an evaluation of the actual index by measuring the recall of the true 20 nearest neighbors. In both measures, the SFC based method has the best potential – it can even be better than the exact indexes – while at the same time it is usually an order of magnitude faster than PINN and two orders of magnitude faster than LSH (both at comparable quality). And even when the parameters are chosen badly (which for SFC usually means using too few curves), the results are still comparable to LSH and PINN.

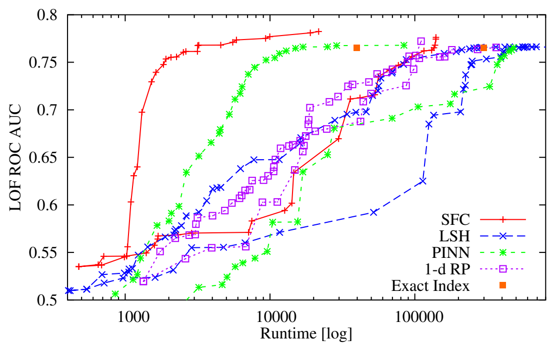
However, there is a surprising difference between these two charts. They are using the same indexes, but the LOF ROC AUC scores for the SFC index start improving quickly at a runtime of 1000-2000 ms. The recall however, starts rising much slower, in the range of 1500-10000 ms. When we choose an average performing combination for the SFC index: 8 curves of different families combined with a random subspace projection to 8 dimensions and a window width of $w = 1$, the runtime is 4989 ms and the ROC AUC is $.747$, and the average recall of the true nearest neighbors is $.132$, it is surprising to see such a good performance *despite* the low recall. Figure 8.5a visualizes the relative error of the 20-nearest neighbor distance compared to the recall. This chart is interesting, because it shows that the SFC curves, even when having a very low recall of less than $.20$ often yield much smaller relative error than the other approaches. This means that while the method does make *more* errors, the errors are *less* severe, i.e. the incorrect nearest neighbors have a smaller distance than with the other methods.

In Figure 8.4d we explore the effect of different random projections. The skyline, marked as "SFC", does not use random projections at all. The curves titled "R SFC" are space filling curves on randomly selected features only (i.e. feature bagging), while "RP SFC" uses full Achlioptas style random projections. Unsurprisingly, the approach not using projections is fastest due to the lower computational cost. Randomly selected features has the highest potential gains (some of which may be an artifact of this data set) and in general the largest variance. Achlioptas random projections offer a similar performance to the full-dimensional SFC, but come at the extra price of having to project the data, which makes them usually slower. Figure 8.4e compares different space filling curves for the full-dimensional SFC subset, with the approach of choosing the curves at random usually giving the best performance: the added diversity improves results at little cost. While Hilbert curves seem to come out worst, this may be an implementation artifact: where Z-curves and Peano curves were implemented using an efficient divide and conquer approach for sorting, the Hilbert curves were implemented by projecting the data to their bitstring representation (using $27 \cdot 31 = 837$ bits) and then sorting this representation. For PINN, we also explore different indexes for the projected data in Figure 8.4f. R*-trees bulkloaded with the sort-tile-recursive (STR, [LEL97]) performed best, as R*-trees usually have better pruning power than k-d trees in medium dimensionality.

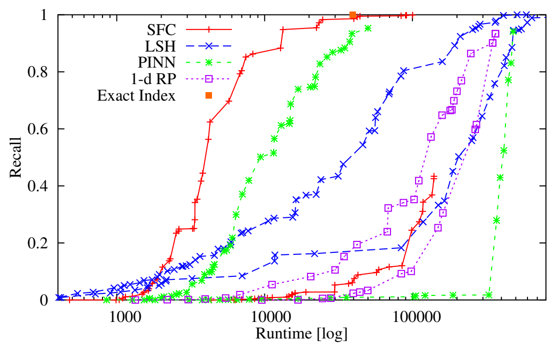
In Figure 8.4c we give a sample of the full parameter space we explored. Obviously, all 4000 runs will be an unreadable cloud of points, but instead we filter the results to LSH and one variant of SFC only (random curve families and random subspaces), which are about 700 runs. To show the continuity of the explored parameter space, we connected similar parameterizations with a line, more specifically for SFC indexes we connected those that differ only in window width (i.e. using $w \cdot k$ candidates) and for LSH we connect those that vary the number of hash tables l. For the exact indexes, the different results are for different page sizes. Apart from seeing that we systematically explored the parameter space, it is interesting to see that with SFC we repeatedly were able to get higher outlier detection quality at a more than 10-fold speedup over the exact indexes – only when using a *single* space-filling curve, the results were not substantially better.

Most of the parameters are expected to increase the number of candidates and thus the approximation quality of the index, but also the overall runtime, which is why this kind of smooth lines arise in visualization. For PINN we only include variants of the backing index in runtime experiments, as they do not affect recall.

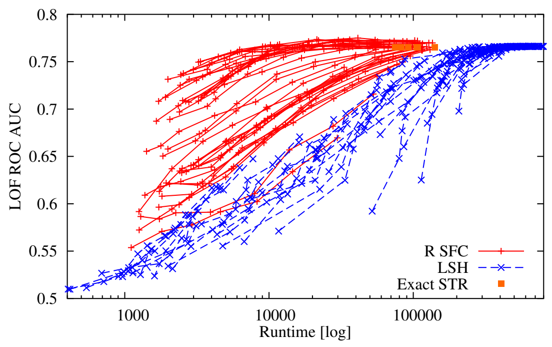
Figure 8.5b is the same sample as Figure 8.4c, but projected to LOF ROC AUC quality and recall. One would naïvely expect that a low recall implies that the method cannot work well. While algorithm performance and recall are correlated for locality sensitive hashing, the SFC approach violates this intuition: even with very low recall, it already works surprisingly well; and some of the best results have a recall of only around $.25$ – and outperform the exact solution. When looking at the skylines of the complete data in Figure 8.5c, this even yields an upper skyline that ends at a recall of 0.20 – no result with a higher recall performed better than this.



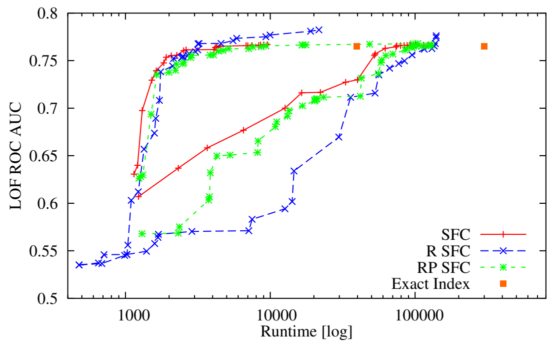
(a) Skyline results by basic methodology



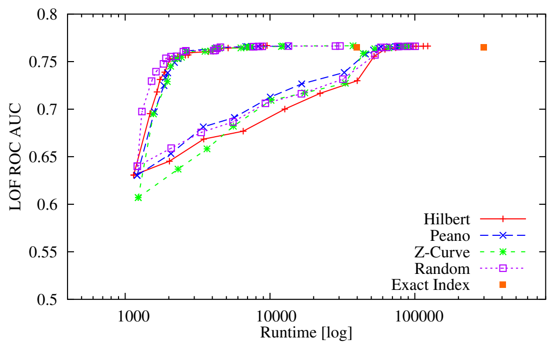
(b) Skyline results for recall of true 20 NN



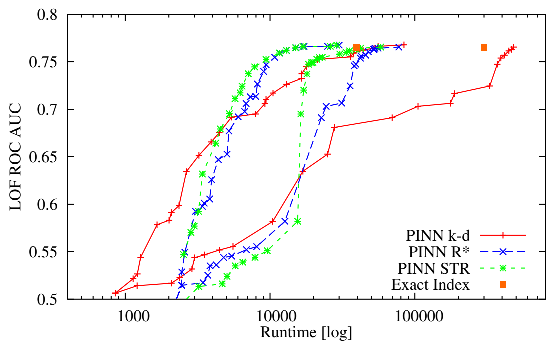
(c) Sample of results: random subspace SFC vs. LSH



(d) Skyline results by SFC projection

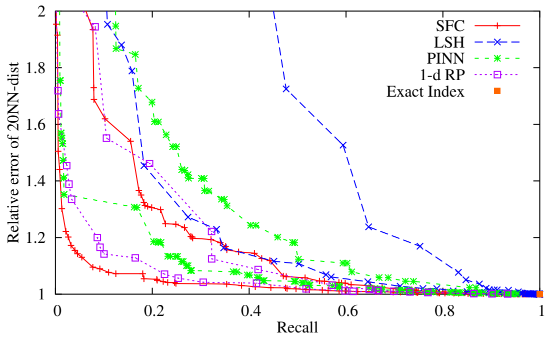


(e) Skyline results by SFC family

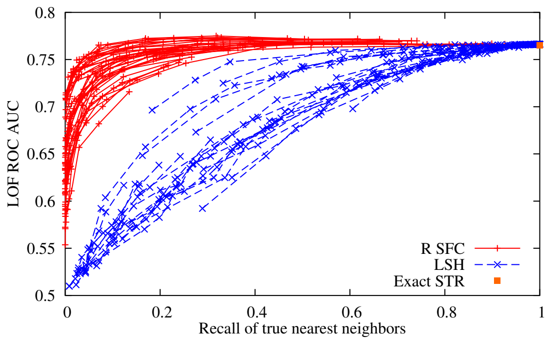


(f) Skyline results by PINN projection

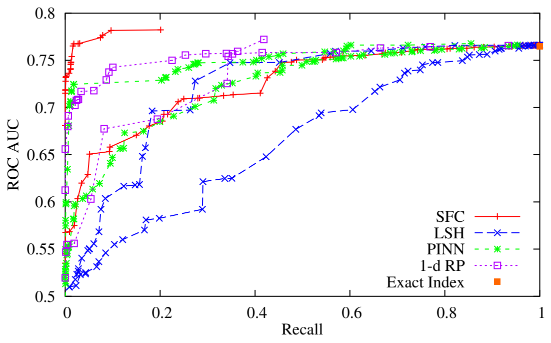
Figure 8.4: Experiments on 27d ALOI data set



(a) Relative 20-dist error, compared to recall.



(b) Sample of results: Recall of 20NN vs. AUC.



(c) Skyline results for AUC vs. Recall

Figure 8.5: Experiments on 27d ALOI data set

8.3 Indexing Geodetic Data

The contents of this section are based upon:

E. Schubert, A. Zimek, and H.-P. Kriegel. “Geodetic Distance Queries on R-Trees for Indexing Geographic Data”. In: *Proceedings of the 13th International Symposium on Spatial and Temporal Databases (SSTD), Munich, Germany.* 2013, pp. 146–164. DOI: 10.1007/978-3-642-40235-7_9

Maps have played an important role in human live for thousands of years. The oldest known maps are over 8000 years old. One of the breakthrough achievements in mapping is due to Ptolemaíos (around 100 A.D.), who essentially introduced the spherical coordinate systems we still use today with latitude and longitude (although using a different prime meridian).

While the spherical shape of the earth has been widely known since the ancient greek and was the dominant belief in the middle ages,[4] we still mostly rely on flat maps. While a globe is a good representation of the complete Earth; flat maps are more useful to us for local navigation and orientation. Yet, when analyzing such data, we must not assume that Euclidean distance in the coordinate system of latitude and longitude is a reasonable measure of distance. In the flat coordinate system, Alaska is the easternmost (as well as the northernmost and westernmost) state in the United States, since the Aleutian islands cross the $180°$ longitude line.

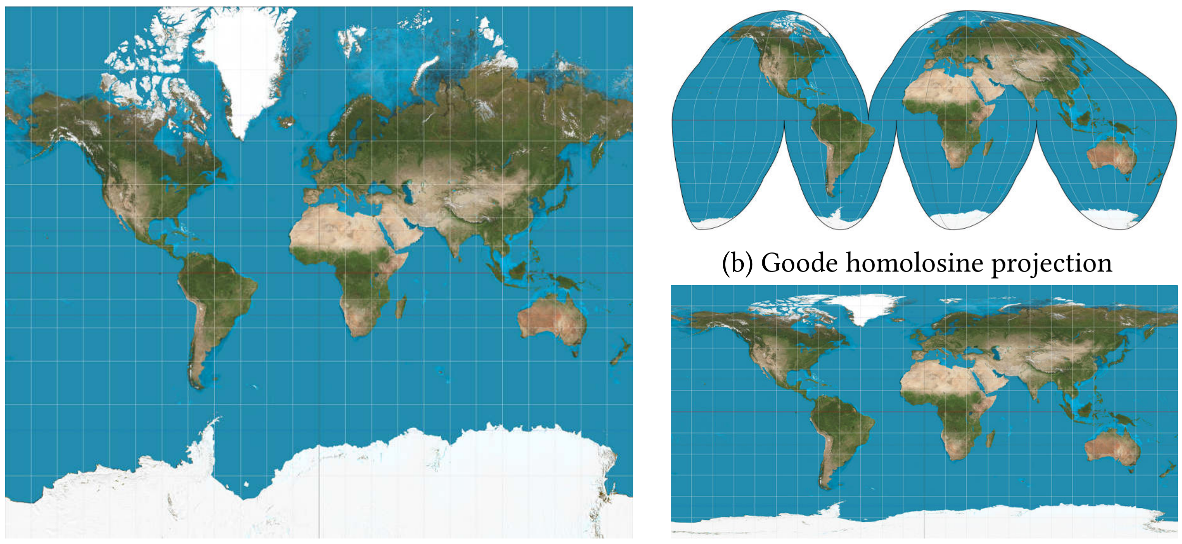
“ I wanna hang a map of the world in my house. Then I’m gonna put pins into all the locations that I’ve traveled to. But first, I’m gonna have to travel to the top two corners of the map so it won’t fall down. — Mitch Hedberg, according to [KW11] ”

In order to avoid bias and error in our data analysis, we need to pay attention to using the appropriate distance computations when handling such data. In this section, we will discuss some of these effects and artifacts, and introduce a function to be able to query R*-tree indexes for geographic data without introducing such errors. Since R*-trees work very well in low-dimensional situations, this will yield substantial speedup for many data mining tasks when processing geographic data. In particular, this method allows us to accelerate k-nearest neighbor search, a query that is used heavily for outlier detection methods.

8.3.1 Introduction

Nowadays, we are much more used to seeing maps than using a globe. But if we look at flat maps of the world, all map projections have some error: they cannot preserve the three components of geographical information, distance, area, and angles, equally well. The projections that we are most used to are the Mercator projection and the even simpler equi-rectangular projection. Google Maps, for example, is based upon a modified Mercator projection. Figure 8.6

[4]It is a myth that until Columbus people believed in a flat Earth, with numerous counterexamples.


(b) Goode homolosine projection

(a) Mercator projection

(c) Equirectangular projection

Figure 8.6: Three different projections of the earth surface.
Images from Wikimedia Commons, by user “Strebe”.[5]

shows three different projections used for maps. Figure 8.6a is a variation of the usual Mercator projection, which has a huge influence on our understanding of the world. Figure 8.6b is an alternate projection developed 1923 by John Paul Goode, which is an equal-area projection. The difference in the ability to preserve area can in particular be seen for the Antarctica, but also Greenland, Canada, Alaska and Russia show massive area distortions in the Mercator projection. The Mercator projection is obtained by wrapping a cylinder around the earth and projecting the earth surface onto this cylinder. This yields a good map where the cylinder touches or intersects the earth, but along the axis of the cylinder the distortion is infinite – the north and south poles do not project to the cylinder at all. Therefore, Mercator maps are commonly truncated at the poles. Figure 8.6a for example is truncated to a latitude of $\pm 82°$. While the Mercator projection does not preserve area and distances, it preserves the shapes quite well – but even more importantly, it preserves angles, which makes it useful for navigation and this probably is the main reason for the popularity of this projection. The equirectangular projection (Figure 8.6c) is probably the simplest projection, where latitude and longitude translate directly into y and x. In contrast to Mercator, it does not preserve angles. However, being able to trivially translate latitude and longitude to pixels on this map projection makes it a popular choice in GIS raster applications, and this is the default projection for placing custom texture overlays on the earth surface, e.g. in Google Earth and NASA WorldWind.

[5]For detailed copyright information on these images (CC-BY-SA 3.0), see `https://commons.wikimedia.org/wiki/Category:Images_of_map_projections`.

There are hundreds of geodetic datums,[6] some of which are historic, but many are still in use for good reasons, e.g. for land survey. We tend to assume that any geographic position is well-defined, but for example due to tectonic plate shift we have to accept the fact that even the largest mountains move with respect to each other over time. One of the most popular geographic coordinate system is that of geographic latitude and longitude, with respect to the World Geodetic System 1984 (WGS84) reference ellipsoid. Geographic position is then measured with three components, known as latitude, longitude, and elevation. Elevation is commonly given with respect to the reference ellipsoid's (or the more complicated geoid's) surface, so that values of 0 are approximately at sea level. In the following, we will use λ to denote latitude and ϕ to denote longitude. We will not be using elevation, because it usually is not available in many data sets. Furthermore, in order to fully specify position, one would also need to take the time of the measurement into account. For all these reasons, we have to live with some error in geo positioning and, thus, distance computations. There exist several reference models of the earth. The simplest is that of a sphere with radius $r = 6371$ km, but there exist more complex models such as the GRS80 (Geodetic Reference System 1980) and the WGS84 ellipsoids, the latter is commonly used with the global positioning system GPS.

In many applications, Euclidean distance is commonly chosen to measure distances on such data. If the variables stored are latitude and longitude, this equals measuring distances in the plate carrée (equirectangular) projection – which preserves neither distances nor area nor angles. The errors resulting from this are considerably larger than one might assume, even at short distances: For example at $45°N$, the latitude of Minneapolis, Turin and Bordeaux, $1°$ of longitude equals, in distance, approximately $0.707°$ of latitude (traveling $1°$ north is approximately 111 km, traveling $1°$ east is just 78.7 km). Therefore, this naïve approach leads to a non-negligible distortion for nearest neighbor and radius queries in large parts of Europe and the U.S. Transforming the data to a different *local* geodetic system can reduce this error significantly if we need a small part of the world only, but this approach does not work for global data sets: local geodetic systems are often based on transversal Mercartor projections and do not cover $360°$ of longitude.

A much more adequate choice for computing distances is the great-circle distance, also known as orthodromic distance and called geodetic distance when using the Earth's radius. This is the shortest distance on the *surface* of a sphere (or ellipsoid). In the simpler case of a sphere of radius r – not using an ellipsoid earth model – it can be computed using $\Delta\lambda := |\lambda_1 - \lambda_2|$ and the spherical law of cosines:

$$d_{\text{cosine}}(\phi_1, \lambda_1, \phi_2, \lambda_2) := r \arccos\left(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos(\Delta\lambda)\right)$$

Unfortunately, this formula has low numerical precision for small distances, so improved formulas such as the haversine (from half versed sine, $\operatorname{haversin}(\theta) = \sin(\frac{\theta}{2})^2$) formula [Sin84]

[6] A geodetic datum defines a specific geodetic system in terms of a coordinate system and reference points. The correct plural is datums, not data.

have been developed since:

$$\begin{aligned} d_{\text{haversine}} :=& 2r \arcsin \sqrt{(\sin \frac{\phi_2 - \phi_1}{2})^2 + \cos\phi_1 \cos\phi_2 (\sin \frac{\Delta\lambda}{2})^2} \\ \equiv& 2r \arctan \frac{\sqrt{(\sin \frac{\phi_2 - \phi_1}{2})^2 + \cos\phi_1 \cos\phi_2 (\sin \frac{\Delta\lambda}{2})^2}}{\sqrt{1 - \left((\sin \frac{\phi_2 - \phi_1}{2})^2 + \cos\phi_1 \cos\phi_2 (\sin \frac{\Delta\lambda}{2})^2\right)}} \end{aligned}$$

The second form of the haversine formula has slight computational advantages over the first by computing it using $\operatorname{atan2}(\sqrt{a}, \sqrt{1-a})$.

Vincenty's formula [Vin75] is an iterative method to compute the great circle distance in an ellipsoidal earth model such as WGS84. For the – simpler – spherical model, this formula would be as follows:

$$d_{\text{vincenty}} := r \arctan \frac{\sqrt{(\cos\phi_2 \sin\Delta\lambda)^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\Delta\lambda)^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda}$$

Additionally to computing distances between points, one could also want to compute distances between points and a desired track: the so called "cross-track distance" (XTD, also "cross track error") is the sideways deviation from a desired path of travel, for example due to wind affecting an airplane. In the simpler, spherical, model, it can be computed as

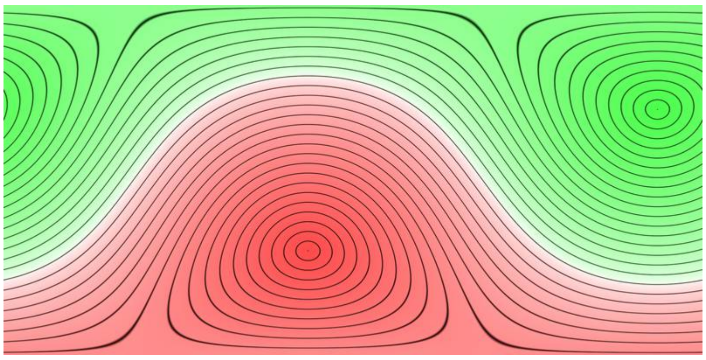
$$d_{\text{xtd}} := r \arcsin\big(\sin(d_{sp}/r) \cdot \sin(\theta_{sp} - \theta_{sd})\big), \tag{8.1}$$

where d_{sp} is the distance from the starting point s to the current position p, θ_{sp} is the initial bearing from the starting point s to the current position p, and θ_{sd} is the initial bearing from the starting point s to the desired destination d.

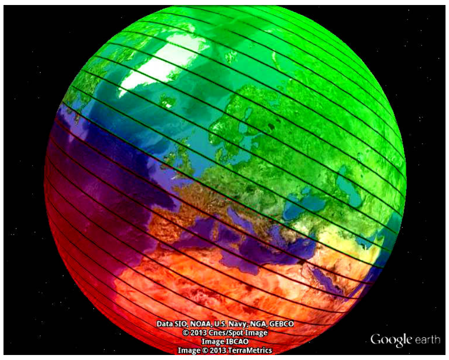
The bearing (or forward azimuth) is the initial direction one needs to travel to the destination on the great-circle path (i.e., on the shortest path). Note that, when traveling along this path, the measured direction will actually change, so it does matter at which point we compute the bearing. The spherical formula for the bearing from a to b is:

$$\theta_{ab} = \arctan \frac{\sin(\Delta\lambda)\cos\phi_b}{\cos\phi_a \sin\phi_b - \sin\phi_a \cos\phi_b \cos(\Delta\lambda)} \tag{8.2}$$

Cross-Track-Distance and its orthogonal component, the along-track-distance (which we will actually not use in the index computations) are visualized in Figure 8.7 and Figure 8.8 for a track from Munich to New York City. The left images (Figure 8.7a and 8.8a) are the equirectangular projections and may appear distorted. But when the map is projected onto the sphere, as seen in Figure 8.7b and 8.8b, it appears regular as expected. Red colors indicate a deviation to the left (in Figure 8.7) or a position along the track before the starting point (in Figure 8.8), which are often represented by negative numbers. Black lines indicate contours of $1/36$ the earth circumference.

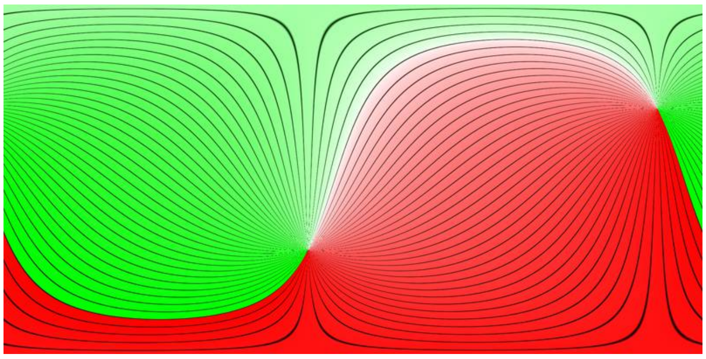


(a) Equirectangular projection

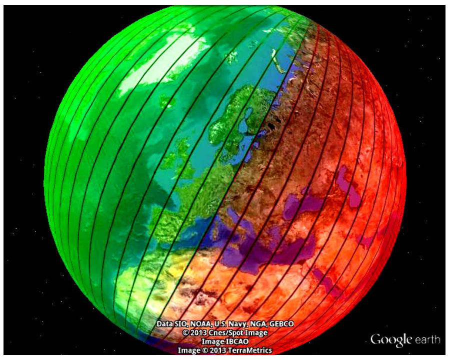

Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2013 Cnes/Spot Image
Image IBCAO
Image © 2013 TerraMetrics
Google earth

(b) Google Earth projection

Figure 8.7: Cross-Track-Distance from the course Munich to New York.



(a) Equirectangular projection


Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2013 Cnes/Spot Image
Image IBCAO
Image © 2013 TerraMetrics
Google earth

(b) Google Earth projection

Figure 8.8: Along-Track-Distance from the course Munich to New York.

8.3.2 Related Work

Considering the background, state-of-the-art, and open problems, we discuss three aspects: first, data indexing methods proposed in scientific literature; second, the functionality commonly available in actual database engines; third, more specifically the problems of geodetic data when combined with these indexes.

8.3.2.1 Data Indexing in Scientific Work

Among the simplest index structures for spatial data are quadtrees [FB74], that recursively split an overflowing cell into four parts (by splitting in the middle of the x and y axes) until the cells contain at most the desired number of objects. A similar idea is the base of k-d trees [Ben75], which split into two parts similar to a B-tree, but rotate through the d axes at each level. By splitting at the median (on an optimal tree) instead of the middle, a k-d tree can be a balanced tree. However, it does not allow easy dynamical rebalancing. Repeated insertion may cause the tree to become unbalanced and require the index to be rebuilt to retain good performance. Hierarchical Triangular Mesh [KST01] can be seen as an adoption of quadtrees to indexing the surface of a sphere. The initial approximation is an octahedron, and each triangle is then repeatedly decomposed into the four smaller triangles obtained by splitting each edge in half.

While quadtrees can be linearized and then stored efficiently in a B+-tree (which effectively turns the quadtree into a Z-curve [Mor66]), these indexes do not make native use of the block structure of harddisks. The R-tree [Gut84] and its variants (such as the R*-tree [Bec+90]) are very popular spatial index structures for the use in databases, since they offer three main benefits over k-d trees: first of all, they are designed with paged memory in mind, and thus can easily be implemented as disk-based indexes. Secondly, they also consider non-singular objects with a spatial extent of their own, such as polygons. But even more importantly, they allow for dynamical rebalancing when inserting and deleting data.

In this respect, the R*-tree [Bec+90] is an important extension of the R-tree when handling dynamic data, since its focus is on rebalancing and, this way, on optimizing the tree. The SS-tree [WJ96] is similar to the R-tree, but uses bounding spheres instead of rectangles for its bounding volume hierarchy. The M-tree [CPZ97] includes the distance to each child in the parent page and uses the triangle inequality to prune search candidates. Both do not rely on projections or coordinates, but index the data solely by their distances. On the other hand, this requires the tree being built for the specific distance function to be used for querying. It can not be used to answer queries with arbitrary other distance functions. While the query process is quite straightforward, the construction and incremental update of these trees is much harder than in the case of the coordinate-oriented R-trees. The SS+-tree [KJS97] uses k-means clustering to find a good split; however k-means only optimizes for (squared) Euclidean distance, and does not search for a minimum cover, but minimizes in-cluster variances. A bulk-loading strategy [CP98] for the M-tree is based on sampling and k-means style clustering to avoid having to compute all pairwise distances. The Slim-Tree [Tra+00; Tra+02] suggests the use of the minimum spanning tree to split nodes and uses R*-tree-inspired reorganization techniques to "slim down" the tree. These difficulties of efficiently computing a good split constitute the largest drawback of the M-tree. There are also hybrid techniques, such as an R-tree which also stores the center and radius of the page (SR-Tree [KS97]). Here, the split strategies of the R-tree can be used, and the covering radius serves as an additional pruning heuristic. Again, this radius can be only used for a specific distance function, chosen at index construction time.

8.3.2.2 Data Indexing in Practical Use

Not all of the techniques discussed above are used in practice. The most used techniques probably are Quadtrees (because of their simplicity and the ability to map them to existing B+-tree indices for harddisk storage) and R*-trees. However, while many database engines (including but not limited to Oracle, IBM Informix, Microsoft SQL Server, PostgreSQL, SQLite) list these indices on their feature sheets, the actual support for using them in query evaluation varies. The main functionality that seems to be widely supported is that of multidimensional range queries, which is for example useful when displaying parts of a map. Few seem to allow index-accelerated distance-based queries, and it is even more unclear, if they do, which distance functions are supported. Most database engines support only Euclidean distance queries.

Microsoft SQL Server uses a multi-level grid-based approach closely related to Quadtrees that requires filter refinement that can (since SQL Server 2012) also accelerate nearest neighbor queries [Mic; Fan+08] (at least for Euclidean distance). PostGIS/PostgreSQL have support for R-trees and M-trees implemented on top of their GiST architecture. However, these indexes can currently only be used with bounding box-based operators and not with queries that use Euclidean `ST_Distance` [Pos], yet the spherical `ST_Distance_Sphere`. Built-in operators of the database such as <#> and <-> also appear to support Euclidean distances only. IBM Informix [IBM] uses a data partitioning scheme with a predefined set of Voronoi cells (see also [Luk87]), based on the population density of the areas. Furthermore, it supports the R-tree with custom strategy functions via an API. It is not clear from the documentation whether the nearest neighbor search functions are provided for any of the predefined geodetic data types. However, the algorithms presented in this Section can be implemented straightforwardly in this API. Oracle Spatial supports quadtrees and R-trees. Only the latter can be used for geodetic distance queries. An empirical evaluation at Oracle found R-trees to "consistently outperform quadtrees by a factor of 3" [KRA02] for distance queries on the GIS data used in the study and to offer equivalent or better performance in almost all cases without parameter tuning. According to [HRA11], the approach used by Oracle is to project the data into a 3-dimensional, "earth-centered-earth-fixed" (ECEF) coordinate system space and index the data there, but details on the exact method used do not seem to be available.

8.3.2.3 Handling Geodetic Data with Non-Geodetic Indexes

As we have seen, many popular index structures are either designed or implemented only with Euclidean distance in mind. When geodetic data are naïvely stored in such an index, it will result in errors in distance computations. In Section 8.3.1 we already noted that the error at $45°N$, the latitude of Minneapolis, Turin and Bordeaux, the same Euclidean distance going east is just 70.7% of that when going south instead. Therefore, we should not use Euclidean distance with data in the non-Cartesian coordinate system of latitude and longitude. Many widely available index structures will only support such distance functions.

In order to get a more reasonable precision using Euclidean distance, the data must be transformed into a locally equidistant projection such as European Datum 1987 (ED87) or the European Terrestrial Reference System (ETRS) for data in Europe. Close to the fundamental point of these coordinate systems, the Euclidean distance can be reasonably used without large distortion. However, on a global data set there exists no optimal fundamental point, and distortions will occur when using the Euclidean distance.

8.3.2.4 Summary and Contributions

Indexing of spatial data is not a new domain. Many methods have been around for a long time. A method that clearly has proved itself in the “test of time” is the R*-tree, which seems to be used by every major database engine. Yet, when processing geodetic data, the index support in common database engines is often surprisingly limited. The R*-tree is – in contrast to the M-tree – actually distance agnostic and by no means limited to the Euclidean distance or L_p-norms. The R*-tree implementations in ELKI [Ach+12] supports various distance functions, including Canberra distance, Histogram intersection distance, and Cosine similarity as long as the data does not contain the origin. In the following we will introduce two approaches to index data with respect to the geodetic distance based on R-trees. Both approaches have strengths and weaknesses, some of which we will look at in detail.

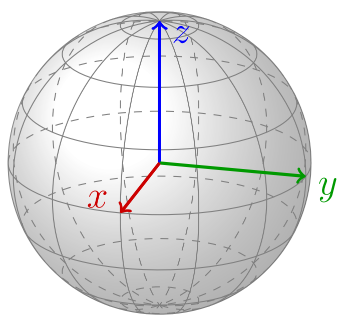
For the simpler approach (which likely is similar to what Oracle Spatial uses [HRA11; Hu+12]), we can project the data into a 3-dimensional ECEF coordinate system and use an R-tree or M-tree with Euclidean distance. The Euclidean distance then is a lower bound for the geodetic distance, and we can therefore get a good approximation of the query set, which then can be refined with geodetic distance. The main drawback of this approach is that the resulting tree is less useful when querying map sections based on a latitude/longitude rectangle; it also needs additional memory to index the data this way.

The second approach we introduce will work on the unmodified data (i.e. latitude and longitude coordinates) and an unmodified R-tree. What is needed to enable this kind of queries is the minimum distance from a query point to an index rectangle, which is a key contribution of this Chapter. The main benefits of this approach are that the same index can be used to answer typical map window queries that consist of latitude and longitude ranges, allowing for a dual-use index. Furthermore, since the index is using the original, 2D data, it requires less memory (and thus less I/O) than the other approach. The drawback is that, since it involves trigonometric computations, it is more CPU intensive.

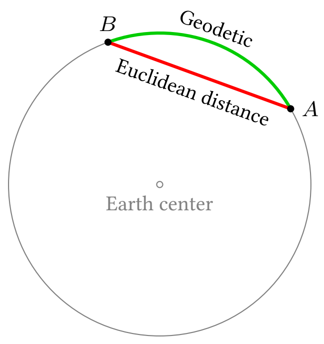
Both approaches can easily be implemented in any existing database that already supports the R-tree: there are no changes needed to the index or index construction. In the first approach the index only needs to be able to accelerate 3D Euclidean distance queries. In the second approach it needs to allow for custom distance functions.

8.3.3 Indexing Geodetic Data

Some index structures such as the M-tree can be used immediately with any metric distance function. Being the shortest path on the surface, the great-circle distance is a proper metric, and can be used with the M-tree immediately. We will discuss two alternate approaches here.



(a) ECEF coordinate system: x is to $0°N, 0°E$; y is to $0°N, 90°E$; z to $90°N$



(b) Euclidean distance is a lower bound for geodetic (great-circle) distance

Figure 8.9: ECEF Cartesian index space.

8.3.3.1 Indexing Geodetic Data Using 3D Euclidean Coordinates

While the geographic latitude and longitude are probably the most popular datum for geographic positions, there exist other coordinate systems. One of these sticks out because it actually uses Cartesian coordinates – which is also the main drawback, because it is not at all map oriented. Instead of giving coordinates on the earth surface, it uses three axes originating from the Earth's centre of mass. The x and y axes span the equatorial plane, with the x-axis pointing to the prime meridian, the y-axis pointing to $90°E$, the z-axis pointing straight north, i.e. it coincides with the average rotational axis of the Earth. Figure 8.9a visualizes the axes with respect to the earth sphere. This coordinate system can be referred to as earth-centered earth-fixed (ECEF) or simply “XYZ” coordinate system. The name originates from the coordinate system being centered on the earth mass point and fixed to be invariant to the earths rotation.

While Euclidean distances in this coordinate system do not follow the Earth's surface, they have an interesting property. As sketched in Figure 8.9b, the Euclidean distance is the chord length in the great circle used by the geodetic distance. This yields two important properties:

- Euclidean distance in the ECEF coordinate system is a *lower bound* for the geodetic distance:

$$L_{2,\text{ECEF}}(a,b) \leq d_{\text{geodetic}}(a,b) \tag{8.3}$$

- Euclidean distance in the ECEF coordinate system is *strictly monotone* to geodetic distances, i.e.

$$L_{2,\text{ECEF}}(a,b) < L_{2,\text{ECEF}}(x,y) \Leftrightarrow d_{\text{geodetic}}(a,b) < d_{\text{geodetic}}(x,y) \tag{8.4}$$

The first property guarantees that for a query radius r, all objects (although also some more) are found that would be in the desired range for the geodetic distance. The second property gives an even stronger guarantee for retrieving the k nearest neighbors: here, no additional objects should be included (notwithstanding numerical issues).

Therefore any index that can support Euclidean distance in 3 dimensions can be used to index geodetic data, after transformation into ECEF Cartesian coordinates. This includes (but is not limited to) gridfiles, octrees, the M-tree, and the R-tree family. For our experiments, we will focus on the M-tree and R-trees. This approach is probably not novel. In [HRA11; Hu+12] the authors mention that Oracle Spatial uses a 3D R-tree to index geodetic data, but without giving further details or properties – it might as well be a 3D R-tree on latitude, longitude and elevation. IBM Informix documentation also mentions 3D bounding boxes for geodetic data, but only mentions intersection queries. Therefore it is not clear if above properties have been realized and are used yet. The PostgreSQL pgSphere project seems to include this transformation, but does not appear to make use of it for indexing yet.

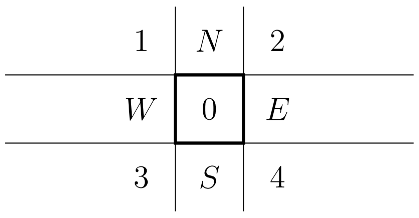
8.3.3.2 Indexing Geodetic Data Using 2D Geodetic Coordinates

The alternative approach we introduce here is designed with the R-tree in mind, although the obtained equations could also prove useful with other indexes such as grid-files, quadtrees, and VA-File [WSB98] indexes. A key benefit of this approach is that it can use a regular R-tree [Gut84], R*-tree [Bec+90], or any of its many variants, as index *without modifications* to the actual index structure. In particular, the index is built on the latitude and longitude coordinates, and can therefore be used for window queries that frequently arise in map applications. Similarly, the search can trivially be bounded with such a window. In contrast to the M-tree [CPZ97], which needs to be built for a specific distance function, the R-tree family of indexes are unspecific, but index the coordinates using bounding boxes. Since the same tree can be used with very different distance functions, R-trees can be considered general purpose spatial indexes, whereas M-trees are highly specific.

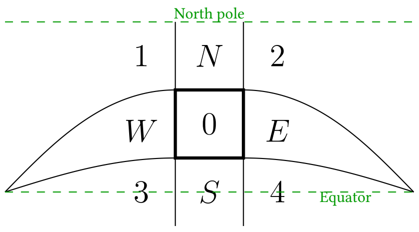
Each object as well as each index page in an R-tree is represented by a minimum bounding rectangle (MBR), which in this case means it is represented by a quadruple $(\lambda_{\min}, \phi_{\min}, \lambda_{\max}, \phi_{\max})$ (plus other attributes, if present; in the following we will assume to only index latitude and longitude). In order to query the R-tree, we need to compute a lower bound for the distance of the query point to an arbitrary – unknown – object within the given rectangle. For L_p-norms, this distance computation is very efficient, which makes the R-tree attractive to use. But also for other – even some non-metric – distances, such a lower bound can be specified. For L_p-norms, the minimum distance can be computed using simple case distinctions in each dimension:

$$\text{mindist}_{L_p}(o, \text{MBR}) := \left(\sum_i \begin{cases} (\min(\text{MBR}, i) - o_i)^p & \text{if } o_i < \min(\text{MBR}, i) \\ (o_i - \max(\text{MBR}, i))^p & \text{if } o_i > \max(\text{MBR}, i) \\ 0 & \text{otherwise} \end{cases} \right)^{1/p}$$

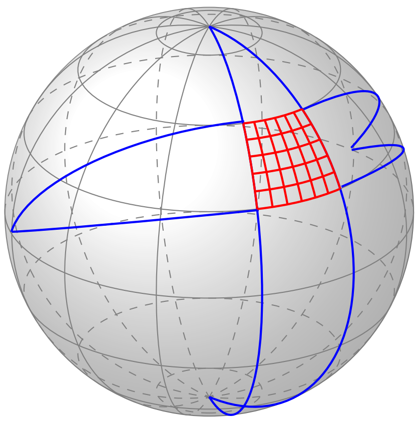
Unfortunately, in geodetic data, the formula will become more complicated. This is largely due to the fact that an MBR in the equirectangular projection – which does not preserve distances



(a) Case distinction in Euclidean data



(b) Case distinction in geodetic data



(c) Spherical interpretation

Figure 8.10: Case distinctions for point-to-MBR distance in geodetic data.

– when projected to the surface of the earth yields a much more complex shape. However, to compute the minimum distance, we will still need to distinguish the same 3×3 cases, just as for the L_p-norms. The 3×3 case distinction in 2-dimensional Euclidean space is shown in Figure 8.10a: if the query point is inside the rectangle – area 0 – the minimum distance will be 0. In the areas N, E, S and W, the shortest path is along the normal vector to the closest edge, while in the corner areas $1 \ldots 4$, the shortest path is to the nearest corner of the MBR. The transfer of this model to geodetic data is shown in Figure 8.10b for an example on the northern hemisphere. The north and south edges of the rectangle are parallels of the equator, while the east and west edges are meridians. Note that the north and south poles in the equirectangular projection are not a single point, but actually the complete northern and southern edges of the projected map. At first, the situation appears to be highly asymmetric. However this largely is an artifact of the geographic coordinate system and the equirectangular projection, in which great-circle paths are only straight lines if they are meridians or the equator, and curves of the type $y = \text{arctan}_{360}(a \cdot \sin(x - x_0))$ otherwise. While it is not evident from the 2D projection of the case distinction (Figure 8.10b), the N and S areas are actually also triangular, since the north and south edge of the projection represent a single point each. When mapped onto a sphere, the regions look approximately similar, as visualized in Figure 8.10c. Note that the four triangles (N, S vs. W, E) nevertheless do *not* have the same mathematical properties: only the east and west edges of the MBR are on great-circles, while the north and south edges are lines of constant latitude. So from the point of view of spherical geometry, the north and south triangles have one bent edge each.

Fortunately, both the test and the distance computation for points in areas N and S remains as simple as for Euclidean distance – the shortest great-circle path to the rectangle is a meridian. In order to distinguish the other cases, we first need to test whether we are on the left or on the right side by rotating the mean longitude of the rectangle by $180°$ – the meridian opposite of the rectangle. The key to distinguishing the cases 1, W and 3 (and, identically, 2, E, 4) then is

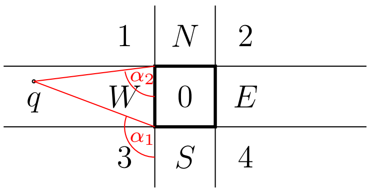


Figure 8.11: Angle-based test in Euclidean space

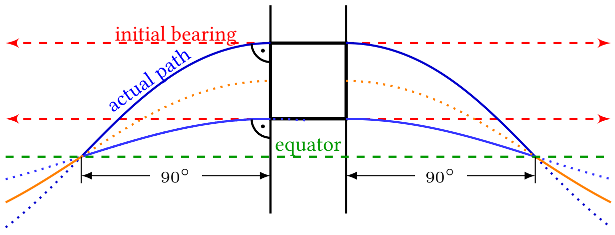


Figure 8.12: Case distinctions for point-to-MBR distance in geodetic data, detail.

the *azimuth* (north-based, also referred to as bearing) from the two corners towards the query point. The azimuth plays roughly the role of the angle in Euclidean geometry. In order to test whether a point is in area W in Euclidean space, we can compute the angle at the north-west and south-west corners of the MBR. This idea is sketched in Figure 8.11: only if the angles at the south-west α_1 is larger than $90°$ and the angle at the north-west corner α_2 is less than $90°$, then the query point is in area W. By substituting the azimuth for the angle, we can perform the same test in the spherical domain:

A shortest path to the west edge of the MBR must be a great circle path that arrives at an azimuth of $90°$ to the edge. If and only if the azimuth at the south corner is larger than $90°$ and the azimuth at the north corner is smaller than $90°$, then there exists a point on the meridian in between where the course is exactly $90°$. The difference between Euclidean space and spherical geometry shows when we travel a path that started at an initial bearing of $90°$: it will not be a straight line in the equirectangular projection. This is visualized in Figure 8.12: with an initial bearing of $270°$ (to north, $90°$ with respect to the south pole!) – indicated by the red lines – the blue curves are obtained. Conversely, for points on the blue lines, an initial bearing of $270°$ is obtained for one of the corners.

Algorithm 4 uses this idea to compute the minimum distance from the query point to an MBR by using the azimuth for case distinction. Note that for practical use, this pseudocode should be optimized by inlining the great-circle and cross-track distances in order to share all redundant trigonometric computations. When implementing this in a database, the number of trigonometric computations must be kept low as they are rather expensive. However, we can further improve this algorithm: we do not need the exact values of the azimuth, but we only need to know whether it will be smaller or larger than $90°$. If we can compute the blue lines in Figure 8.12 directly, we can easily test whether a point is between the two blue lines. As noted before, each great-circle (that is not a meridian) can be expressed as $\lambda = \arctan_{360}(a \cdot \sin_{360}(\phi - \phi_0))$. If

Algorithm 4: Min. Dist. Point to MBR by Azimuth (non-optimized).

```
Data: $c$ circumference of earth (spherical model)
Data: $(\phi_q, \lambda_q)$ query point
Data: $(\phi_l, \lambda_l, \phi_h, \lambda_h)$ index MBR
if $\phi_l \leq \phi_q \leq \phi_l$ then
    if $\lambda_q \leq \lambda_l$ then return $c \cdot (\lambda_l - \lambda_q)/360°$ ;          /* South of MBR */
    if $\lambda_q \geq \lambda_t$ then return $c \cdot (\lambda_q - \lambda_t)/360°$ ;          /* North of MBR */
    return 0;                                                                  /* Inside MBR */
else if $mod_{360}(\phi_l - \phi_q) \leq mod_{360}(\phi_q - \phi_h)$ then                     /* West of MBR */
    $\theta_h \leftarrow$ Azimuth$(\phi_l, \lambda_h, \phi_q, \lambda_q)$;
    if $\theta_h \geq 270°$ then return Great-Circle-Distance$(\phi_l, \lambda_h, \phi_q, \lambda_q)$ ;   /* North-West */
    $\theta_l \leftarrow$ Azimuth$(\phi_l, \lambda_l, \phi_q, \lambda_q)$;
    if $\theta_l \leq 270°$ then return Great-Circle-Distance$(\phi_l, \lambda_l, \phi_q, \lambda_q)$ ;   /* South-West */
    return |Cross-Track-Distance$(\phi_l, \lambda_l, \phi_l, \lambda_h, \phi_q, \lambda_q)$| ;             /* West */
else                                                                           /* East of MBR */
    $\theta_h \leftarrow$ Azimuth$(\phi_h, \lambda_h, \phi_q, \lambda_q)$;
    if $\theta_h \leq 90°$ then return Great-Circle-Distance$(\phi_h, \lambda_h, \phi_q, \lambda_q)$ ;    /* North-East */
    $\theta_l \leftarrow$ Azimuth$(\phi_h, \lambda_l, \phi_q, \lambda_q)$;
    if $\theta_l \geq 90°$ then return Great-Circle-Distance$(\phi_h, \lambda_l, \phi_q, \lambda_q)$ ;    /* South-East */
    return |Cross-Track-Distance$(\phi_h, \lambda_h, \phi_h, \lambda_l, \phi_q, \lambda_q)$| ;             /* East */
end
```

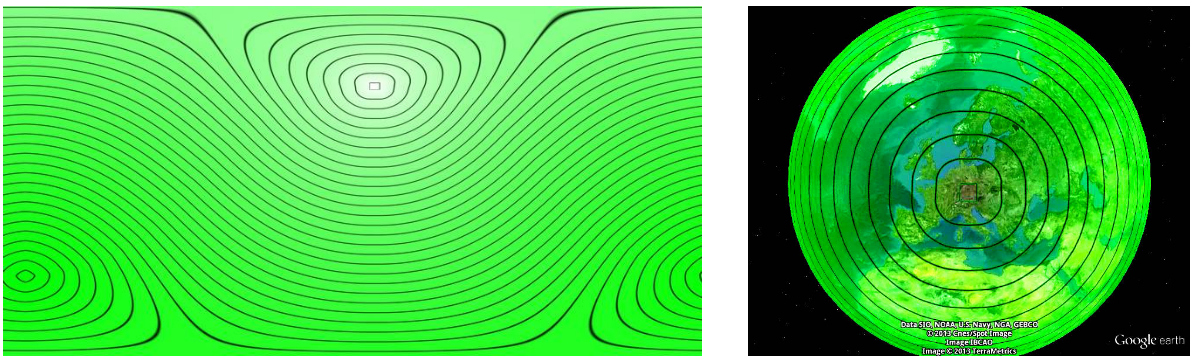
we know the parameters ϕ_0 and a we can easily test whether a point is north or south of these lines. ϕ_0 is the longitude where the great-circle crosses the equator, and the maximum longitude is achieved when $\phi - \phi_0 = 90°$, with $\lambda = \arctan_{360}(a)$. Since we want the curves to be orthogonal to the meridian, this is where they must have a maximum or minimum. Therefore, we can choose $\phi_0 = \phi_r - 90$ and $a = \tan_{360}(\lambda_r)$ for a given reference point r. A point q is south of the great-circle path that goes orthogonally to the meridian through (ϕ_r, λ_r) if

$$\lambda_q < \arctan_{360}(\tan_{360}(\lambda_r) \cdot \sin_{360}(\phi_q - \phi_r + 90))$$

or, equivalently,

$$\tan_{360}(\lambda_q) < \tan_{360}(\lambda_r) \cdot \cos_{360}(\phi_r - \phi_q),$$

in which we can reuse $\tan_{360}(\lambda_r)$ for the second test and preserve numerical precision slightly better. Since this test is faster to compute (since it involves fewer trigonometric functions), it allows for a further optimized version of the algorithm, which is given in Algorithm 5. For points close to the rectangle, we can just test whether they are above the great-circle through the upper corner of the MBR, below the great-circle through the lower corner, or in between. However, these two lines will intersect when crossing the equator at $\Delta\phi = 90°$ from the MBRs edge. Starting at this distance, we will instead look at the great-circle through the middle of the MBR, and use this for distinguishing the remaining two cases: at this distance we know that one of the two corners must be closest.


Data SIO, NOAA, U.S. Navy, NGA, GEBCO
© 2013 Cnes/Spot Image
Image IBCAO
Image © 2013 TerraMetrics
Google earth

(a) Equirectangular projection (b) Google Earth projection

Figure 8.13: Minimum distance from a bounding box around Bavaria.

Figure 8.13 visualizes the minimum distance from a bounding box around Bavaria. Again, in the equirectangular projection in Figure 8.13a, it appears to be irregular, but projected onto the sphere in Figure 8.13b the rectangles with increasingly rounded corners become visible.

8.3.3.3 Non-spherical Earth Models

The equations we presented were so far all for a spherical earth model for simplicity. For geographic data it is however a best practice to use for example a spheroid model such as WGS84. For some of the equations, formulas for the spheroid model are readily available, for others they are not. For our use case of finding a *lower* bound of the point to MBR distance, the most efficient approach is to use the spherical model with a reduced earth radius of the polar radius b. This usually has a negligible impact on the pruning power of the tree, and this way easily outweighs the more complex computations needed for the spherical model. The actual point to point distances are then computed with Vincenty's formula on the spheroid model.

8.3.4 Experiments

8.3.4.1 Data Sets

For our benchmarking experiments we use data from DBpedia 3.7, a parsed version of Wikipedia. From this data set we obtained 442775 points of interest around the world, and for 109577 of these we also obtained a region of interest which we use as query radius. The second data set we use is road accidents data set from the UK government, spanning the years 2005 to 2011 containing data on 1.2 million road accidents in the UK (also used in Section 9.2). The third data set consists of 6.3 million radiation measurements taken in Japan after the Fukushima nuclear disaster (also used in Section 9.3).[7]

[7]The data sets are publicly available and can be downloaded from `http://dbpedia.org/`, `http://data.gov.uk/` and `http://safecast.org/`.

Algorithm 5: Optimized Minimum Distance Point to MBR

```
Data: $c$ circumference of earth (spherical model)
Data: $(\phi_q, \lambda_q)$ query point
Data: $(\phi_l, \lambda_l, \phi_h, \lambda_h)$ index MBR
if $\phi_l \leq \phi_q \leq \phi_l$ then
    if $\lambda_q \leq \lambda_l$ then return $c \cdot (\lambda_l - \lambda_q)/360°$ ;                /* South of MBR */
    if $\lambda_q \geq \lambda_t$ then return $c \cdot (\lambda_q - \lambda_t)/360°$ ;                /* North of MBR */
    return 0;                                                                  /* Inside MBR */
else if $mod_{360}(\phi_l - \phi_q) \leq mod_{360}(\phi_q - \phi_h)$ then                       /* West of MBR */
    $\tau \leftarrow \tan_{360}(\lambda_q)$;
    if $mod_{360}(\phi_l - \phi_q) \geq 90°$ then                                      /* Large $\Delta\phi$ */
        if $\tau \leq \tan_{360}((\lambda_l + \lambda_h)/2) \cos_{360}(\phi_l - \phi_q)$ then
            return Great-Circle-Distance($\phi_l, \lambda_h, \phi_q, \lambda_q$) ;                /* North-West */
        else
            return Great-Circle-Distance($\phi_l, \lambda_l, \phi_q, \lambda_q$) ;                /* South-West */
        end
    end
    if $\tau \geq \tan_{360}(\lambda_h) \cos_{360}(\phi_l - \phi_q)$ then
        return Great-Circle-Distance($\phi_l, \lambda_h, \phi_q, \lambda_q$) ;                    /* North-West */
    if $\tau \leq \tan_{360}(\lambda_l) \cos_{360}(\phi_l - \phi_q)$ then
        return Great-Circle-Distance($\phi_l, \lambda_l, \phi_q, \lambda_q$) ;                    /* South-West */
    return $|$Cross-Track-Distance($\phi_l, \lambda_l, \phi_l, \lambda_h, \phi_q, \lambda_q$)$|$ ;                 /* West */
else                                                                           /* East of MBR */
    $\tau \leftarrow \tan_{360}(\lambda_q)$ ;
    if $mod_{360}(\phi_q - \phi_h) \geq 90°$ then                                      /* Large $\Delta\phi$ */
        if $\tau \leq \tan_{360}((\lambda_l + \lambda_h)/2) \cos_{360}(\phi_h - \phi_q)$ then
            return Great-Circle-Distance($\phi_h, \lambda_h, \phi_q, \lambda_q$) ;                /* North-East */
        else
            return Great-Circle-Distance($\phi_h, \lambda_l, \phi_q, \lambda_q$) ;                /* South-East */
        end
    end
    if $\tau \geq \tan_{360}(\lambda_h) \cos_{360}(\phi_h - \phi_q)$ then
        return Great-Circle-Distance($\phi_h, \lambda_h, \phi_q, \lambda_q$) ;                    /* North-East */
    if $\tau \leq \tan_{360}(\lambda_l) \cos_{360}(\phi_h - \phi_q)$ then
        return Great-Circle-Distance($\phi_h, \lambda_l, \phi_q, \lambda_q$) ;                    /* South-East */
    return $|$Cross-Track-Distance($\phi_h, \lambda_h, \phi_h, \lambda_l, \phi_q, \lambda_q$)$|$ ;                 /* East */
end
```

8.3.4.2 Efficiency

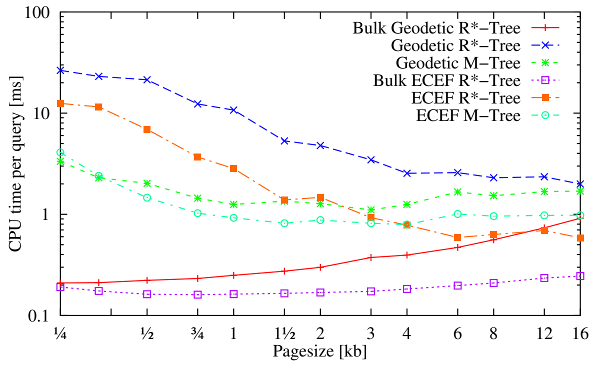
To study the behavior of the 2D- and the 3D-model in existing index structures, we use the R*-tree (both incrementally built and bulk loaded) and the M-tree (incrementally built only). Since the M-tree supports any metric distance function – and the geodetic distance is metric – it can be used with the geodetic distance. Figure 8.14 shows the results for 100 nearest neighbor and range queries on the DBpedia data set. From a mere CPU perspective (Figure 8.14a and Figure 8.14b), the 3-dimensional ECEF approach appears to be best, due to the rather costly trigonometric functions needed for the direct indexing approach. However, this may be misleading in an actual database context, since for larger data sets the input and output cost must be taken into account. And with this, the reduced memory requirements of the direct indexing approach pay off, which allow storing about 40% more objects per page. Figure 8.14c and Figure 8.14d show the I/O cost (in number of page accesses times page size, to make values across different page sizes more comparable) for querying the trees. The stronger pruning power of the 2D rectangles manifests itself in requiring fewer distance computations (Figure 8.14e and Figure 8.14f). Figure 8.14g shows the time needed to build the index, while Figure 8.14h visualizes the resulting index sizes. Note that the bulk loading actually has less work to do (fewer sorting passes) with larger page sizes, while for the incremental M-tree construction, which requires the computation of all pairwise distances, quickly becomes rather expensive as the page size increases.

Except for the expensive index construction in particular of the M-tree implementation we used, all indexes offered a significant performance improvements over a linear scan which took 176.6 ms CPU time per 100NN query, 442775 distance computations and needs to read about 9 MB of data. The bulk-loaded geodetic R*-tree took on average just 0.52 ms CPU per query, 779.5 distance computations and read 24 kB of data.

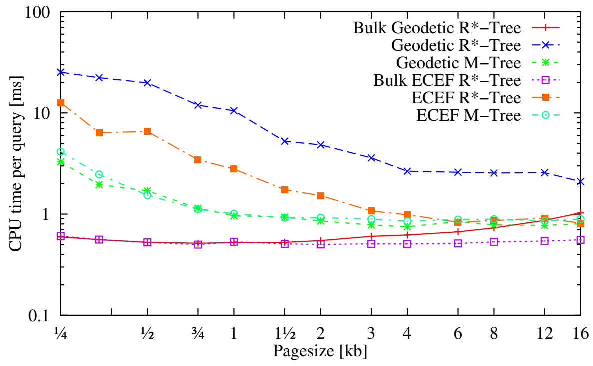
8.3.4.3 Accuracy and Efficiency

The UK traffic accidents and the Safecast data sets are good examples for regionally constrained data sets that can reasonably be handled with an appropriate local projection: for the traffic accidents we can use for example the UK Ordnance Survey 1936 (OSGB36) datum, which is a transversal Mercator projection that is expected to have low error in the UK. For the Safecast data set, UTM Zone 54S covers this part of Japan well. The projection library PROJ.4[8] we used for transforming the data refused to project coordinates outside of their design range (i.e. would not project the traffic accidents data to UTM 54S or SafeCast data to OSGB36). For DBpedia, neither could be used. We compare the nearest neighbors obtained by a linear scan over the data using geodetic distance to the nearest neighbors found using different settings: Euclidean distance in (non-Cartesian) latitude and longitude coordinates, but also after the transformation to the local coordinate system. Furthermore, we also used our index structures with geodetic distance. For a sample of 100000 objects, we computed the 100 nearest neighbors each and

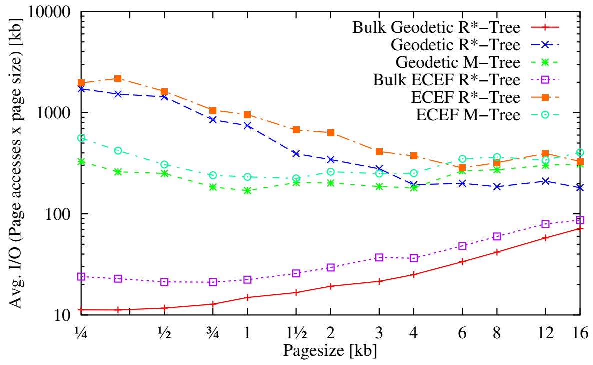
[8]Available at `http://trac.osgeo.org/proj/`



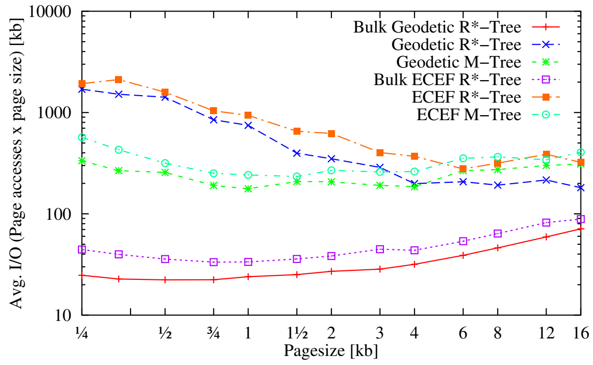
(a) In-memory runtime for 100NN queries



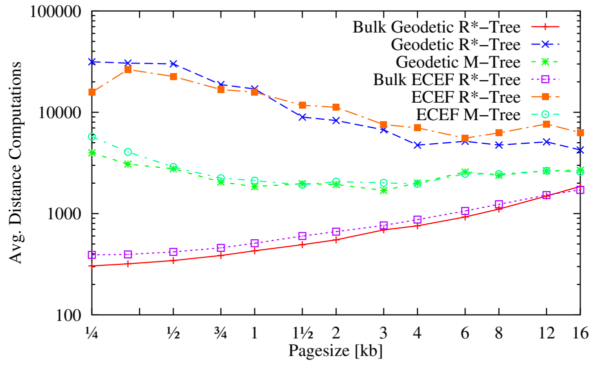
(b) In-memory runtime for range queries



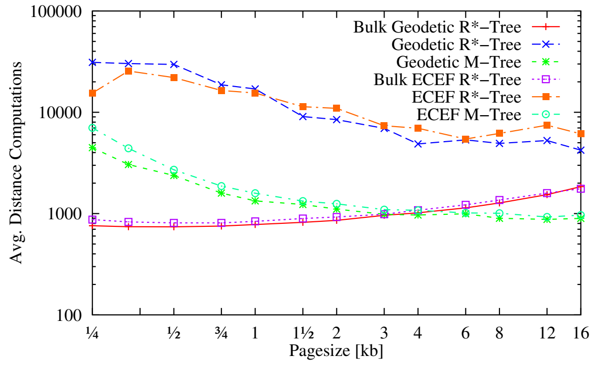
(c) I/O cost for 100NN queries



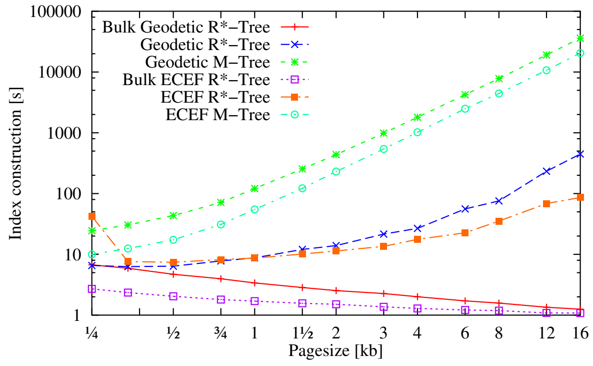
(d) I/O cost for range queries



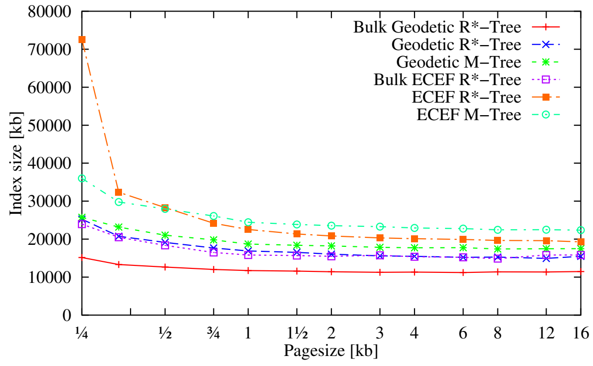
(e) Distance computations for 100NN



(f) Distance computations for range



(g) Construction time of index



(h) Size of index

Figure 8.14: Results for 100NN and range queries on DBpedia data set.

compute precision with respect to the ground truth from a linear scan. Table 8.2 gives the average CPU time per query (not taking I/O time into account). Unsurprisingly, Euclidean distance-based approaches were fastest, and index-based approaches were substantially faster than using a linear scan. On the linear scan, the spheroid model using the WGS84 spheroid and Vincenty formula was substantially slower – around 5 times – than the haversine formula using a spherical model. This is of course due to the use of trigonometric functions. When using an index, this difference is much smaller, since the point to rectangle minimum distance can always be computed using the simpler spherical model. Furthermore, the index-based queries scale much better with the data set size. Note that the 3D ECEF trees are faster than the R-tree built on the raw coordinates (λ, ϕ). However, the resulting tree will need more I/O, because the 3D vectors needs more storage. For these indexes, the index is actually queried using Euclidean distance; then the final result is again refined. Furthermore, while the 2D tree can be queried with any of the given distance functions, the 3D projection is specific to one particular earth model.

Table 8.3, Table 8.4, and Table 8.5 compare the results of the different approaches on the three data sets used. The given values are the average agreement of the 100NN results returned by the indexes. The WGS84 model is the most accurate earth model used in this experiment. The results of the cosine, haversine and spherical Vincenty formulas – which all use the same spherical earth model – differ only marginally due to numerical differences in particular at colocated and antipodal points. Clearly the worst quality is obtained by using the naïve Euclidean approach on the coordinates. On the world-scale DBpedia data set, it has an average agreement of just 90% to all other methods, on Great Britain only about 87% and on SafeCast around 96%. Since Great Britain is further north than Japan, the larger distortion is to be expected. The transformation to a local datum – OSGB36 for Great Britain and UTM Zone 54S for Japan – work reasonably well. In particular OSGB36 for Great Britain gives next to identical results to the spheroid earth model. For the world wide DBpedia data set, however, no such projection exists.

Furthermore, it can be seen that the two proposed R-tree indexes work as desired: they offer next to perfect agreement (sometimes slight numerical differences may arise, in particular on SafeCast with lots of duplicate coordinates) with the result obtained by a linear scan.

Table 8.2: CPU time of different approaches to 100NN search per query.

Formula	Index	DBpedia Places	Traffic Accidents	SafeCast
Data set size		475001	1209933	6303979
Bulk Load	λ, ϕ R-Tree	4 s	14 s	168 s
Bulk Load	ECEF 3D R-Tree	3 s	18 s	1330 s
Haversine	linear scan	165 s	376 s	1768 s
Sph. Vincenty	linear scan	231 s	516 s	2126 s
Cosine	linear scan	323 s	1159 s	5356 s
WGS84 Vincenty	linear scan	910 s	1738 s	8433 s
Euclidean	λ, ϕ R-Tree	78 ms	80 ms	114 ms
Haversine	λ, ϕ R-Tree	177 ms	174 ms	208 ms
Sph. Vincenty	λ, ϕ R-Tree	198 ms	198 ms	235 ms
Cosine	λ, ϕ R-Tree	328 ms	321 ms	383 ms
WGS84 Vincenty	λ, ϕ R-Tree	459 ms	412 ms	479 ms
Haversine	ECEF 3D R-Tree	128 ms	116 ms	163 ms
WGS84 Vincenty	ECEF 3D R-Tree	263 ms	248 ms	289 ms
Euclidean	UTM54S R-Tree	n/a	n/a	90 ms
Euclidean	OSGB36 R-Tree	n/a	76 ms	n/a

8.3.5 Conclusions

We introduced two approaches for indexing geographic data with support for geodetic distance queries (both nearest neighbor and radius queries). One approach is based on the well-known ECEF transformation of the data set into a 3-dimensional coordinate system, exploiting that the Euclidean distance is a monotone lower bound of the geodetic distance there. The other approach is even more elegant: it uses an R*-tree built on the raw geographic data in longitude, latitude coordinates that is also useful for example for window queries. However, the use of trigonometric formulas instead of the much simpler Euclidean distance results in higher CPU query cost. At the same time, the smaller index reduces the I/O cost due to the lower dimensionality.

Both approaches perform excellent, scaling logarithmic with data set size as opposed to a linear scan, yieding a more than 10000 fold speedup on the largest data set evaluated. Yet, validation showed that the results agree with the linear scan except for slight differences that can be attributed to numerical imprecision of floating point arithmetic (and clearly within the accuracy limits our earth models can offer). The ECEF-based index is computationally simpler, but requires more I/O and a dedicated index, whereas the computationally more intensive direct indexing approach allows the dual-use of the index structure for other queries (in particular, for map window queries) and can be applied directly to existing R-trees.

The missing piece of this puzzle to using 2D R*-trees for geodetic distance was the accurate lower-bound distance for point-to-rectangle distance computations introduced in this section.

It was not obvious that the existing R*-trees could be as easily reused without further modifications even without taking the spheroid nature of the earth into account at index construction time. This does, however, not rule out that a modified tree may sometimes perform better. For example when indexing data for the United States, a few objects in Alaska will be beyond the $180°$ boundary. Instead of inserting these objects in the far East – where they may end up sharing data pages with objects from Maine on the east coast to ensure minimum page fill – it may for example be beneficial to insert them in the far West. Similarly, there might exist an improved split strategy that produces a more efficient page structure. For bulk loading, it may be desirable to put extra emphasis on the longitude. But the benefits of these modifications may be highly application dependent and not be of general use, whereas the introduced distance computation lays the foundation for querying the resulting indexes, no matter how they were constructed.

Table 8.3: Precision of different approaches to 100NN search on DBpedia data set.

		WGS84	Havers.	Cosine	SphVin.	Euclid.	WGS84	Havers.	Cosine	SphVin.	WGS84	Havers.
		Linear scan				λ, ϕ R-Tree					ECEF 3D R-Tree	
WGS84 Vin.	linear scan	-	0.998713	0.998715	0.998712	0.903459	1.000000	0.998713	0.998715	0.998712	0.999983	0.998697
Havers.		0.998713	-	1.000000	0.999999	0.902482	0.998716	1.000000	1.000000	0.999999	0.998701	0.999982
Cosine		0.998715	1.000000	-	0.999974	0.902473	0.998693	0.999975	1.000000	0.999974	0.998684	0.999961
Sph. Vin.		0.998712	0.999999	0.999974	-	0.902482	0.998717	1.000000	1.000000	1.000000	0.998702	0.999983
Euclid.	λ, ϕ R-Tree	0.903459	0.902482	0.902473	0.902482	-	0.903416	0.902436	0.902449	0.902435	0.903411	0.902428
WGS84 Vin.		1.000000	0.998716	0.998693	0.998717	0.903416	-	0.998713	0.998715	0.998712	0.999983	0.998697
Havers.		0.998713	1.000000	0.999975	1.000000	0.902436	0.998713	-	1.000000	0.999999	0.998701	0.999982
Cosine		0.998715	1.000000	1.000000	1.000000	0.902449	0.998715	1.000000	-	0.999974	0.998684	0.999961
Sph. Vin.		0.998712	0.999999	0.999974	1.000000	0.902435	0.998712	0.999999	0.999974	-	0.998702	0.999983
WGS84 Vin.	ECEF	0.999983	0.998701	0.998684	0.998702	0.903411	0.999983	0.998701	0.998684	0.998702	-	0.998703
Havers.		0.998697	0.999982	0.999961	0.999983	0.902428	0.998697	0.999982	0.999961	0.999983	0.998703	-

Table 8.4: Precision of different approaches to 100NN search on Traffic data set

		WGS84	Havers.	Cosine	SphVin.	Euclid.	Euclid.	WGS84	Havers.	Cosine	SphVin.	WGS84	Havers.
		Linear scan				OSGB36	λ, ϕ R-Tree					ECEF 3D R-Tree	
WGS84 Vin.	linear scan	-	0.999094	0.999095	0.999094	0.999995	0.868654	1.000000	0.999094	0.999095	0.999094	1.000000	0.999094
Havers.		0.999094	-	0.999999	1.000000	0.999084	0.867957	0.999086	1.000000	0.999999	1.000000	0.999086	1.000000
Cosine		0.999095	0.999999	-	0.999997	0.999083	0.867956	0.999085	0.999997	1.000000	0.999997	0.999085	0.999997
Sph. Vin.		0.999094	1.000000	0.999997	-	0.999084	0.867957	0.999086	1.000000	0.999999	1.000000	0.999086	1.000000
Euclid. OSGB36		0.999995	0.999084	0.999083	0.999084	-	0.868634	0.999996	0.999093	0.999094	0.999093	0.999996	0.999093
Euclid.	λ, ϕ R-Tree	0.868654	0.867957	0.867956	0.867957	0.868634	-	0.868635	0.867945	0.867946	0.867945	0.868635	0.867945
WGS84 Vin.		1.000000	0.999086	0.999085	0.999086	0.999996	0.868635	-	0.999094	0.999095	0.999094	1.000000	0.999094
Havers.		0.999094	1.000000	0.999997	1.000000	0.999093	0.867945	0.999094	-	0.999999	1.000000	0.999086	1.000000
Cosine		0.999095	0.999999	1.000000	0.999999	0.999094	0.867946	0.999095	0.999999	-	0.999997	0.999085	0.999997
Sph. Vin.		0.999094	1.000000	0.999997	1.000000	0.999093	0.867945	0.999094	1.000000	0.999997	-	0.999086	1.000000
WGS84 Vin.	ECEF	1.000000	0.999086	0.999085	0.999086	0.999996	0.868635	1.000000	0.999086	0.999085	0.999086	-	0.999094
Havers.		0.999094	1.000000	0.999997	1.000000	0.999093	0.867945	0.999094	1.000000	0.999997	1.000000	0.999094	-

Table 8.5: Precision of different approaches to 100NN search on SafeCast data set

		WGS84	Havers.	Cosine	SphVin.	Euclid.	Euclid.	WGS84	Havers.	Cosine	SphVin.	WGS84	Havers.
		Linear scan				UTM54S	λ, ϕ R-Tree					ECEF 3D R-Tree	
WGS84 Vin.	linear scan	-	0.998056	0.997838	0.997932	0.989779	0.959528	0.999967	0.998056	0.997103	0.997920	0.999579	0.997767
Havers.		0.998056	-	0.999499	0.999853	0.988476	0.957975	0.998076	1.000000	0.998736	0.999842	0.997805	0.999538
Cosine		0.997838	0.999499	-	0.996392	0.987060	0.956424	0.994857	0.996418	0.998949	0.996393	0.994635	0.996133
Sph. Vin.		0.997932	0.999853	0.996392	-	0.988549	0.957975	0.998081	0.999982	0.998836	0.999989	0.997899	0.999630
Euclid. UTM54S		0.989779	0.988476	0.987060	0.988549	-	0.955715	0.989686	0.988358	0.989274	0.988295	0.989541	0.988242
Euclid.	λ, ϕ R-Tree	0.959528	0.957975	0.956424	0.957975	0.955715	-	0.958524	0.956921	0.957437	0.956810	0.958338	0.956672
WGS84 Vin.		0.999967	0.998076	0.994857	0.998081	0.989686	0.958524	-	0.998037	0.997087	0.997913	0.999566	0.997751
Havers.		0.998056	1.000000	0.996418	0.999982	0.988358	0.956921	0.998037	-	0.998736	0.999842	0.997805	0.999538
Cosine		0.997103	0.998736	0.998949	0.998836	0.989274	0.957437	0.997087	0.998736	-	0.996641	0.994892	0.996401
Sph. Vin.		0.997920	0.999842	0.996393	0.999989	0.988295	0.956810	0.997913	0.999842	0.996641	-	0.997890	0.999628
WGS84 Vin.	ECEF	0.999579	0.997805	0.994635	0.997899	0.989541	0.958338	0.999566	0.997805	0.994892	0.997890	-	0.997928
Havers.		0.997767	0.999538	0.996133	0.999630	0.988242	0.956672	0.997751	0.999538	0.996401	0.999628	0.997928	-

9 Customization Case Studies

In this chapter we want to study some of the new applications enabled by the generalization and abstraction of outlier detection performed in Chapter 6. In particular, we want to study the ability to detect non-standard types of outliers by customizing the general method accordingly to suite the particular needs of these challenges.

> “Far better an approximate answer to the *right* question, which is often vague, than an *exact* answer to the wrong question, which can always be made precise.
> — John W. Tukey [Tuk62]”

While a lot of effort was put into mining the top-n outliers with respect to different algorithms such as kNN-Outlier and LOF, few scientists questioned whether the method is generally applicable for real problems. In Chapter 6 we abstracted from the exact methods to their general scheme, in the following we will go the opposite direction: we start with the generalized scheme, and adopt it to a particular problem at hand: we try to ask the *right* question. We will accept some imprecision in the computation, if we then get faster results and the error remains on the level that we have to expect from the data anyway.

> “As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality. — Albert Einstein [Ein23]”

Contents of this chapter have since been published in:

> E. Schubert, A. Zimek, and H.-P. Kriegel. “Generalized Outlier Detection with Flexible Kernel Density Estimates”. In: *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*. 2014

9.1 Case Study: Kernel Density Estimation Outlier Detection (KDEOS)

In Section 5.1.3 we discussed the close relationship of local outlier detection to kernel density estimation; but how none of the discussed method uses a proper kernel function. Our proposed method does not divert substantially from the existing methods – we also perform density estimation, then compare the densities within local neighborhoods. As such, it is also an instantiation of the general pattern of *local outlier detection* [SZK12]. However, we propose to use classic kernel density estimation directly instead of experimenting with non-standard kernels without giving a good reason for this.

As a side effect, it solves some of the numerical issues in the existing methods: for example the LOF density estimation becomes undefined in the case of more than k duplicate points, as all distances for these points become 0, yielding $\mathrm{lrd} = 1/0$. The original LOF publication included a workaround for this, by using the k-distinct nearest neighbor, i.e. the k nearest neighbor that is actually different.

9.1.1 Density Estimation Step

The kernel function to use with our method is best to be considered an input parameter to the algorithm. We suggest to use either the radially symmetric Gaussian or Epanechnikov kernels of bandwidth h and dimensionality d (visualized in Section 2.3):

$$K_{\text{gauss},h}(u) := \frac{1}{(2\pi)^{d/2}h^d} e^{-\frac{1}{2}\frac{u^2}{h^2}}, \tag{9.1}$$

$$K_{\text{epanechnikov},h}(u) := \frac{3}{4h^d}\left(1 - \frac{u^2}{h^2}\right). \tag{9.2}$$

The radially symmetric versions have the benefit that we only have one bandwidth to estimate, instead of having to estimate full bandwidth matrices for each object, which continues to be a difficult problem [SS05]. The *balloon estimator* and *sample smoothing estimator* [TS92] are:

$$\mathrm{KDE}_{\text{balloon},h}(o) := \frac{1}{n}\sum_p K_{h(o)}\left(o - p\right), \tag{9.3}$$

$$\mathrm{KDE}_{\text{sample},h}(o) := \frac{1}{n}\sum_p K_{h(p)}\left(o - p\right). \tag{9.4}$$

The difference between these two is whether the bandwidth estimate depends on the evaluation point o or on the individual data samples p. In order to estimate the local kernel bandwidth $h(o)$ respectively $h(p)$, a classic approach is to use the nearest-neighbor distances [LQ65], i.e. $h(o) = k\text{-dist}(o)$. The similarity of LOF to KDE has been discussed in Section 5.1.3. For our method, we will use the balloon estimators, because research in kernel density estimation shows both theoretical and experimental benefits in multivariate KDE [TS92]. Nevertheless, there are concerns about the bias of this method on the long tail [SS05]. Sheather and Jones [SJ91] discuss a data-driven method to estimate kernel bandwidths known as the “plug-in bandwidth estimator”. For robustness, we use $h(o) = \min\{\mathrm{mean}_{p\in k\mathrm{NN}} d(p, o), \varepsilon\}$ to avoid division by 0 and to be more resistant to outliers in the $k\mathrm{NN}$. But in general, any advanced kernel density estimation method can be used for this step.

Essentially, for density estimation, we recommend to stick to the established and proven methods of density estimation, but additionally to consider runtime. For example, if you have database indexes available to accelerate range queries or k nearest neighbor queries, an approximate density estimation exploiting these indexes is desirable. If the kernel function $K(o-p)$ bears next to no weight beyond the k-nearest neighbor, we do not need to use these for density estimation. We can also drop constant scaling factors, which then yields:

$$n \cdot \mathrm{KDE}_{k\mathrm{NN}}(o) := \sum_{p \in k\mathrm{NN}(o)} K_{h(o)}\left(o-p\right). \tag{9.5}$$

Since the parameter k can be hard to choose, we propose to extend the method to cover a range of $k = k_{\min} \dots k_{\max}$ to produce a series of density estimates (one for each k). This approach is similar to LOCI [Pap+03], yet it is computationally more efficient and elegant, as the values of k are well-defined steps, while the LOCI model needs to test arbitrary ε-radii. By this extension, the method becomes an ensemble method [Agg12], typically yielding more stable and reliable results. In the broader sense, this can be seen as an ensemble method as discussed in Chapter 7, but we do not evaluate or optimize the individual ensemble members.

9.1.2 Density Comparison Step

The density comparison function used in LOF and its variants can be written as:

$$\mathrm{LOF}(o) := \operatorname*{mean}_{p \in k\mathrm{NN}(o)} \frac{\mathrm{lrd}(p)}{\mathrm{lrd}(o)} \equiv \frac{\mathrm{mean}_{p \in k\mathrm{NN}(o)} \mathrm{lrd}(p)}{\mathrm{lrd}(o)}.$$

For an object that has an average density, this factor will be close to 1, while for objects with neighborhoods of much higher density than that of the object, this value will be larger. However, there is little control over how large the values become, or when a value is to be considred significant.

For our approach, we use a slightly different comparison method. We assume that not only the local densities vary, but also the variability itself is sensitive to locality. Therefore, to standardize the deviation from normal density, we apply the well-known z-score transformation: Let μ_X be the mean of the set X and σ_X the standard deviation. The z-score of x then is $z(x,X) := (x - \mu_X)/\sigma_X$ (if $\sigma_X = 0$, then use $z(x,X) := 0$). Alternatively, one could use more robust statistics such as the median absolute deviation from median (MAD) [Ham74]. However, for small sample sizes, the mean often works better than the median. Only for large values, the median and MAD become more robust to outliers. Intuitively, a z-score of $+3$ indicates a deviation of three standard deviations. When using multiple values of k, we use the average z-score:

$$s(o) := \operatorname*{mean}_{k_{\min} \dots k_{\max}} z\left(\mathrm{KDE}(o), \{\mathrm{KDE}(p)\}_{p \in k\mathrm{NN}(o)}\right) \tag{9.6}$$

9.1.3 Score Normalization Step

For normalization, we use the cdf based score normalization discussed in Section 5.3.2.6.

Algorithm 6: Basic KDE Outlier Algorithm

```
$s :=$ array for output scores
$S :=$ two dimensional array, $o \times k_{\max}$
// Perform kernel density estimation (KDE):
foreach $o$ in DB do
    $N_{\max} =$ compute $k_{\max}$-nearest neighbors of $o$
    foreach $k$ in $k_{min} \ldots k_{max}$ do
        $h =$ compute kernel bandwidth from $N_{\max}[1;k]$
        foreach $n$ in $N_{max}[1;k]$ do
            $u =$ distance $(o, n)$
            $S[o][k] = S[o][k] + K(u, h)$
        end
    end
end
// Compare densities:
foreach $o$ in DB do
    $N_{\max} =$ compute/get $k_{\max}$-nearest neighbors of $o$
    foreach $k$ in $k_{min} \ldots k_{max}$ do
        $\mu :=$ mean of $S[N_{\max}][k]$
        $\sigma :=$ standard deviation of $S[N_{\max}][k]$
        $s[o] = s[o] + (\mu - S[o][k])/\sigma$
    end
    $s[o] = s[o]/(k_{\max} - k_{\min} + 1)$
end
// Return normalized scores:
return normalizeScores(s)
```

9.1.4 Algorithm and Complexity

Algorithm 6 gives the basic computation of the KDEOS scores for a range of $k = k_{\min} \ldots k_{\max}$. But note that we advocate to adapt this code to the particular problem at hand by integrating domain knowledge and specific requirements (we will demonstrate this in the experimental section). The overall code complexity is not high, if the evaluation framework can efficiently provide the neighbor sets. Also notice that the majority of this code is what is called "embarrassingly parallel": if the neighbor sets are precomputed, the main loops can be executed by many mappers in parallel. The aggregation of the $S[o]$ and $s[o]$ values then is the canonical reduce step. The intermediate data produced is of size $\mathcal{O}(nk^2)$, thus only linear in the input data size. Therefore, except for the neighborhood computation step, this algorithm is easy to implement in a distributed computation framework such as MapReduce.

The complexity of this method is comparable to LOF and many other outlier detection algorithms. In practice, the runtime is dominated by the cost of computing the kNN, which without index support requires $\mathcal{O}(n^2)$ distance computations. Index structures such as the R*-tree [Bec+90] for nearest neighbor search can reduce this runtime to $\mathcal{O}(n \log n)$.

Excluding the cost of computing the kNN, the main analysis loop then requires $\mathcal{O}(n \cdot k \cdot \Delta k)$

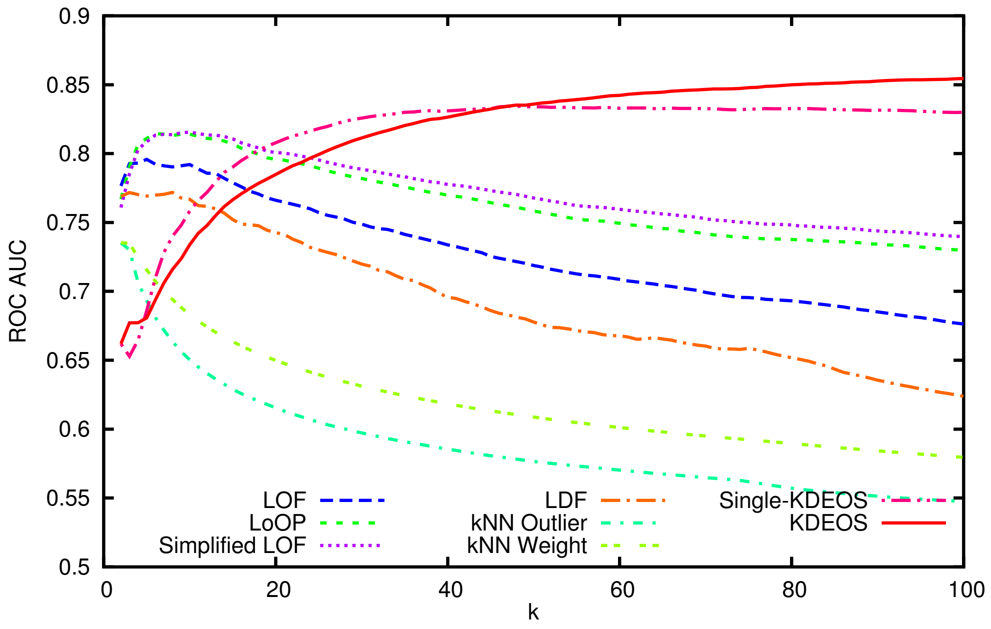


Figure 9.1: Performance over different values of k.

Table 9.1: Best performance of different algorithms.

Method	LOF	LoOP	S-LOF	LDF	kNN	kNN-W	Single-KDEOS	KDEOS
Best AUC	.795	.815	.816	.772	.735	.735	.834	**.855**
Best k	5	6	10	8	2	2	$k_{\text{min}} = k_{\text{max}} = 48$	$k_{\text{max}} = 100$

operations ($\Delta k = k_{\text{max}} - k_{\text{min}} + 1$), usually with $k \ll n$. While there is obviously a computational overhead for the more complex kernel functions, it is small compared to the data management and distance computation costs.

9.1.5 Experimental Results

The Amsterdam Library of Object Images (ALOI) [GBS05] is a collection of 110250 images of 1000 small objects, i.e. about 110 images of each object, from different angles and with different light conditions. By downsampling some of these classes to become rare objects, we obtained a data set retaining 75000 images, 717 of which are rare objects (known outliers, up to 4 from the same class).For our experiments, we used the 27-dimensional RGB histogram representation. Overall, this data set and task is a classic setting for density-based outlier detection: the rare objects are expected to be in less dense areas than the members of the clustered images. This data set is non-trivial: besides the labeled rare objects there are also outliers within the other classes coming from rare light situations, and on the other hand, some objects are very similar, and as such some downsampled objects may indeed have another full class that looks alike.

Figure 9.1 visualizes the performance of various algorithms on this data set, using the well-known ROC AUC measure. As competitors, we use LOF [Bre+00], LoOP [Kri+09a], Simplified-LOF [SZK12], LDF [LLP07], kNN outlier [RRS00], and kNN weight [AP02] (implementations in ELKI [Ach+13]). For Single-KDEOS we set $k_{\min} = k_{\max}$, while for KDEOS we used $k_{\min} = 1$. We report the numbers for the Gaussian kernel, but results using Epanechnikov kernel were almost identical. The best results and the k for the best result are given in Table 9.1. kNN outlier works best with very small values of k. In ELKI $k = 2$ is the 1NN distance, as the query point is part of the database. For LOF and most of its variants, there is a sweet spot in the small ks. Interestingly, the Simplified-LOF variants work much better than LOF on this data set, probably due to the structure of micro-clusters in this particular data set. Our proposed method produces much more stable results, in particular when choosing a large enough range of ks. Besides offering the best performance of the evaluated algorithms, the parameter k is also much easier to choose – it just needs to be large enough for the kernel density estimation to yield meaningful results. This also is the main limitation: for low values of k, the density estimates are not yet meaningful, and KDEOS thus does not (yet) yield good results, whereas a method such as kNN outlier detection, using a very simple density estimate, can often yield satisfying results with the 1-nearest neighbor distance.

9.2 Customization Case Study: Road Accidents Blackspots

9.2.1 Experimental Setup

To demonstrate the flexibility of our approach, we use a fairly large data set, the road accidents data set from the UK government, spanning the years 2005 to 2011 containing data on 1.2 million road accidents in the UK.[1] To be able to scale to a data set of this size, we employ the indexing techniques discussed in Section 8.3. The results of traditional outlier detection methods such as LOF are of little interest to the user: these outliers consist mostly of accidents on low use side roads that only see a single accident every dozen years. Instead, areas of interest in this data set are regions with a high concentration of car accidents, and again within these hotspots – many crossroads and roundabouts will show up as hotspots mainly due to the high volume of traffic – we are interested in those that particularly stick out with respect to the usual hotspots in their larger neighborhood.

The generalized method discussed in this thesis can be easily adapted to the particular needs of this data set. First of all, we modify the density estimation to use a fixed size kernel instead of a dynamic bandwidth. Since cars and roads have the same size across the country, we do not need to adapt the kernel size to local data trends. Instead, we chose a radius of 50 meters along with the Epanechnikov kernel (which drops to 0 beyond the maximal bandwidth). As distance function we choose the great-circle distance and an adapted R*-tree index to accelerate search (using Section 8.3). In the comparison step, we use a much larger radius of 2 kilometers. Instead

[1]UK government, data publicly available on `http://data.gov.uk/`

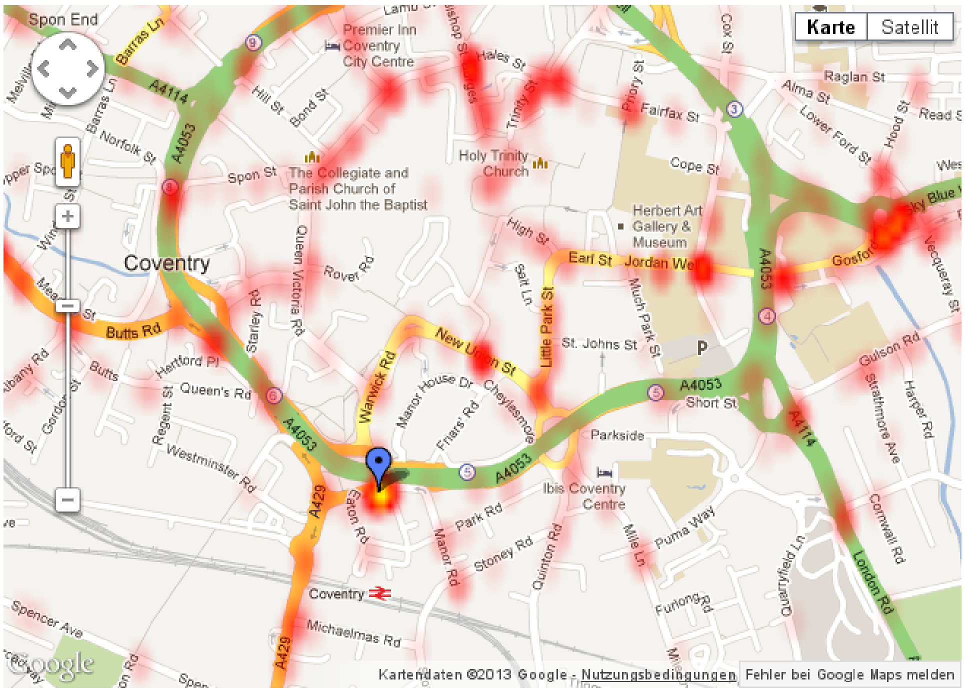
of looking for objects of unusually low density, we look for observations of an unusually high accident rate. Observations with a variance of 0 – which only happens in remote areas – are not reported as outliers. Last but not least, in order to find the top outliers, we do not need to apply normalization beyond the z-score, which is a user-friendly measure of magnitude by itself. Furthermore, in a post-processing step, we extract only the object of the largest density within a radius of 50 meters as representative of the hotspot. Obviously, other accidents at the same location will achieve nearly the same score, and reporting all of these to the user does not yield any benefit.

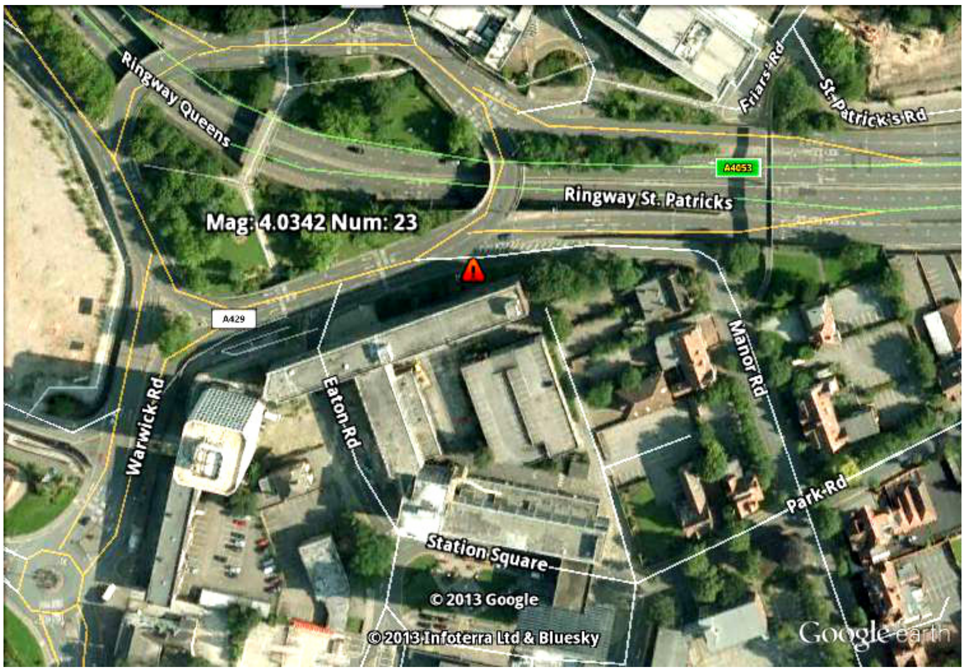
9.2.2 Results

Figure 9.2 shows an unusual hotspot in Coventry, UK visualized in Google Maps and Google Earth. In Figure 9.2a the traffic accident rate (i.e., our density estimation) is indicated using a heat map. We can see multiple other hotspots in the area, but the detected outlier clearly is an unusually high concentration (bright yellow indicates twice the concentration of bright red). When removing the overlay and zooming in further for Figure 9.2b, we can see what may be the cause for this particular hotspot: a three way merge with two lanes coming from the roundabout, two lanes coming from the ringway, and a fifth lane coming from the short term parking and car park at Coventry Station. This hotspot is found to have a 4 standard deviations higher accident rate than other accident sites.

A very different hotspot can be seen in Figure 9.3, near Sunderland in northern UK. A lookup on the internet confirmed Tunstall Hope Road to be a known "accident blackspot", "deathtrap", and "one of the most dangerous in Sunderland". Markings on the street now warn drivers to drive slowly. Figure 9.3a indicates that there is a particularly dangerous spot on this road, with 11 accidents at this particular corner. With the overlay removed in Figure 9.3b, the place seems to be very usual, but it is easy to imagine that this blind corner, combined with the lack of street lighting and probably slippery foliage can indeed be dangerous.

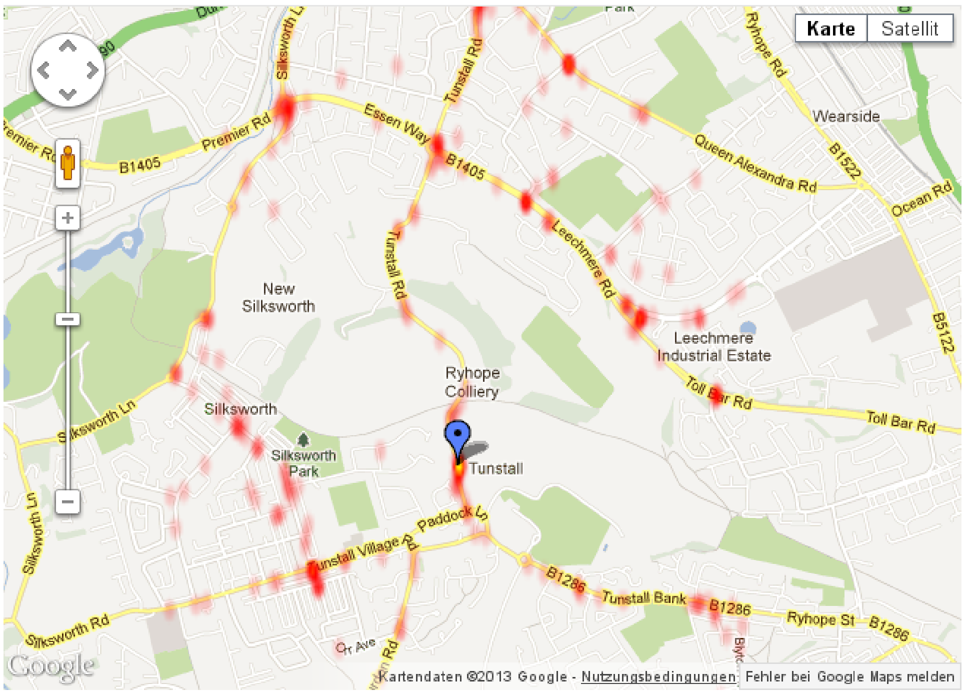
Yet, only two single spots of the top 50 outliers detected are in Greater London. One of them is seen in Figure 9.4, a huge roundabout with two multi-lane cut-throughs just north of the M4 motorway and Heathrow airport. This place was a top 10 serious accident site in Greater London, and has since been remodeled, removing one of the cut-throughs in an attempt to reduce car accidents. The reason why so few outliers were detected in London probably lies in the fact that in London there are so many high-accident junctions that none of them sticks out as substantially more serious than the others. Of course, none of the analyzed outliers is a new result. Traffic accident blackspots are usually well known to local police, and can be better analyzed. However, that this fairly general method can be easily adapted to this particular problem shows the flexibility of the approach. At the same time, it shows the following need: instead of looking for an off-the-shelf and parameterless algorithm, a good algorithm is modular and can this way easily be adapted to the domain knowledge for the desired use case.


(a) Accident density overlay in Google Maps

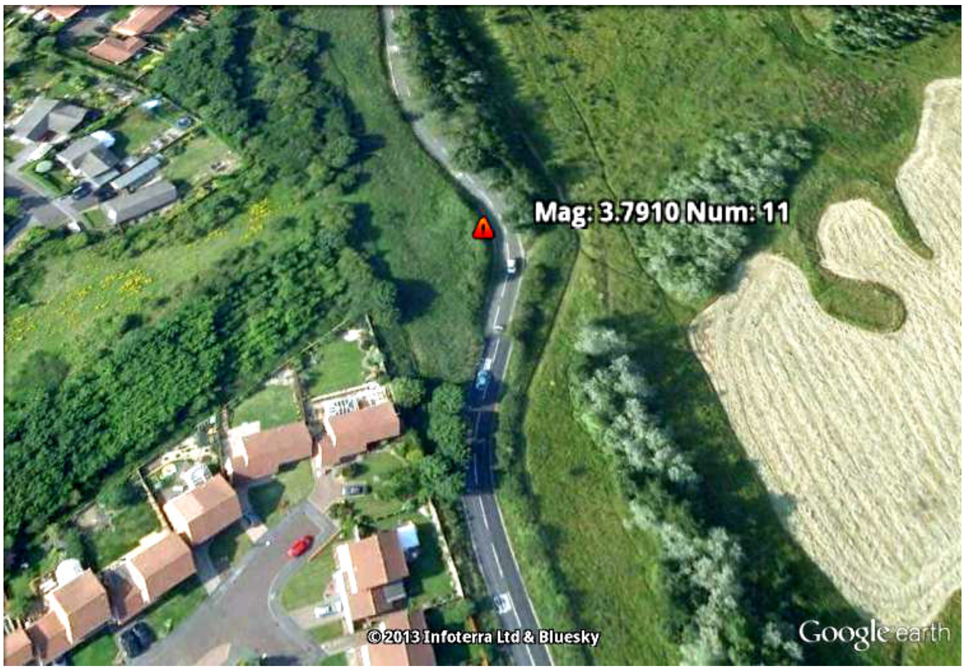


(b) 23 accidents at the three-way merge with the entrance to and exit from station square and the car parks there via Manor Road.

Figure 9.2: Traffic accident hotspot in Coventry, UK.
Background imagery © 2013 Google, Infoterra Ltd & Bluesky



(a) Accident density overlay in Google Maps



(b) Dangerous blind corner with 11 accidents. “Slow” markings were added to the street to warn drivers of the dangers ahead

Figure 9.3: Traffic accident hotspot in Sunderland, UK.
Background imagery © 2013 Google, Infoterra Ltd & Bluesky

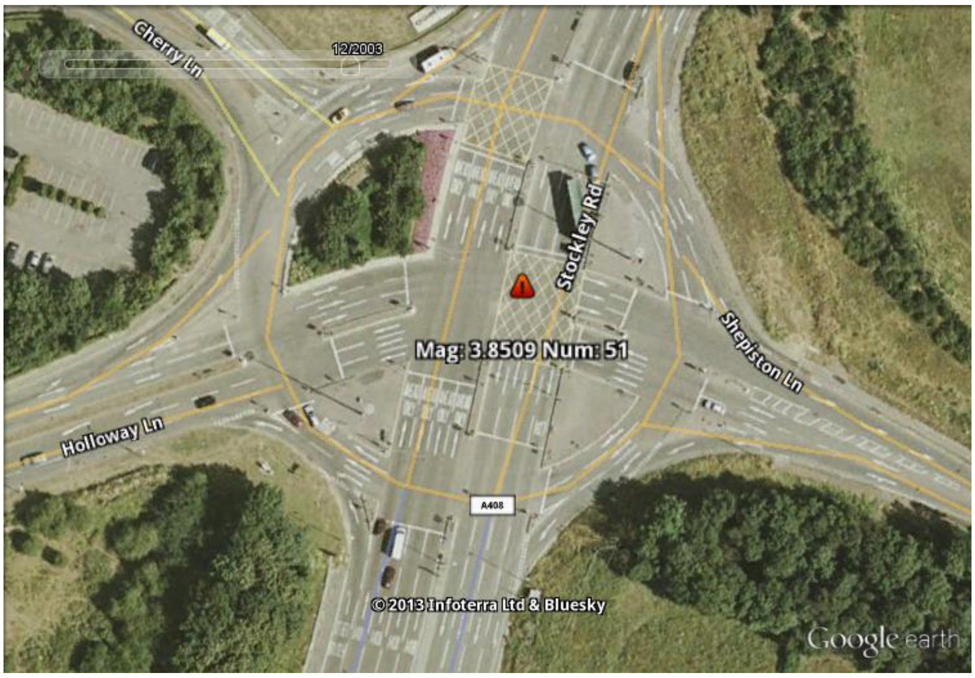


Figure 9.4: Traffic accident hotspot north of Heathrow airport.
The east-west cut-through has since been blocked.
Background imagery © 2013 Google, Infoterra Ltd & Bluesky

9.3 Customization Case Study: Radiation Measurements

9.3.1 Experimental Setup

For the second case study, we use a data set of 6.8 million radiation measurements taken (mostly) in Japan after the Fukushima nuclear disaster.[2] This is a spatio-temporal data set with very different data density due to different sampling rates, automated measurements and a high amount of noise due to different sensors, mobile sensors and low sensor quality. Again, classic outlier detection methods will be of little use, as they do not take spatial and temporal relationship into account. Outliers detected by running LOF on such a data set will not bear useful semantics, as LOF does not treat time, location, and radiation level attributes differently. Therefore, we again need to customize our method for this problem to get meaningful results.

The main customization point here is the definition of neighbors since here the temporal aspect is just as important as the spatial aspect. We define a specialized distance function as follows:

$$d(x \leftarrow y) := \begin{cases} \frac{\text{spatial}(x,y)}{100\text{ meter}} + \frac{\text{temporal}(x,y)}{1\text{ day}} & \text{if } x \text{ before } y \\ \infty & \text{otherwise} \end{cases}$$

[2]Publicly available for download on `http://safecast.org/`, average values can be explored on `http://map.safecast.org/`

This distance function combines both spatial and temporal differences, but it also excludes measurements that come later in time. The reason is that we only want to use a measurement x to estimate the value of a measurement y if it is in the future, in order to detect unexpected changes.

Using this combined (and asymmetric) distance function, we compute the $k\mathrm{NN}$ of each object. But instead of estimating the density of observations, we want to estimate the radiation level. For this we use the kernel density as *weight* for averaging the neighboring radiation levels. (An improved method could also take radioactive decay into account.) Mathematically, the method changes only marginally: each object now has a different weight instead of a unit value. This yields two different values for each point: an estimated (or "predicted") radiation level and an actual measurement. However, the raw difference of these two is not very useful yet: in areas where a lot of measurements were taken, we will be having much better predictions, and in areas with higher radiation levels, the natural differences will be much higher. So in order to be able to detect outliers in this data set, we need to put these measurements into a local context. For this we can again use the concepts of local outlier detection as introduced by LOF and discussed throughout this thesis. However, the SafeCast data are very noisy: there are measurement errors, varying general radiation levels, and an highly imbalanced number of measurements[3]. Therefore, we need to choose a rather large value of k as reference set, and instead of using mean and variance, we will this time use the median absolute deviation (MAD) between the predicted and the actual values to standardize the deviation. We can then again pinpoint the most "unusually extreme" deviations to the user.

9.3.2 Results

Many of the detected outliers are probably due to measurement errors and badly calibrated sensors. The data provided by SafeCast is in cpm, a unit that is considered to be highly sensor model dependent. Nevertheless, the inspected values all indicate that the outlier detection worked as desired: it was able to detect outliers of very different magnitudes in different areas of the map.

Some of the most extreme outliers were found along National Route 6, the closest highway to the Fukushima nuclear plants and closed since (in the bottom right of Figure 9.5). Here, readings of 8000 cpm have been reported in September 2011 and October 2011, while values of 3000 cpm (respectively 700) were predicted. On the online map by SafeCast, peak values of 10000 cpm and averages of 6000 are listed, but these averages do not take temporal aspects into account. On the other hand, there are also a number of outliers reported for Koriyama, consisting of measurements around 200, where 90 are considered normal. A large amount of outliers was detected in the area between Mount Hiyama and Mount Ryozen, an area that was in wind direction when radioactive material was released at Fukushima. Other outliers seem

[3]Some users contributed up to 50.000 automatic measurements mostly from their back yard, which are extremely redundant.

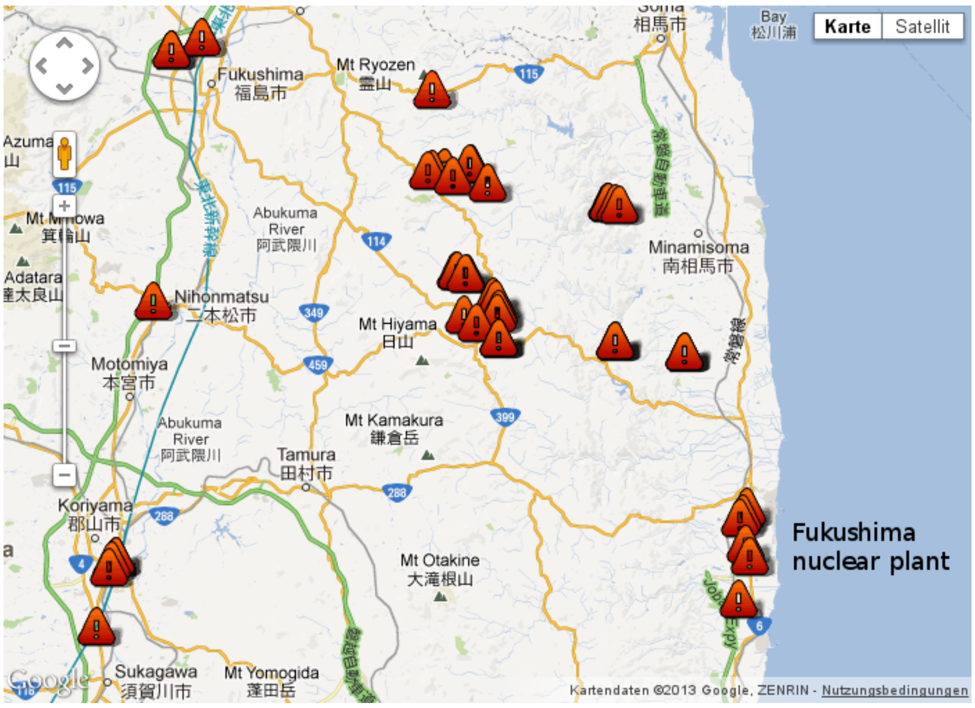


Figure 9.5: Outliers in radiation measurements, Fukushima prefecture, Japan, in Google Maps. Background imagery © 2013 Google, ZENRIN

to correlate with highway restaurants: It is plausible that a driver stopping there, by bringing in fresh air and dirt with his shoes into his car, can cause a measurable increase of the sensor readings.

A lesson learned from this analysis is that flexible index structures can help a lot with the computations. This is a fairly large data set. Yet, by using an R*-tree index with flexible distance function support (as we have not been using a standard Euclidean distance) for acceleration we were able to process it in reasonable time on a single host. Details on the indexing method used for acceleration can be found in Section 8.3.

10 Conclusions and Outlook

At this point, we have learned the basic principles of local outlier detection (Chapter 5) and the abstract structure of such algorithms (Chapter 6). Chapter 7 taught us how to combine the strengths of multiple algorithm runs, and Chapter 8 exemplified how to use approximations and indexing to scale them to larger data sets than before. Chapter 9 rounded up these results by carefully designing outlier detection algorithms for particular use cases.

While this thesis may first appear to be an exhaustive coverage of the issues of local outlier detection, it cannot be complete; any good research will lead to as many new questions as it answered. Other researchers have performed recent research in parallel, and I cannot include all of it. However, the thesis will hopefully be a good starting point for other researchers to continue this research. In the following, I will discuss some of the loose ends of the thesis, which are viable directions for future research.

Curse of Dimensionality: The performed study of the curse of dimensionality brought some new insights: there are different aspects to the curse, that may need different treatment. Grid based approaches do not scale well, however the combination of random projections with grids may actually be feasible (although better suited for cluster detection than for outlier detection). Instead of looking at the raw dimensionality, the intrinsic dimensionality and the signal-to-noise ratio may be more important; improved measures of intrinsic dimensionality may yield new way of detecting outliers in high-dimensional data. Shared nearest neighbors were shown to improve distance measurements despite the concentration effect of the primary distance, but are not a general cure for the problems of high-dimensional data. We may also be able to further improve with new distance measures specifically designed for high-dimensional data. While the approximate nearest neighbor search presented in Chapter 8 can handle medium dimensionality by using random projections, it will not scale to really high dimensionality either; nor does it support arbitrary distance measures.

High-dimensional Outlier Detection: The proposed methods SOD and COP are two early drafts of local outlier detection in high dimensional data; but in particular SOD was an early, simple approach. The ideas of COP could be fused into an improvement of SOD; and similarly, COP could benefit from the much more efficient dimension selection in SOD: COP does not scale too well to truly high-dimensional data. Last but not least, COP is designed with Euclidean neighborhoods for the correlation estimate, as other neighborhoods may be biased towards certain directions (intuitively, to find the optimal rotation, the neighborhood should be spherical!)

Density Estimation: We discussed the relationship of LOF to kernel density estimation, and in the case studies (Chapter 9) mostly used KDE based methods. However, in practical use, the classic LOF heuristic will often still work best. The data sets in the case studies were very beneficial to KDE, in particular the geographic data sets that have just 2-dimensional data, and the kernel bandwidth actually related to a physical dimension. In abstract data spaces, KDE itself is much more challenging; KDE is known to be itself prone to the curse of dimensionality. In particular, choosing the right bandwidth becomes next to impossible then. In such situations, the simple heuristic of LOF will work much better. Furthermore, kernel density estimation is also implicitly tied to Euclidean and Mahalanobis distance (it will also behave well with geodetic distances, obviously; but does KDE combine sensibly with e.g. Canberra distance?).

Choosing the Right Mean: LOF was shown to use the harmonic mean, and LoOP even the M^{-2} mean when interpreted as kernel density estimation. Given the general pattern, it would be an obvious choice to try the arithmetic M^1 and M^2 means instead. However, most likely, the results will not be fundamentally different (Chapter 9 actually used arithmetic means), as both means try to produce the most typical output. It may well be possible to plug-in very different statistics instead: the MAD, L-Moments, rank-order statistics, $\ldots$ – but instead of trying out all these one-by-one, we should step back, and figure out a proper theory to prefer one over the other. In this thesis, we assumed that the arithmetic mean, as used in kernel density estimation, would be the proper choice. But on the other hand, other means may be more robust.

Rescaling of Outlier Scores: The proposed rescaling of outlier scores connect the scores back to the statistical roots of probability distributions through cdf and pdf functions. The ensemble use case showed how to use the scores for Bayesian reasoning. Yet, it remains unclear if all of this lives up to the goal of giving the user a "user-friendly" outlier probability. No usability studies have been performed (which, in an expert domain such as data mining, probably is a lot harder than e.g. studying the usability of a mass-market physical device). Also the goal of "calibration" is not entirely achieved. The process of fitting a distribution to the observed outlier scores is an heuristic; in a perfect world we would know the true distribution of the outlier scores and not need to fit the distribution (and even choose the right distribution to fit). Replacing e.g. the quotient used in LOF for comparing models with a different combination function may yield a much more tractable score distribution. Maybe we can find an approach that even yields a provable distribution to use for modeling the score distribution.

Abstaining Methods: While the option of abstaining was discussed in the section on score normalization – detectors should be allowed to not find any outlier in the data set – this is not entirely achieved by the score normalization. By fitting a distribution to the scores, this approach to some extent enforces that the detectors also use the whole score range. As seen in Figure 5.15b, even on the overly uniform Halton sequence, the methods will detect some variation (although not with an overly high score), and cannot "opt out" from finding outliers.

Evaluation Measures: The newly proposed evaluation methods have the benefit of being able to measure also the similarity (and thus, diversity) of two results, not only compare the output to a binary ground truth. The measures were motivated from a cost model, so there exists some theory this is built upon. For supervised evaluation, they were shown to largely agree with the results of ROC AUC evaluation, but not yet on a formal level; and neither of the three weighted measures (Euclidean, Manhattan and Pearson based) has a clear advantage over the others in the experiments. Manhattan corresponds to linear errors, Euclidean to squared errors, so both are equally sound in their theory.

Ensembles: The greedy ensemble approach is a very basic heuristic that happens to work well in our experiments; but it mostly serves as a proof-of-concept for score rescaling and for measuring diversity. In particular, the weight and target vectors are crude heuristics, and could be refined.

Scalability: The need for scalability is becoming more and more important. While Chapter 8 suggested two approaches to improve performance, these could be enhanced by an efficient candidate generation process; so that only likely outliers have to be analyzed in detail. The approaches in Section 8.2 and Section 9.1 also show scalability to cluster computing, but the viability has not been evaluated in practice. Scaling up these approaches to truly "big data" will likely yield new challenges, such as parameterization problems of methods.

Other Data Types: Most of this thesis was built on data that can be represented as more-or-less dense vectors.[1] This is most evident in Chapter 4 and Chapter 8; and while many of the discussed outlier algorithms can work with any distance, the results may be much less convincing on very sparse data such as text in practice. Chapter 6 showed for example how to use the generalized method on graph data, but there remains a lot more work to be done.

Experiments and Data Sets: Most of the experiments reused the same data set over and over again. While it would be desirable to widen up the experiments to more data sets, it is surprisingly hard to find good data sets for evaluating anomaly detection; there is an ongoing effort to build a public repository of data for this. A good evaluation data set must have well-defined outliers, but these must actually be unusual; not just have a rare class label (a rare class with low variance is still a dense cluster). Downsampling classes works sometimes, but

[1]Technically, the color histograms are sparse vectors; but not comparable to e.g. sparse text vectors.

it neglects the fact that there could be in-class outliers, too. A good data set must also be non-trivial. Many data sets can be solved with an ROC AUC of 0.999 using the 1 nearest neighbor distance with Euclidean metric. Some seem to be intractable with some parameters (e.g. the Satimage-2 data set with $k = 20$), but with a large enough k they become too easy. How do you compare two methods that both score near-perfect on such data?

Flexible Codebase: While the currently available source-code in ELKI is quite flexible and covers a large number of methods, it does not reflect the level of abstraction presented in this thesis. Chapter 6 presents graphical representations of many algorithms, composed of primitive modules, and discusses the virtues of sharing, reusing and exchanging these modules to adapt an algorithm to a new problem. The ELKI source however reflects modularity at a more traditional Java level. While distance functions and indexes can easily be replaced, implementing a more complex variation of the existing algorithms cannot be done simply by reconnecting such modules or in a few lines of scripting code. And even then, the process cannot be trivially scaled up to a cluster computing framework.

Despite of these various open ends for future research, the thesis gives a broad overview of local outlier detection, and shares novel insights into how the methods work, their theoretical background, and how they can be extended and improved. The resulting scores are no longer "just a number", but are now rooted on statistical reasoning; not based on the original data, but on the distribution of scores.

Bibliography

[ABK98] M. Ankerst, S. Berchtold, and D. A. Keim. "Similarity Clustering of Dimensions for an Enhanced Visualization of Multidimensional Data". In: *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS), Phoenix, AZ.* 1998, pp. 52–60. DOI: 10.1109/INFVIS.1998.729559.

[Ach+06a] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. "Finding Hierarchies of Subspace Clusters". In: *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Berlin, Germany.* 2006, pp. 446–453. DOI: 10.1007/11871637_42.

[Ach+06b] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. "Deriving Quantitative Models for Correlation Clusters". In: *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA.* 2006, pp. 4–13. DOI: 10.1145/1150402.1150408.

[Ach+06c] E. Achtert, C. Böhm, P. Kröger, and A. Zimek. "Mining Hierarchies of Correlation Clusters". In: *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM), Vienna, Austria.* 2006, pp. 119–128. DOI: 10.1109/SSDBM.2006.35.

[Ach+07] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. "On Exploring Complex Relationships of Correlation Clusters". In: *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada.* 2007, pp. 7–16. DOI: 10.1109/SSDBM.2007.21.

[Ach+08] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek. "Global Correlation Clustering Based on the Hough Transform". In: *Statistical Analysis and Data Mining* 1.3 (2008), pp. 111–127. DOI: 10.1002/sam.10012.

[Ach+09] E. Achtert, T. Bernecker, H.-P. Kriegel, E. Schubert, and A. Zimek. "ELKI in Time: ELKI 0.2 for the Performance Evaluation of Distance Measures for Time Series". In: *Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD), Aalborg, Denmark.* 2009, pp. 436–440. DOI: 10.1007/978-3-642-02982-0_35.

[Ach+10] E. Achtert, H.-P. Kriegel, L. Reichert, E. Schubert, R. Wojdanowski, and A. Zimek. "Visual Evaluation of Outlier Detection Models". In: *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA), Tsukuba, Japan.* 2010, pp. 396–399. DOI: 10.1007/978-3-642-12098-5_34.

[Ach+11] E. Achtert, A. Hettab, H.-P. Kriegel, E. Schubert, and A. Zimek. “Spatial Outlier Detection: Data, Algorithms, Visualizations”. In: *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN.* 2011, pp. 512–516. DOI: 10.1007/978-3-642-22922-0_41.

[Ach+12] E. Achtert, S. Goldhofer, H.-P. Kriegel, E. Schubert, and A. Zimek. “Evaluation of Clusterings – Metrics and Visual Support”. In: *Proceedings of the 28th International Conference on Data Engineering (ICDE), Washington, DC.* 2012, pp. 1285–1288. DOI: 10.1109/ICDE.2012.128.

[Ach+13] E. Achtert, H.-P. Kriegel, E. Schubert, and A. Zimek. “Interactive Data Mining with 3D-Parallel-Coordinate-Trees”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), New York City, NY.* 2013, pp. 1009–1012. DOI: 10.1145/2463676.2463696.

[Ach01] D. Achlioptas. “Database-friendly Random Projections”. In: *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Santa Barbara, CA.* 2001.

[Ach03] D. Achlioptas. “Database-friendly Random Projections: Johnson-Lindenstrauss with binary coins”. In: *Journal of Computer and System Sciences* 66 (2003), pp. 671–687. DOI: 10.1016/S0022-0000(03)00025-4.

[AD52] T. W. Anderson and D. A. Darling. “Asymptotic theory of certain “goodness of fit” criteria based on stochastic processes”. In: *The Annals of Mathematical Statistics* 23.2 (1952), pp. 193–212.

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. “Efficient Similarity Search in Sequence Databases”. In: *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO), Chicago, IL.* 1993, pp. 69–84. DOI: 10.1007/3-540-57301-1_5.

[Agg+99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. “Fast Algorithms for Projected Clustering”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA.* 1999, pp. 61–72. DOI: 10.1145/304182.304188.

[Agg12] C. C. Aggarwal. “Outlier Ensembles”. In: *ACM SIGKDD Explorations* 14.2 (2012), pp. 49–58.

[Agg13] C. C. Aggarwal. *Outlier Analysis.* Springer, 2013. ISBN: 9781461463955.

[AHK01] C. C. Aggarwal, A. Hinneburg, and D. Keim. “On the Surprising Behavior of Distance Metrics in High Dimensional Space”. In: *Proceedings of the 8th International Conference on Database Theory (ICDT), London, UK.* 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27.

[AKZ08] E. Achtert, H.-P. Kriegel, and A. Zimek. “ELKI: A Software System for Evaluation of Subspace Clustering Algorithms”. In: *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China.* 2008, pp. 580–585. DOI: 10.1007/978-3-540-69497-7_41.

[AN07] A. Asuncion and D. J. Newman. *UCI Machine Learning Repository.* http://archive.ics.uci.edu/ml/. 2007.

[Ank+99] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. “OPTICS: Ordering Points To Identify the Clustering Structure”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA.* 1999, pp. 49–60. DOI: 10.1145/304182.304187.

[ANR74] N. Ahmed, T. Natarajan, and K. R. Rao. “Discrete Cosine Transform”. In: *IEEE Transactions on Computers* 23.1 (1974), pp. 90–93.

[Ans95] L. Anselin. “Local Indicators of Spatial Association–LISA”. In: *Geographical Analysis* 27.2 (1995), pp. 93–115.

[AP02] F. Angiulli and C. Pizzuti. “Fast Outlier Detection in High Dimensional Spaces”. In: *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discoverys (PKDD), Helsinki, Finland.* 2002, pp. 15–26. DOI: 10.1007/3-540-45681-3_2.

[AP05] F. Angiulli and C. Pizzuti. “Outlier mining in large high-dimensional data sets”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.2 (2005), pp. 203–215. DOI: 10.1109/TKDE.2005.31.

[Ass+07a] I. Assent, R. Krieger, E. Müller, and T. Seidl. “DUSC: Dimensionality Unbiased Subspace Clustering”. In: *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), Omaha, NE.* 2007, pp. 409–414. DOI: 10.1109/ICDM.2007.49.

[Ass+07b] I. Assent, R. Krieger, E. Müller, and T. Seidl. “Subspace Outlier Mining in Large Multimedia Databases”. In: *Parallel Universes and Local Patterns.* 2007.

[Ass+08] I. Assent, R. Krieger, E. Müller, and T. Seidl. “EDSC: efficient density-based subspace clustering”. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), Napa Valley, CA.* 2008, pp. 1093–1102. DOI: 10.1145/1458082.1458227.

[AY00] C. C. Aggarwal and P. S. Yu. “Finding Generalized Projected Clusters in High Dimensional Space”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX.* 2000, pp. 70–81. DOI: 10.1145/342009.335383.

[AY01] C. C. Aggarwal and P. S. Yu. “Outlier Detection for High Dimensional Data”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Santa Barbara, CA.* 2001, pp. 37–46. DOI: 10.1145/375663.375668.

[Bam75] Donald Bamber. “The area above the ordinal dominance graph and the area below the receiver operating characteristic graph”. In: *Journal of Mathematical Psychology* 12.4 (1975), pp. 387–415. DOI: 10.1016/0022-2496(75)90001-2.

[Bau+04] C. Baumgartner, K. Kailing, H.-P. Kriegel, P. Kröger, and C. Plant. “Subspace Selection for Clustering High-Dimensional Data”. In: *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), Brighton, UK.* 2004, pp. 11–18. DOI: 10.1109/ICDM.2004.10112.

[BC57] J. R. Bray and J. T. Curtis. “An ordination of the upland forest communities of southern Wisconsin”. In: *Ecological monographs* 27.4 (1957), pp. 325–349.

[BC94] D. Berndt and J. Clifford. “Using Dynamic Time Warping to Find Patterns in Time Series”. In: *KDD Workshop.* 1994, pp. 359–370.

[Bec+90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. “The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Atlantic City, NJ.* 1990, pp. 322–331. DOI: 10.1145/93597.98741.

[Bel61] R. Bellman. *Adaptive Control Processes. A Guided Tour.* Princeton: Princeton University Press, 1961.

[Ben75] J. L. Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517. DOI: 10.1145/361002.361007.

[Ber+10a] T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. “Subspace Similarity Search: Efficient k-NN Queries in Arbitrary Subspaces”. In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 555–564. DOI: 10.1007/978-3-642-13818-8_38.

[Ber+10b] T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. “Subspace Similarity Search Using the Ideas of Ranking and Top-k Retrieval”. In: *Proceedings of the 26th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank), Long Beach, CA.* 2010, pp. 4–9. DOI: 10.1109/ICDEW.2010.5452771.

[Ber+11] T. Bernecker, M. E. Houle, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. “Quality of Similarity Rankings in Time Series”. In: *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN.* 2011, pp. 422–440. DOI: 10.1007/978-3-642-22922-0_25.

[Bey+99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. “When Is “Nearest Neighbor” Meaningful?” In: *Proceedings of the 7th International Conference on Database Theory (ICDT), Jerusalem, Israel.* 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15.

[BFG99] K. P. Bennett, U. Fayyad, and D. Geiger. “Density-Based Indexing for Approximate Nearest-Neighbor Queries”. In: *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA.* 1999, pp. 233–243. DOI: 10.1145/312129.312236.

[BGV92] B.E. Boser, I. Guyon, and V. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT), New York, NY.* 1992, pp. 144–152. DOI: `10.1145/130385.130401`.

[Bil12] D. Bilková. “Lognormal distribution and using L-moment method for estimating its parameters”. In: *International Journal of Mathematical Models and Methods in Applied Sciences* 6.1 (2012), pp. 30–44.

[BK10] G. Brown and L. I. Kuncheva. ““Good” and “Bad” Diversity in Majority Vote Ensembles”. In: *9th International Workshop on Multiple Classifier Systems (MCS), Cairo, Egypt.* 2010, pp. 124–133. DOI: `10.1007/978-3-642-12127-2_13`.

[BL94] V. Barnett and T. Lewis. *Outliers in Statistical Data.* 3rd. John Wiley&Sons, 1994.

[Bla90] S. Blakeslee. *Lost on Earth: Wealth of Data Found in Space.* The New York Times. Mar. 1990. URL: `http://www.nytimes.com/1990/03/20/science/lost-on-earth-wealth-of-data-found-in-space.html` (visited on 10/30/2012).

[Böh+04a] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. “Density Connected Clustering with Local Subspace Preferences”. In: *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), Brighton, UK.* 2004, pp. 27–34. DOI: `10.1109/ICDM.2004.10087`.

[Böh+04b] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. “Computing Clusters of Correlation Connected Objects”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France.* 2004, pp. 455–466. DOI: `10.1145/1007568.1007620`.

[Bou+01] N. Boujemaa, J. Fauqueur, M. Ferecatu, F. Fleuret, V. Gouet, B. Le Saux, and H. Sahbi. “IKONA: Interactive Generic and Specific Image Retrieval”. In: *Proceedings of the International Workshop on MultiMedia Content-Based Indexing and Retrieval (MMCBIR 2001), Rocquencourt, France.* 2001, pp. 25–28.

[BPM04] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. “A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data”. In: *ACM SIGKDD Explorations* 6.1 (2004), pp. 20–29. DOI: `10.1145/1007730.1007735`.

[Bre+00] M. M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander. “LOF: Identifying Density-based Local Outliers”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX.* 2000, pp. 93–104. DOI: `10.1145/342009.335388`.

[Bre+01] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. “Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Santa Barbara, CA.* 2001, pp. 79–90. DOI: `10.1145/375663.375672`.

[Bre01] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: `10.1023/A:1010933404324`.

[Bre96] Leo Breiman. “Bagging predictors”. In: *Machine Learning* 24.2 (1996), pp. 123–140. DOI: 10.1007/BF00058655.

[Bri50] G. W. Brier. “Verification of Forecasts Expressed in terms of Probability”. In: *Monthly Weather Review* 78.1 (1950), pp. 1–3.

[Bro+05] G. Brown, J. Wyatt, R. Harris, and X. Yao. “Diversity creation methods: a survey and categorisation”. In: *Information Fusion* 6 (2005), pp. 5–20. DOI: 10.1016/j.inffus.2004.04.004.

[Bro04] J. Brockman. *A theory of roughness: A talk with Benoit Mandelbrot*. EDGE. Dec. 19, 2004. URL: http://edge.org/conversation/a-theory-of-roughness (visited on 05/29/2013).

[Bro97] A. Z. Broder. “On the resemblance and containment of documents”. In: *Compression and Complexity of Sequences 1997. Proceedings.* 1997, pp. 21–29.

[But71] A. R. Butz. “Alternative algorithm for Hilbert’s space-filling curve”. In: *IEEE Transactions on Computers* 100.4 (1971), pp. 424–426.

[CF03] A. L.-M. Chiu and A. W.-C. Fu. “Enhancements on Local Outlier Detection”. In: *Proceedings of the 7th International Database Engineering and Applications Symposium (IDEAS 2003), Hong Kong, China.* 2003, pp. 298–307. DOI: 10.1109/IDEAS.2003.1214939.

[CF99] K.-P. Chan and A. W.-C. Fu. “Efficient Time Series Matching by Wavelets”. In: *Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia.* 1999, pp. 126–133. DOI: 10.1109/ICDE.1999.754915.

[Cha+08] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi. “Automatically countering imbalance and its empirical relationship to cost”. In: *Data Mining and Knowledge Discovery* 17.2 (2008), pp. 225–252. DOI: 10.1007/s10618-008-0087-0.

[Cha08] W. Chauvenet. *A manual of spherical and practical astronomy: Embracing the general problems of spherical astronomy, the special applications to nautical astronomy, and the theory and use of fixed and portable astronomical instruments, with an appendix on the method of least squares.* Vol. 2. JB Lippincott, 1908.

[Cha98] T. M. Chan. “Approximate Nearest Neighbor Queries Revisited”. In: *Discrete & Computational Geometry* 20.3 (1998), pp. 359–373. DOI: 10.1007/PL00009390.

[CLB10] F. Chen, C.-T. Lu, and A. P. Boedihardjo. “GLS-SOD: A Generalized Local Statistical Approach for Spatial Outlier Detection”. In: *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC.* 2010, pp. 1069–1078. DOI: 10.1145/1835804.1835939.

[CN04a] Y. Cai and R. Ng. “Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France.* 2004, pp. 599–610. DOI: 10.1145/1007568.1007636.

[CN04b] L. Chen and R. Ng. “On the marriage of Lp-norms and edit distance”. In: *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada.* 2004, pp. 792–803.

[CÖO05] L. Chen, M. T. Özsu, and V. Oria. “Robust and fast similarity search for moving object trajectories”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Baltimore, ML.* 2005, pp. 491–502. DOI: `10.1145/1066157.1066213`.

[CP98] P. Ciaccia and M. Patella. “Bulk loading the M-tree”. In: *Proceedings of th 9th Australasian Database Conference (ADC), Perth, Australia.* 1998.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. “M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces”. In: *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB), Athens, Greece.* 1997, pp. 426–435.

[CR86] J. Chen and H. Rubin. “Bounds for the difference between median and mean of Gamma and Poisson distributions”. In: *Statistics & Probability Letters* 4.6 (1986), pp. 281–283.

[CR92] C. Croux and P. Rousseeuw. “Time-efficient algorithms for two highly robust estimators of scale”. In: *Proceedings of the 10th Symposium on Computational Statistics, COMPSTAT, Neuchâtel, Switzerland* (1992), pp. 411–428.

[Cra07] “Forum for: Use the Simples Model, But Not Too Simple”. In: *Communications of the ACM* 50.6 (June 2007). Ed. by Diane Crawford, pp. 7–9. ISSN: 0001-0782. DOI: `10.1145/1247001.1247014`. URL: `http://doi.acm.org/10.1145/1247001.1247014`.

[CS06] S. Chawla and P. Sun. “SLOM: A New Measure for Local Spatial Outliers”. In: *Knowledge and Information Systems (KAIS)* 9.4 (2006), pp. 412–429. DOI: `10.1007/s10115-005-0200-2`.

[CW69] S. C. Choi and R. Wette. “Maximum likelihood estimation of the parameters of the gamma distribution and their bias”. In: *Technometrics* (1969), pp. 683–690.

[Dan+14] X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert. “Discriminative Features for Identifying and Interpreting Outliers”. In: *Proceedings of the 30th International Conference on Data Engineering (ICDE), Chicago, IL.* 2014.

[Dat+04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. “Locality-sensitive hashing scheme based on p-stable distributions”. In: *Proceedings of the 20th ACM Symposium on Computational Geometry (ACM SoCG), Brooklyn, NY.* 2004, pp. 253–262.

[dCH10] T. de Vries, S. Chawla, and M. E. Houle. “Finding Local Anomalies in Very High Dimensional Space”. In: *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), Sydney, Australia.* 2010, pp. 128–137. DOI: `10.1109/ICDM.2010.151`.

[dCH12] T. de Vries, S. Chawla, and M. E. Houle. “Density-preserving projections for large-scale local anomaly detection”. In: *Knowledge and Information Systems (KAIS)* 32.1 (2012), pp. 25–52. DOI: `10.1007/s10115-011-0430-4`.

[DD06] M.-M. Deza and E. Deza. *Dictionary of distances.* Elsevier, 2006. ISBN: 0080465544.

[Dea+13] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. “Fast, Accurate Detection of 100,000 Object Classes on a Single Machine”. In: *Proceedings of the 2013 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2013), Portland, OR.* 2013, pp. 2431–2438.

[Deu97] Deutsche Forschungsgemeinschaft. *Empfehlungen der Kommission “Selbstkontrolle in der Wissenschaft”: Vorschläge zur Sicherung guter wissenschaftlicher Praxis.* `http://www.dfg.de/foerderung/grundlagen_rahmenbedingungen/gwp/`. 1997.

[DF83] M. DeGroot and S. E. Fienberg. “The Comparison and Evaluation of Forecasters”. In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 32.1/2 (1983), pp. 12–22.

[Dic45] L. R. Dice. “Measures of the amount of ecologic association between species”. In: *Ecology* 26.3 (1945), pp. 297–302.

[Die00] T. G. Dietterich. “Ensemble Methods in Machine Learning”. In: *First International Workshop on Multiple Classifier Systems (MCS), Cagliari, Italy.* 2000, pp. 1–15. DOI: `10.1007/3-540-45014-9_1`.

[Dij87] E.W. Dijkstra. “On the nature of computing science”. In: *Control Flow and Data Flow: concepts of distributed programming.* 1987, pp. 1–3.

[Dom99] P. Domingos. “MetaCost: a General Method for Making Classifiers Cost-Sensitive”. In: *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA.* 1999, pp. 155–164. DOI: `10.1145/312129.312220`.

[DP96] P. Domingos and M. Pazzani. “Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier”. In: *Proceedings of the 13th International Conference on Machine Learning (ICML), Bari, Italy.* 1996, pp. 105–112.

[Ehl12] J. Ehlers. “Self-Adaptive Performance Monitoring for Component-Based Software Systems”. PhD thesis. Christian-Albrechts-Universität zu Kiel, 2012.

[Ein23] A. Einstein. *Geometrie und Erfahrung.* Dover Books on Physics. 1923. ISBN: 9780486245119.

[ES03] D. M. Endres and J. E. Schindelin. “A new metric for probability distributions”. In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1858–1860.

[ESK03] L. Ertöz, M. Steinbach, and V. Kumar. “Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data”. In: *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM), San Francisco, CA.* 2003.

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR.* 1996, pp. 226–231.

[Fan+08] Yi Fang, Marc Friedman, Giri Nair, Michael Rys, and Ana-Elisa Schmid. “Spatial indexing in microsoft SQL server 2008”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Vancouver, BC.* 2008, pp. 1207–1216. DOI: 10.1145/1376616.1376737.

[Fär+10] I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. “On Using Class-Labels in Evaluation of Clusterings”. In: *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD 2010, Washington, DC.* 2010.

[FB74] R. A. Finkel and J. L. Bentley. “Quad trees. A data structure for retrieval on composite keys”. In: *Acta Informatica* 4.1 (1974), pp. 1–9. DOI: 10.1007/BF00288933.

[FCI05] E. Fanea, S. Carpendale, and T. Isenberg. “An interactive 3D integration of parallel coordinates and star glyphs”. In: *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS), Minneapolis, MN.* IEEE. 2005, pp. 149–156. DOI: 10.1109/INFOVIS.2005.5.

[Fis36] R. A. Fisher. “The Use of Multiple Measurements in Taxonomic Problems”. In: *Annals of Eugenics* 7 (1936), pp. 179–188.

[Flo95] J. Flower. “The structure of organized change: a conversation with Kevin Kelly”. In: *The Healthcare Forum Journal.* Vol. 38. 1. 1995.

[FLS63] R. P. Feynman, R. B. Leighton, and M. Sands. *Feynman lectures on physics. vol. 1: Mainly mechanics, radiation and heat.* Addison-Wesley, 1963.

[For65] E. W. Forgy. “Cluster analysis of multivariate data: efficiency versus interpretability of classifications”. In: *Biometrics* 21 (1965), pp. 768–769.

[FWV07] D. François, V. Wertz, and M. Verleysen. “The Concentration of Fractional Distances”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.7 (2007), pp. 873–886. DOI: 10.1109/TKDE.2007.1037.

[GA11] J. Ghosh and A. Acharya. “Cluster ensembles”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.4 (2011), pp. 305–315. DOI: 10.1002/widm.32.

[Gao+10] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. “On Community Outliers and their Efficient Detection in Information Networks”. In: *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC.* 2010, pp. 813–822. DOI: 10.1145/1835804.1835907.

[GBS05] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. “The Amsterdam Library of Object Images”. In: *International Journal of Computer Vision* 61.1 (2005), pp. 103–112. DOI: 10.1023/B:VISI.0000042993.50813.60.

[Gea54] R. C. Geary. “The Contiguity Ratio and Statistical Mapping”. In: *The Incorporated Statistician* 5.3 (1954), pp. 115–146.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. "Similarity Search in High Dimensions via Hashing". In: *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), Edinburgh, Scotland.* 1999, pp. 518–529.

[Goo08] S. Goodman. "A Dirty Dozen: Twelve P-Value Misconceptions". In: *Seminars in Hematology.* Vol. 45. 3. 2008, pp. 135–140. DOI: `10.1053/j.seminhematol.2008.04.003`.

[GRS98] S. Guha, R. Rastogi, and K. Shim. "CURE: An efficient Clustering Algorithm for Large Databases". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA.* 1998, pp. 73–84. DOI: `10.1145/276304.276312`.

[Gru69] F. E. Grubbs. "Procedures for Detecting Outlying Observations in Samples". In: *Technometrics* 11.1 (1969), pp. 1–21.

[GT06] J. Gao and P.-N. Tan. "Converting Output Scores from Outlier Detection Algorithms into Probability Estimates". In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM), Hong Kong, China.* 2006, pp. 212–221. DOI: `10.1109/ICDM.2006.43`.

[Guo03] D. Guo. "Coordinating computational and visual approaches for interactive feature selection and multivariate clustering". In: *Information Visualization* 2.4 (2003), pp. 232–246. DOI: `10.1057/palgrave.ivs.9500053`.

[Gut84] A. Guttman. "R-Trees: A Dynamic Index Structure for Spatial Searching". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Boston, MA.* 1984, pp. 47–57. DOI: `10.1145/602259.602266`.

[HAK00] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. "What is the nearest neighbor in high dimensional spaces?" In: *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt.* 2000, pp. 506–515.

[Hal64] J. H. Halton. "Algorithm 247: Radical-inverse quasi-random point sequence". In: *Communications of the ACM* 7.12 (1964), pp. 701–702.

[Ham74] Frank R. Hampel. "The Influence Curve and Its Role in Robust Estimation". In: *Journal of the American Statistical Association* 69.346 (1974), pp. 383–393. ISSN: 01621459.

[Has+47] C. Hastings, F. Mosteller, J. W. Tukey, and C. P. Winsor. "Low moments for small samples: a comparative study of order statistics". In: *The Annals of Mathematical Statistics* 18.3 (1947), pp. 413–426.

[Haw80] D. Hawkins. *Identification of Outliers.* Chapman and Hall, 1980.

[Hel09] E. Hellinger. "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen." In: *Journal für die reine und angewandte Mathematik* 136 (1909), pp. 210–271.

[Hil91] D. Hilbert. "Ueber die stetige Abbildung einer Linie auf ein Flächenstück". In: *Mathematische Annalen* 38.3 (1891), pp. 459–460.

[HKN12] M. E. Houle, H. Kashima, and M. Nett. "Generalized Expansion Dimension". In: *ICDM Workshop Practical Theories for Exploratory Data Mining (PTDM).* 2012, pp. 587–594. DOI: 10.1109/ICDMW.2012.94.

[HM82] J. A. Hanley and B. J. McNeil. "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve". In: *Radiology* 143 (1982), pp. 29–36.

[HNB03] S.-B. Hong, W. Nah, and J.-H. Baek. "Abrupt Shot Change Detection Using Multiple Features and Classification Tree". In: *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Hong Kong, China.* 2003, pp. 553–560. DOI: 10.1007/978-3-540-45080-1_76.

[HNP09] A. Halevy, P. Norvig, and F. Pereira. "The unreasonable effectiveness of data". In: *International Journal of Intelligent Systems* 24.2 (2009), pp. 8–12.

[Hof87] P. Hoffman. "The man who loves only numbers". In: *Atlantic Monthly* 260.5 (1987), p. 60.

[Hos91] J. R. M. Hosking. *Fortran routines for use with the method of L-moments Version 3.03.* Tech. rep. RC17097. IBM Research Division, 1991.

[Hou+10] M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 482–500. DOI: 10.1007/978-3-642-13818-8_34.

[Hou+12] M. E. Houle, X. Ma, M. Nett, and V. Oria. "Dimensional Testing for Multi-Step Similarity Search". In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 299–308. DOI: 10.1109/ICDM.2012.91.

[Hou03] M. E. Houle. "Navigating Massive Data Sets via Local Clustering". In: *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC.* 2003, pp. 547–552. DOI: 10.1145/956750.956817.

[Hou08] M. E. Houle. "The Relevant-Set Correlation Model for Data Clustering". In: *Statistical Analysis and Data Mining* 1.3 (2008), pp. 157–176. DOI: 10.1002/sam.10013.

[Hou62] P. V. C. Hough. *Methods and Means for Recognizing Complex Patterns.* U.S. Patent 3069654. Dec. 1962.

[HRA11] Ying Hu, Siva Ravada, and Richard Anderson. "Geodetic Point-In-Polygon Query Processing in Oracle Spatial". In: *SSTD.* 2011, pp. 297–312. DOI: 10.1007/978-3-642-22922-0_18.

[HS05] M. E. Houle and J. Sakuma. "Fast Approximate Similarity Search in Extremely High-Dimensional Data Sets". In: *Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan.* 2005, pp. 619–630. DOI: 10.1109/ICDE.2005.66.

[HSD73] R. M. Haralick, K. Shanmugam, and I. Dinstein. "Textural features for image classification". In: *IEEE Transactions on Speech and Audio Processing* 3.6 (1973), pp. 610–623.

[Hu+12] Y. Hu, S. Ravada, R. Anderson, and B. Bamba. "Topological relationship query processing for complex regions in Oracle Spatial". In: *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), Redondo Beach, CA.* 2012, pp. 3–12. DOI: 10.1145/2424321.2424323.

[HWW85] J. R. M. Hosking, J. R. Wallis, and E. F. Wood. "Estimation of the generalized extreme-value distribution by the method of probability-weighted moments". In: *Technometrics* 27.3 (1985), pp. 251–261.

[IBM] IBM Informix. *IBM Informix Geodetic DataBlade Module User's Guide , Version 3.12.* http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.geod.doc/ids_geod_006.htm. URL: http://publib.boulder.ibm.com/infocenter/idshelp/v117/topic/com.ibm.geod.doc/ids_geod_006.htm.

[ID90] A. Inselberg and B. Dimsdale. "Parallel coordinates: a tool for visualizing multidimensional geometry". In: *Proceedings of the IEEE Visualization Conference (VIS).* 1990, pp. 361–378. DOI: 10.1109/VISUAL.1990.146402.

[IM98] P. Indyk and R. Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *Proceedings of the 30th annual ACM symposium on Theory of computing (STOC), Dallas, TX.* 1998, pp. 604–613. DOI: 10.1145/276698.276876.

[Ins09] A. Inselberg. *Parallel coordinates: visual multidimensional geometry and its applications.* Springer, 2009.

[IWS11] A. M. Ivanescu, M. Wichterich, and T. Seidl. "ClasSi: Measuring Ranking Quality in the Presence of Object Classes with Similarity Information". In: *Proceedings of the PAKDD Workshop on Quality Issues, Measures of Interestingness and Evaluation of Data Mining Models (QIMIE).* 2011. DOI: 10.1007/978-3-642-28320-8_16.

[Jin+06] W. Jin, A. K. H. Tung, J. Han, and W. Wang. "Ranking Outliers Using Symmetric Neighborhood Relationship". In: *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore.* 2006, pp. 577–593. DOI: 10.1007/11731139_68.

[JJ04] T. Jo and N. Japkowicz. "Class Imbalances versus Small Disjuncts". In: *ACM SIGKDD Explorations* 6.1 (2004), pp. 40–49. DOI: 10.1145/1007730.1007737.

[JKA01] M. V. Joshi, V. Kumar, and R. Agarwal. "Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements". In: *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM), San Jose, CA.* 2001, pp. 257–264. DOI: 10.1109/ICDM.2001.989527.

[JL84] W. B. Johnson and J. Lindenstrauss. “Extensions of Lipschitz mappings into a Hilbert space”. In: *Conference in Modern Analysis and Probability.* Vol. 26. Contemporary Mathematics. American Mathematical Society, 1984, pp. 189–206.

[Joh+06] J. Johansson, P. Ljung, M. Jern, and M. Cooper. “Revealing structure in visualizations of dense 2D and 3D parallel coordinates”. In: *Information Visualization* 5.2 (2006), pp. 125–136. DOI: 10.1057/palgrave.ivs.9500117.

[Joh00] M.L. Johnson. “Outliers and robust parameter estimation”. In: *Numerical Computer Methods, Part C.* Ed. by M.L. Johnson and L. Brand. Vol. 321. Methods in enzymology. Academic Press, 2000, pp. 417–424. DOI: 10.1016/S0076-6879(00)21206-8.

[JP73] R. A. Jarvis and E. A. Patrick. “Clustering Using a Similarity Measure based on Shared Near Neighbors”. In: *IEEE Transactions on Computers* C-22.11 (1973), pp. 1025–1034.

[Jr51] A. C. Cohen Jr. “Estimating parameters of logarithmic-normal distributions by maximum likelihood”. In: *Journal of the American Statistical Association* 46.254 (1951), pp. 206–212.

[JS02] N. Japkowicz and S. Stephen. “The class imbalance problem: A systematic study”. In: *Intelligent Data Analysis* 6 (2002), pp. 429–449.

[JTH01] W. Jin, A.K. Tung, and J. Han. “Mining Top-n Local Outliers in Large Databases”. In: *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA.* 2001, pp. 293–298. DOI: 10.1145/502512.502554.

[JW92] R.A. Johnson and D.W. Wichern. *Applied multivariate statistical analysis.* Prentice Hall, 1992. ISBN: 9780130417732.

[Kab11] A. Kabán. “On the distance concentration awareness of certain data reduction techniques”. In: *Pattern Recognition* 44.2 (2011), pp. 265–277. DOI: 10.1016/j.patcog.2010.08.018.

[KF94] I. Kamel and C. Faloutsos. “Hilbert R-tree: An improved R-tree using fractals”. In: *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago de Chile, Chile.* 1994, pp. 500–509.

[KJS97] R. Kurniawati, J. S. Jin, and J. A. Shepherd. “The SS+-tree: An improved index structure for similarity searches in a high-dimensional feature space”. In: *Proceedings of the 5th Storage and Retrieval for Media Databases (SPIE), San Jose, CA.* Vol. 3022. 1997, pp. 110–120.

[KK10] H. H. Kim and Y. H. Kim. “Toward a Conceptual Framework of Key-Frame Extraction and Storyboard Display for Video Summarization”. In: *Journal of the American Society for Information Science and Technology* 61.5 (2010), pp. 927–939. DOI: 10.1002/asi.21317.

[KKZ09] H.-P. Kriegel, P. Kröger, and A. Zimek. “Clustering High Dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation Clustering”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1 (2009), pp. 1–58. DOI: 10.1145/1497577.1497578.

[KKZ12] H.-P. Kriegel, P. Kröger, and A. Zimek. “Subspace Clustering”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.4 (2012), pp. 351–364. DOI: 10.1002/widm.1057.

[KLC06] Y. Kou, C.-T. Lu, and D. Chen. “Spatial Weighted Outlier Detection”. In: *Proceedings of the 6th SIAM International Conference on Data Mining (SDM), Bethesda, MD.* 2006.

[KLD07] Y. Kou, C.-T. Lu, and R. F. Dos Santos. “Spatial outlier detection: a graph-based approach”. In: *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Patras, Greece.* 2007, pp. 281–288. DOI: 10.1109/ICTAI.2007.169.

[Kle+11] W. Klement, P. Flach, N. Japkowicz, and S. Matwin. “Smooth Receiver Operating Characteristics (smROC) Curves”. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), Athens, Greece.* 2011, pp. 193–208. DOI: 10.1007/978-3-642-23783-6_13.

[KMB12] F. Keller, E. Müller, and K. Böhm. “HiCS: High Contrast subspaces for Density-Based Outlier Ranking”. In: *Proceedings of the 28th International Conference on Data Engineering (ICDE), Washington, DC.* 2012. DOI: 10.1109/ICDE.2012.88.

[KN97a] E. M. Knorr and R. T. Ng. “A unified approach for mining outliers”. In: *Proceedings of the conference of the Centre for Advanced Studies on Collaborative research (CASCON), Toronto, Canada.* 1997, pp. 11–23. DOI: 10.1145/782010.782021.

[KN97b] E. M. Knorr and R. T. Ng. “A Unified Notion of Outliers: Properties and Computation”. In: *Proceedings of the 3rd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Newport Beach, CA.* 1997, pp. 219–222. DOI: 10.1145/782010.782021.

[KN98] E. M. Knorr and R. T. Ng. “Algorithms for Mining Distance-Based Outliers in Large Datasets”. In: *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York City, NY.* 1998, pp. 392–403.

[KNT00] E. M. Knorr, R. T. Ng, and V. Tucanov. “Distance-based Outliers: Algorithms and Applications”. In: *The VLDB Journal* 8.3–4 (2000), pp. 237–253. DOI: 10.1007/s007780050006.

[Kol33] A.N. Kolmogorov. “Sulla determinazione empirica di una legge di distribuzione”. In: *Giornale dell’ Istituto Italiano degli Attuari* 4.1 (1933), pp. 83–91.

[KR95] R. E. Kass and A. E. Raftery. “Bayes factors”. In: *Journal of the American Statistical Association* 90.430 (1995), pp. 773–795.

[KRA02] R. K. V. Kothuri, S. Ravada, and D. Abugov. “Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI.* 2002, pp. 546–557. DOI: `10.1145/564691.564755`.

[Kri+08] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “A General Framework for Increasing the Robustness of PCA-based Correlation Clustering Algorithms”. In: *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China.* 2008, pp. 418–435. DOI: `10.1007/978-3-540-69497-7_27`.

[Kri+09a] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “LoOP: Local Outlier Probabilities”. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China.* 2009, pp. 1649–1652. DOI: `10.1145/1645953.1646195`.

[Kri+09b] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data”. In: *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Bangkok, Thailand.* 2009, pp. 831–838. DOI: `10.1007/978-3-642-01307-2_86`.

[Kri+11] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “Interpreting and Unifying Outlier Scores”. In: *Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.* 2011, pp. 13–24.

[Kri+12] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. “Outlier Detection in Arbitrarily Oriented Subspaces”. In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 379–388. DOI: `10.1109/ICDM.2012.21`.

[KS97] N. Katayama and S. Satoh. “The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Tucson, AZ.* 1997, pp. 369–380. DOI: `10.1145/253260.253347`.

[KST01] Peter Kunszt, Alexander Szalay, and Aniruddha Thakar. “The hierarchical triangular mesh”. In: *Mining the sky* (2001), pp. 631–637.

[KSZ08] H.-P. Kriegel, M. Schubert, and A. Zimek. “Angle-Based Outlier Detection in High-dimensional Data”. In: *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Las Vegas, NV.* 2008, pp. 444–452. DOI: `10.1145/1401890.1401946`.

[KSZ11] H.-P. Kriegel, E. Schubert, and A. Zimek. “Evaluation of Multiple Clustering Solutions”. In: *2nd MultiClust Workshop: Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with ECML PKDD 2011, Athens, Greece.* 2011, pp. 55–66.

[KW11] John Krygier and Denis Wood. *Making maps: a visual guide to map design for GIS.* Guilford Press, 2011.

[Lan01] D. Laney. *3D Data Management: Controlling Data Volume, Velocity and Variety.* META Group. 2001.

[LCK03] C.-T. Lu, D. Chen, and Y. Kou. "Algorithms for Spatial Outlier Detection". In: *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM), Melbourne, FL.* 2003, pp. 597–600. DOI: 10.1109/ICDM.2003.1250986.

[Lee+04] W. Lee, H. Kim, H. Kang, J. Lee, Y. Kim, and S. Jeon. "Video Cataloging System for Real-Time Scene Change Detection of News Video". In: *Proceedings of teh 10th International Workshop on Combinatorial Image Analysis (IWCIA), Auckland, New Zealand.* 2004, pp. 705–715.

[LEL97] S. T. Leutenegger, J. M. Edgington, and M. A. Lopez. "STR: A Simple and Efficient Algorithm for R-Tree Packing". In: *Proceedings of the 13th International Conference on Data Engineering (ICDE), Birmingham, UK.* 1997, pp. 497–506. DOI: 10.1109/ICDE.1997.582015.

[Lev44] K. Levenberg. "A method for the solution of certain problems in least squares". In: *The Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168.

[Lil67] H. W. Lilliefors. "On the Kolmogorov-Smirnov test for normality with mean and variance unknown". In: *Journal of the American Statistical Association* 62.318 (1967), pp. 399–402.

[LK05] A. Lazarevic and V. Kumar. "Feature Bagging for Outlier Detection". In: *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL.* 2005, pp. 157–166. DOI: 10.1145/1081870.1081891.

[LLC10] X. Liu, C.-T. Lu, and F. Chen. "Spatial Outlier Detection: Random Walk Based Approaches". In: *Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), San Jose, CA.* 2010, pp. 370–379. DOI: 10.1145/1869790.1869841.

[LLL01] S. Liao, M. A. Lopez, and S. T. Leutenegger. "High Dimensional Similarity Search With Space Filling Curves". In: *Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany.* 2001, pp. 615–622. DOI: 10.1109/ICDE.2001.914876.

[LLP07] L. J. Latecki, A. Lazarevic, and D. Pokrajac. "Outlier Detection with Kernel Density Functions". In: *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM), Leipzig, Germany.* 2007, pp. 61–75. DOI: 10.1007/978-3-540-73499-4_6.

[LM98] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer international series in engineering and computer science. Springer, 1998. ISBN: 9780792381983.

[Low+13] T. Low, C. Borgelt, S. Stober, and A. Nürnberger. “The Hubness Phenomenon: Fact or Artifact?” In: *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*. Ed. by C. Borgelt, M. Á. Gil, J. M. C. Sousa, and M. Verleysen. Studies in Fuzziness and Soft Computing. Springer Berlin / Heidelberg, 2013, pp. 267–278.

[LQ65] D. O. Loftsgaarden and C. P. Quesenberry. “A Nonparametric Estimate of a Multivariate Density Function”. In: *The Annals of Mathematical Statistics* 36.3 (1965), pp. 1049–1051.

[Luk87] Hrvoje Lukatela. “Hipparchus geopositioning model: An overview”. In: *Proceedings of the 8th International Symposium on Computer-Assisted Cartography, Baltimore, Maryland.* Vol. 8. 1987, pp. 87–96.

[MA08] A. G. Money and H. Agius. “Video summarisation: A conceptual framework and survey of the state of the art”. In: *Journal of Visual Communication and Image Representation* 19.2 (2008), pp. 121–143. DOI: `10.1016/j.jvcir.2007.04.002`.

[Mar63] D.W. Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the Society for Industrial and Applied Mathematics (SIAM)* 11.2 (1963), pp. 431–441.

[Mat08] J. Matoušek. “On variants of the Johnson–Lindenstrauss lemma”. In: *Random Structures & Algorithms* 33.2 (2008), pp. 142–156. DOI: `10.1002/rsa.20218`.

[ME67] A. H. Murphy and E. S. Epstein. “Verification of Probabilistic Predictions: A Brief Review”. In: *Journal of Applied Meteorology* 6.5 (1967), pp. 748–755.

[Mej12] F. Mejia. “Advanced Computing Methods for Knowledge Discovery and Prognosis in Acoustic Emission Monitoring”. PhD thesis. University of Miami, 2012.

[Mic] Microsoft Corporation. *Whitepaper “New Spatial Features in SQL Server 2012”*. `http://go.microsoft.com/fwlink/?LinkId=226407`. URL: `http://go.microsoft.com/fwlink/?LinkId=226407`.

[Mor50] P.A.P. Moran. “Notes on continuous stochastic phenomena”. In: *Biometrika* 37.1/2 (1950), pp. 17–23.

[Mor66] G. M. Morton. *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. Tech. rep. International Business Machines Co., 1966.

[MSS10] E. Müller, M. Schiffer, and T. Seidl. “Adaptive Outlierness for Subspace Outlier Ranking”. In: *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM), Toronto, ON, Canada.* 2010, pp. 1629–1632. DOI: `10.1145/1871437.1871690`.

[Mül+08] E. Müller, I. Assent, U. Steinhausen, and T. Seidl. “OutRank: ranking outliers in high dimensional data”. In: *Proceedings of the 24th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank), Cancun, Mexico.* 2008, pp. 600–603. DOI: `10.1109/ICDEW.2008.4498387`.

[Mül+12] E. Müller, I. Assent, P. Iglesias, Y. Mülle, and K. Böhm. “Outlier Ranking via Subspace Analysis in Multiple Views of the Data”. In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 529–538. DOI: 10.1109/ICDM.2012.112.

[Mur66] A. H. Murphy. “A Note on the Utility of Probabilistic Predictions and the Probability Score in the Cost-Loss Ratio Decision Situation”. In: *Journal of Applied Meteorology* 5.4 (1966), pp. 534–537.

[MW77] A. H. Murphy and R. L. Winkler. “Reliability of Subjective Probability Forecasts of Precipitation and Temperature”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 26.1 (1977), pp. 41–47.

[NAG10] H. V. Nguyen, H. H. Ang, and V. Gopalkrishnan. “Mining Outliers with Ensemble of Heterogeneous Detectors on Random Subspaces”. In: *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA), Tsukuba, Japan.* 2010, pp. 368–383. DOI: 10.1007/978-3-642-12026-8_29.

[NGA11] H. V. Nguyen, V. Gopalkrishnan, and I. Assent. “An Unbiased Distance-based Outlier Detection Approach for High-dimensional Data”. In: *Proceedings of the 16th International Conference on Database Systems for Advanced Applications (DASFAA), Hong Kong, China.* 2011, pp. 138–152. DOI: 10.1007/978-3-642-20149-3_12.

[Ngu+12] G. Nguyen, P. Franco, R. Mullot, and J.-M. Ogier. “Mapping high dimensional features onto Hilbert curve: Applying to fast image retrieval”. In: *ICPR12.* 2012, pp. 425–428.

[NM65] J.A. Nelder and R. Mead. “A simplex method for function minimization”. In: *CompuJ* 7.4 (1965), pp. 308–313.

[Oli98] D. Olive. *Applied Robust Statistics.* University of Minnesota (Preprint), 1998.

[ON10] J. Olivier and M.M. Norberg. “Positively skewed data: revisiting the Box-Cox power transformation”. In: *International Journal of Psychological Research* 3.1 (2010), pp. 69–78.

[Pap+03] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faloutsos. “LOCI: Fast Outlier Detection Using the Local Correlation Integral”. In: *Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India.* 2003, pp. 315–326. DOI: 10.1109/ICDE.2003.1260802.

[Pea01] K. Pearson. “On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.6 (1901), pp. 559–572.

[Pea90] G. Peano. “Sur une courbe, qui remplit toute une aire plane”. In: *Mathematische Annalen* 36.1 (1890), pp. 157–160. DOI: 10.1007/BF01199438.

[Péb08] P. Pébay. “Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments”. In: *Sandia Report SAND2008-6212, Sandia National Laboratories* (2008).

[PF97] F. Provost and T. Fawcett. "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions". In: *Proceedings of the 3rd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Newport Beach, CA*. 1997, pp. 43–48.

[Pie05] T. Pietraszek. "Optimizing Abstaining Classifiers using ROC Analysis". In: *Proceedings of the 22nd International Conference on Machine Learning (ICML), Bonn, Germany*. 2005, pp. 665–672. DOI: `10.1145/1102351.1102435`.

[Pla00] J. C. Platt. "Probabilities for SV Machines". In: *Advances in Large Margin Classifiers*. Ed. by A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans. MIT Press, 2000, pp. 61–74.

[Pos] PostGIS project. *PostGIS 2.0 Manual*. `http://postgis.net/docs/manual-2.0/`. URL: `http://postgis.net/docs/manual-2.0/`.

[PP12] N. Pham and R. Pagh. "A Near-linear Time Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data". In: *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Beijing, China*. 2012.

[PZ09] L. Pickup and A. Zisserman. "Automatic retrieval of visual continuity errors in movies". In: *Proceedings of the 8th ACM International Conference on Image and Video Retrieval (CIVR), Santorini, Greece*. 2009. DOI: `10.1145/1646396.1646406`.

[PZG06] Y. Pei, O. Zaïane, and Y. Gao. "An Efficient Reference-based Approach to Outlier Detection in Large Datasets". In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM), Hong Kong, China*. 2006, pp. 478–487. DOI: `10.1109/ICDM.2006.17`.

[Ras+11] A. Rasmussen, G. Porter, M. Conley, H.V. Madhyastha, R.N. Mysore, A. Pucher, and A. Vahdat. "TritonSort: a balanced large-scale sorting system". In: *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. 2011.

[RC92] P.J. Rousseeuw and C. Croux. "L_1-statistical analysis and related methods". In: ed. by Y. Dodge. North-Holland, 1992. Chap. Explicit Scale Estimators with High Breakdown Point, pp. 77–92. ISBN: 978-0444894441.

[RC93] P.J. Rousseeuw and C. Croux. "Alternatives to the median absolute deviation". In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1273–1283.

[RK04] B. Raskutti and A. Kowalczyk. "Extreme Re-balancing for SVMs: a case study". In: *ACM SIGKDD Explorations* 6.1 (2004), pp. 60–69. DOI: `10.1145/1007730.1007739`.

[RN88] J. L. Rodgers and W. A. Nicewander. "Thirteen Ways to Look at the Correlation Coefficient". In: *The American Statistician* 42.1 (1988), pp. 59–66.

[RNI09] M. Radovanović, A. Nanopoulos, and M. Ivanović. "Nearest neighbors in high-dimensional data: the emergence and influence of hubs". In: *Proceedings of the 26th International Conference on Machine Learning (ICML), Montreal, QC, Canada*. 2009, pp. 865–872. DOI: `10.1145/1553374.1553485`.

[RNI10a] M. Radovanović, A. Nanopoulos, and M. Ivanović. “Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data”. In: *Journal of Machine Learning Research* 11 (2010), pp. 2487–2531.

[RNI10b] M. Radovanović, A. Nanopoulos, and M. Ivanović. “On the Existence of Obstinate Results in Vector Space Models”. In: *Proceedings of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR), Geneva, Switzerland.* 2010, pp. 186–193. DOI: 10.1145/1835449.1835482.

[RRS00] S. Ramaswamy, R. Rastogi, and K. Shim. “Efficient algorithms for mining outliers from large data sets”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX.* 2000, pp. 427–438. DOI: 10.1145/342009.335437.

[San67] F. Sanders. “The Verification of Probability Forecasts”. In: *Journal of Applied Meteorology* 6.5 (1967), pp. 756–761.

[SB91] M.J. Swain and D.H. Ballard. “Color indexing”. In: *International Journal of Computer Vision* 7.1 (1991), pp. 11–32. DOI: 10.1007/BF00130487.

[SC04] P. Sun and S. Chawla. “On Local Spatial Outliers”. In: *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM), Brighton, UK.* 2004, pp. 209–216. DOI: 10.1109/ICDM.2004.10097.

[Sch+12] E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. “On Evaluation of Outlier Rankings and Outlier Scores”. In: *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA.* 2012, pp. 1047–1058.

[Sch12] B. Schneier. *Liars and Outliers: Enabling the Trust that Society Needs to Thrive.* Wiley, 2012. ISBN: 9781118239018.

[Sil86] B. W. Silverman. “Density estimation for statistics and data analysis”. In: *Monographs on statistics and applied probability.* Vol. 26. Chapman & Hall/CRC, 1986.

[Sim+13] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. “A survey on enhanced subspace clustering”. In: *Data Mining and Knowledge Discovery* 26.2 (2013), pp. 332–397. DOI: 10.1007/s10618-012-0258-x.

[Sim96] J. S. Simonoff. *Smoothing Methods in Statistics.* Springer Series in Statistics. Springer, 1996. ISBN: 9780387947167.

[Sin84] R.W. Sinnott. “Virtues of the haversine”. In: *Sky and telescope* 68 (1984), pp. 158–159.

[SJ91] S. J. Sheather and M. C. Jones. “A reliable data-based bandwidth selection method for kernel density estimation”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (1991), pp. 683–690.

[SLZ03] S. Shekhar, C.-T. Lu, and P. Zhang. “A Unified Approach to Detecting Spatial Outliers”. In: *GeoInformatica* 7.2 (2003), pp. 139–166. DOI: 10.1023/A:1023455925009.

[Smi48] N. Smirnov. “Table for estimating the goodness of fit of empirical distributions”. In: *The Annals of Mathematical Statistics* 19.2 (1948), pp. 279–281.

[Sør48] Thorvald Sørensen. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons”. In: *Biologiske Skrifter* 5.4 (1948), pp. 1–34.

[Spa89] K. A. Spackman. “Signal detection theory: Valuable tools for evaluating inductive learning”. In: *Proceedings of the 6th International Workshop on Machine Learning (ML), Ithaca, NY.* 1989, pp. 160–163.

[SS05] D.W. Scott and S.R. Sain. “Multi-dimensional density estimation”. In: *Handbook of Statistics* 24 (2005), pp. 229–261.

[SSB05] E. Schubert, S. Schaffert, and F. Bry. “Structure-Preserving Difference Search for XML Documents”. In: *Proceedings of the Extreme Markup Languages 2005 Conference, Montreal, Quebec, Canada.* 2005.

[SW65] S. S. Shapiro and M. B. Wilk. “An analysis of variance test for normality (complete samples)”. In: *Biometrika* 52.3/4 (1965), pp. 591–611.

[SZK12] E. Schubert, A. Zimek, and H.-P. Kriegel. “Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection”. In: *Data Mining and Knowledge Discovery* (2012). DOI: 10.1007/s10618-012-0300-z.

[SZK13] E. Schubert, A. Zimek, and H.-P. Kriegel. “Geodetic Distance Queries on R-Trees for Indexing Geographic Data”. In: *Proceedings of the 13th International Symposium on Spatial and Temporal Databases (SSTD), Munich, Germany.* 2013, pp. 146–164. DOI: 10.1007/978-3-642-40235-7_9.

[SZK14a] E. Schubert, A. Zimek, and H.-P. Kriegel. “Generalized Outlier Detection with Flexible Kernel Density Estimates”. In: *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA.* 2014.

[SZK14b] E. Schubert, A. Zimek, and H.-P. Kriegel. “Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection”. In: *Data Mining and Knowledge Discovery* 28.1 (2014), pp. 190–237. DOI: 10.1007/s10618-012-0300-z.

[SZM98] J. A. Shepherd, X. Zhu, and N. Megiddo. “Fast indexing method for multidimensional nearest-neighbor search”. In: *Proceedings of the Storage and Retrieval for Image and Video Databases VII (SPIE), San Jose, CA.* 1998, pp. 350–355. DOI: 10.1117/12.333854.

[Tat+11] A. Tatu, G. Albuquerque, M. Eisemann, P. Bak, H. Theisel, M. Magnor, and D. Keim. “Automated analytical methods to support visual exploration of high-dimensional data”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.5 (2011), pp. 584–597. DOI: 10.1109/TVCG.2010.242.

[Tat+12] A. Tatu, F. Maaß, I. Färber, E. Bertini, T. Schreck, T. Seidl, and D. A. Keim. “Subspace Search and Visualization to Make Sense of Alternative Clusterings in High-Dimensional Data”. In: *Proceedings of the 2012 IEEE Symposium on Visual Analytics Science and Technology (VAST), Seattle, WA*. 2012, pp. 63–72. DOI: 10.1109/VAST.2012.6400488.

[Ter08] T. B. Terriberry. *Computing higher-order moments online*. 2008. URL: http://people.xiph.org/~tterribe/notes/homs.html.

[Tra+00] Caetano Traina, Agma Traina, Bernhard Seeger, and Christos Faloutsos. “Slim-trees: High performance metric trees minimizing overlap between nodes”. In: *Proceedings of the 7th International Conference on Extending Database Technology (EDBT), Konstanz, Germany* (2000), pp. 51–65. DOI: 10.1007/3-540-46439-5_4.

[Tra+02] Caetano Traina Jr, Agma Traina, Christos Faloutsos, and Bernhard Seeger. “Fast indexing and visualization of metric data sets using slim-trees”. In: *IEEE Transactions on Knowledge and Data Engineering* 14.2 (2002), pp. 244–260. DOI: 10.1109/69.991715.

[TS92] G.R. Terrell and D.W. Scott. “Variable kernel density estimation”. In: *The Annals of Statistics* 20.3 (1992), pp. 1236–1265.

[TSK06] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

[Tuk62] J. W. Tukey. “The future of data analysis”. In: *The Annals of Mathematical Statistics* 33.1 (1962), pp. 1–67.

[Vin75] T. Vincenty. “Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations”. In: *Survey review* 23.176 (1975), pp. 88–93.

[VKG02] M. Vlachos, G. Kollios, and D. Gunopulos. “Discovering similar multidimensional trajectories”. In: *Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA*. 2002, pp. 673–684. DOI: 10.1109/ICDE.2002.994784.

[VM02] G. Valentini and F. Masulli. “Ensembles of Learning Machines”. In: *Proceedings of the 13th Italian Workshop on Neural Nets, Vietri, Italy*. 2002, pp. 3–22. DOI: 10.1007/3-540-45808-5_1.

[vM53] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. 3rd. Princeton University Press, 1953.

[VW11] S. Venkatasubramanian and Q. Wang. “The Johnson-Lindenstrauss Transform: An Empirical Study”. In: *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX) SIAM, San Francisco, CA*. 2011, pp. 164–173.

[Web85] M. Weber. “Wissenschaft als Beruf (1919)”. In: *Gesammelte Aufsätze zur Wissenschaftslehre*. 6th ed. Vol. 17. J. C. B. Mohr, 1985. Chap. XII. ISBN: 9783165449952.

[Wei04] G. M. Weiss. “Mining with Rarity: A Unifying Framework”. In: *ACM SIGKDD Explorations* 6.1 (2004), pp. 7–19. DOI: 10.1145/1007730.1007734.

[Wel62] B. P. Welford. “Note on a Method for Calculating Corrected Sums of Squares and Products”. In: *Technometrics* 4.3 (1962), pp. 419–420. DOI: 10.2307/1266577.

[Wes79] D. H. D. West. “Updating mean and variance estimates: an improved method”. In: *Communications of the ACM* 22.9 (1979), pp. 532–535. DOI: 10.1145/359146.359153.

[Win96] R. L. Winkler. “Scoring Rules and the Evaluation of Probabilities”. In: *TEST* 5.1 (1996), pp. 1–60.

[WJ94] P. Wand and C. Jones. *Kernel Smoothing*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994. ISBN: 9780412552700.

[WJ96] D. A. White and R. Jain. “Similarity indexing with the SS-tree”. In: *Proceedings of the 12th International Conference on Data Engineering (ICDE), New Orleans, LA.* 1996, pp. 516–523. DOI: 10.1109/ICDE.1996.492202.

[WLG97] R. Wegenkittl, H. Löffelmann, and E. Gröller. “Visualizing the behaviour of higher dimensional dynamical systems”. In: *Proceedings of the IEEE Visualization Conference (VIS), Phoenix, AZ.* IEEE. 1997, pp. 119–125. DOI: 10.1145/266989.267038.

[WMZ07] G. M. Weiss, K. McCarthy, and B. Zabar. “Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?” In: *Proceedings of the International Conference on Data Mining (DMIN), Las Vegas, NV.* 2007, pp. 35–41.

[WPT11] Y. Wang, S. Parthasarathy, and S. Tatikonda. “Locality Sensitive Outlier Detection: A ranking driven approach”. In: *Proceedings of the 27th International Conference on Data Engineering (ICDE), Hannover, Germany.* 2011, pp. 410–421. DOI: 10.1109/ICDE.2011.5767852.

[WSB98] R. Weber, H.-J. Schek, and S. Blott. “A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”. In: *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York City, NY.* 1998, pp. 194–205.

[Yag+11] J. Yagnik, D. Strelow, D. A. Ross, and R.-S. Lin. “The power of comparative reasoning”. In: *Proceedings of the 13th IEEE International Conference on Computer Vision.* 2011, pp. 2431–2438. DOI: 10.1109/ICCV.2011.6126527.

[Yan+03] J. Yang, M.O. Ward, E.A. Rundensteiner, and S. Huang. “Visual hierarchical dimension reduction for exploration of high dimensional datasets”. In: *Proc. Symp. Data Visualisation 2003.* 2003, pp. 19–28.

[YJF00] B. K. Yi, H. Jagadish, and C. Faloutsos. “Fast time sequence indexing for arbitrary Lp norms”. In: *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt.* 2000, pp. 385–394.

[ZE01] B. Zadrozny and C. Elkan. “Learning and Making Decisions when Costs and Probabilities are Both Unknown”. In: *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA.* 2001, pp. 204–213. DOI: 10.1145/502512.502540.

[ZE02] B. Zadrozny and C. Elkan. “Transforming Classifier Scores into Accurate Multiclass Probability Estimates”. In: *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Edmonton, AB.* 2002, pp. 694–699. DOI: 10.1145/775047.775151.

[Zha+04] J. Zhang, M. Lou, T. W. Ling, and H. Wang. “HOS-Miner: A System for Detecting Outlying Subspaces of High-dimensional Data”. In: *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada.* 2004, pp. 1265–1268.

[ZHJ09] K. Zhang, M. Hutter, and H. Jin. “A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data”. In: *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Bangkok, Thailand.* 2009, pp. 813–822. DOI: 10.1007/978-3-642-01307-2_84.

[Zim+13] A. Zimek, M. Gaudet, R. J. G. B. Campello, and J. Sander. “Subsampling for Efficient and Effective Unsupervised Outlier Detection Ensembles”. In: *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL.* 2013, pp. 428–436. DOI: 10.1145/2487575.2487676.

[Zim08] A. Zimek. “Correlation Clustering”. PhD thesis. Ludwig-Maximilians-Universität München, Munich, Germany, 2008.

[Zol86] V. M. Zolotarev. *One-dimensional stable distributions.* Vol. 65. Translations of Mathematical Monographs. American Mathematical Society, 1986.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. “BIRCH: An Efficient Data Clustering Method for Very Large Databases”. In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Montreal, QC, Canada.* 1996, pp. 103–114. DOI: 10.1145/233269.233324.

[ZSK12a] A. Zimek, E. Schubert, and H.-P. Kriegel. “A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data”. In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: 10.1002/sam.11161.

[ZSK12b] A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 12th International Conference on Data Mining (ICDM), Brussels, Belgium. 2012. DOI: 10.1109/ICDM.2012.9.

[ZSK13] A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Gold Coast, Australia. 2013.

Appendix

1 Weighted Pearson Correlation Coefficient

The Pearson product-moment correlation coefficient [Pea01] (also denoted as PPMCC, PCC or Pearson's r) is a classic statistical measure of linear dependence (correlation). It has been applied to various problems successfully [RN88], and is an interesting similarity measure for scores which are not well calibrated yet, as it includes an implicit standardization (see Section 5.4 for details on the use in this thesis). However, the original method is unweighted, and thus will likely be dominated by the non-outliers. In order to apply it to an imbalanced problem such as outlier detection, we need to employ a weighted covariance instead of the regular covariance, which mostly just involves weighted averages.

Let each object o_i be assigned a weight ω_i, and denote the set of all weights by $\omega := \{\omega_i\}$.

Weighted Pearson correlation can then canonically be defined by

$$\rho_\omega(X,Y) := \frac{\mathrm{Cov}_\omega(X,Y)}{\sigma_\omega(X)\sigma_\omega(Y)}. \tag{1}$$

In this formula, $\mathrm{Cov}_\omega(X,Y)$ is the weighted covariance and $\sigma_\omega(X) = \sqrt{\mathrm{Var}_\omega(X)}$ is the weighted standard deviation. By substituting the unweighted functions, we can obtain the classic Pearson product-moment correlation coefficient.

Weighted Pearson correlation, just like regular Pearson correlation, is in the range of $-1 \ldots +1$, and can be used as a dissimilarity function either via $1-\rho_\omega$ or via $1-\rho_\omega^2$. The key difference between these two is that in the latter, a perfect negative correlation ($\rho = -1$) also yields a distance of 0, which may be desirable in some situations.

Pearson correlation is not a metric distance. This is easy to see in particular as it obviously is not defined when a vector has variance 0. However, when the vectors have been z standardized (i.e. to zero mean and unit variance), it surprisingly turns out to be a variant of Euclidean distance. Obviously, the formula simplifies to $\rho(X,Y) = \mathrm{Cov}(X,Y)$. Euclidean distance on the other hand can be rewritten as follows:

$$\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} = \sqrt{\sum_{i=1}^{n} x_i + \sum_{i_1}^{n} y_i - 2\sum_{i_1}^{n} x_i \cdot y_i} = \sqrt{2n - 2\sum_{i_1}^{n} x_i \cdot y_i}$$
$$= \sqrt{2n(1-\rho(X,Y))}.$$

1.1 Weighted Covariance

The weighted mean and the weighted population variance are defined (using $\Omega := \sum_i \omega_i$) as:

$$E_\omega(X) := \frac{1}{\Omega} \sum_i \omega_i X_i \qquad \mathrm{Var}(X) \equiv \sigma^2_\omega(X) := \frac{1}{\Omega} \sum_i \omega_i (X_i - E_\omega(X))^2$$

Along this line, population covariance can be defined as:

$$\mathrm{Cov}_\omega(X,Y) := \frac{1}{\Omega} \sum_i \omega_i \left(X_i - E_\omega(X)\right)\left(Y_i - E_\omega(Y)\right) \tag{2}$$

In order to obtain an unbiased estimate when using a sample, we need to multiply both variance and covariance with the same bias correction factor:

$$\frac{\Omega^2}{\Omega^2 - \sum_i \omega_i^2} \tag{3}$$

It is easy to see that this definition satisfies the desired equality $\mathrm{Var}(X) = \mathrm{Cov}(X,X)$ and when $\forall_i \omega_i = 1$ and thus $\Omega = n$ it also yields the usual defintion of population variance.

1.2 Numerical Instabilities

For traditional variance and covariance, Steiner translation produces a formula known as "computational formula for the variance"[2] (since it allows computing the variance with a single pass over the data set):

$$\mathrm{Cov}_\omega(X,Y) = E[X \cdot Y] - E[X] \cdot E[Y]$$

Unfortunately (and ironically, given its nickname), this formula can suffer from catastrophic cancellation when computed with floating point numbers and $E[X \cdot Y] \approx E[X] \cdot E[Y]$ due to the squared terms. In these cases, a two-pass computation using Equation 2 will yield much higher accuracy. Other solutions to this problem have been proposed in the form of various online algorithms that can compute variances in a numerical stable way by avoiding computing the difference of two squared numbers. The most prominent result is the method by Welford [Wel62], which was extended to weighted variances by West [Wes79], to skewness and kurtosis by Terriberry [Ter08] and to arbitrary higher order moments by Pébay [Péb08].

Adapting these algorithms to weighted covariance turns out to be rather simple, because the algorithms are already performing weighted updates of their estimates (weighting the previous n observations with n and the new observation with 1). Note that depending on memory access and arithmetic performance, the two-pass algorithm described by Equation 2 can sometimes be faster and at least as accurate. However, the online algorithm can also be used to combine results from multiple partitions, useful e.g. in distributed computation on a Hadoop cluster.

[2]The Wikipedia article has since been renamed to "algebraic formula for the variance"

1.3 A Numerically Stable On-line Algorithm

The following algorithm is the canonical adaptation of the method proposed by Welford [Wel62] and its weighted variant by West [Wes79] to the problem of *weighted covariance*:

Input data:	
x_n	x value of nth observation
y_n	y value of nth observation
ω_n	weight of nth observation
Variables:	
$\hat{x}_n$	current mean x
$\hat{y}_n$	current mean y
Ω_n	current sum of weights
XX_n	current $\sum_{i=1}^{n} \omega_i (x_i - \hat{x}_n)(x_i - \hat{x}_n)$
XY_n	current $\sum_{i=1}^{n} \omega_i (x_i - \hat{x}_n)(y_i - \hat{y}_n)$
YY_n	current $\sum_{i=1}^{n} \omega_i (y_i - \hat{y}_n)(y_i - \hat{y}_n)$

Update step:

$$\begin{aligned}
\Omega_n &\leftarrow \Omega_{n-1} + \omega_n \\
\delta_x &\leftarrow x_n - \hat{x}_{n-1} \\
\delta_y &\leftarrow y_n - \hat{y}_{n-1} \\
\hat{x}_n &\leftarrow \hat{x}_{n-1} + \delta_x \cdot \omega_n / \Omega_n \\
\hat{y}_n &\leftarrow \hat{y}_{n-1} + \delta_y \cdot \omega_n / \Omega_n \\
\delta_{xx} &\leftarrow (x_n - \hat{x}_n) \cdot (x_n - \hat{x}_{n-1}) \\
\delta_{xy} &\leftarrow (x_n - \hat{x}_n) \cdot (y_n - \hat{y}_{n-1}) \\
\delta_{yy} &\leftarrow (y_n - \hat{y}_n) \cdot (y_n - \hat{y}_{n-1}) \\
XX_n &\leftarrow XX_{n-1} + \delta_{xx} \cdot \omega_n \\
XY_n &\leftarrow XY_{n-1} + \delta_{xy} \cdot \omega_n \\
YY_n &\leftarrow YY_{n-1} + \delta_{yy} \cdot \omega_n
\end{aligned}$$

It is correct to use once the current mean $\hat{x}_n$, and once the previous mean $\hat{y}_{n-1}$ [Wel62]. The variances and standard deviations of X and Y can be computed similarly in the same pass.

Result scaling: The population covariances and variances can then be computed by:

$$\begin{aligned}
\mathrm{Cov}_\omega(X, Y) &:= XY_n / \Omega_n \\
\mathrm{Var}_\omega(X) &:= XX_n / \Omega_n \\
\mathrm{Var}_\omega(Y) &:= YY_n / \Omega_n
\end{aligned}$$

In order to obtain sample covariance, use the bias correction factor from Equation 3.

2 Overview of Prior Published Parts

Several parts of this thesis have been previously published on appropriate conferences and journals. The entries are given ordered by pulication time.

2009

1. H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data". In: *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Bangkok, Thailand.* 2009, pp. 831–838. DOI: `10.1007/978-3-642-01307-2_86`
   Introduced the method SOD discussed in Section 5.2.1.
2. H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "LoOP: Local Outlier Probabilities". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China.* 2009, pp. 1649–1652. DOI: `10.1145/1645953.1646195`
   Introduced the method LoOP analyzed in Section 5.1.2 and was the first publication to discuss "probabilistic" outlier scores (Section 5.3).

2010

3. M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 482–500. DOI: `10.1007/978-3-642-13818-8_34`
   Is the original study on the curse of dimensionality and shared nearest neighbor distances, discussed in Section 4.4.

2011

4. H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Interpreting and Unifying Outlier Scores". In: *Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ.* 2011, pp. 13–24
   Discusses statistical scaling of scores for arbitary methods, and an early version of Section 5.3 as well as the first experiments with ensembles Chapter 7 (but not yet greedy ensemble pruning).
5. E. Achtert, A. Hettab, H.-P. Kriegel, E. Schubert, and A. Zimek. "Spatial Outlier Detection: Data, Algorithms, Visualizations". In: *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN.* 2011, pp. 512–516. DOI: `10.1007/978-3-642-22922-0_41`
   Analyzing spatial outlier detection showed the need to generalize the local outlier pattern to other data types, and thus lays the foundations for Chapter 6.

6. T. Bernecker, M. E. Houle, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. "Quality of Similarity Rankings in Time Series". In: *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN.* 2011, pp. 422–440. DOI: `10.1007/978-3-642-22922-0_25`

   Extends the study of shared nearest neighbors (Section 4.4) to time series.

2012

7. E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel. "On Evaluation of Outlier Rankings and Outlier Scores". In: *Proceedings of the 12th SIAM International Conference on Data Mining (SDM), Anaheim, CA.* 2012, pp. 1047–1058

   Introduced the evaluation measure discussed in Section 5.4 as well as an early version of greedy ensemble construction in Chapter 7.
8. A. Zimek, E. Schubert, and H.-P. Kriegel. "A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: `10.1002/sam.11161`

   The survey of high-dimensional outlier detection contributed substantially to the understanding of high-dimensional data in Chapter 4.
9. H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier Detection in Arbitrarily Oriented Subspaces". In: *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium.* 2012, pp. 379–388. DOI: `10.1109/ICDM.2012.21`

   The method COP introduced in Section 5.2.2 is an extension of the original COP [Zim08, Chapter 18], but with major changes.
10. A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 12th International Conference on Data Mining (ICDM), Brussels, Belgium. 2012. DOI: `10.1109/ICDM.2012.9`

    Overlaps with Chapter 4 and contains material from [ZSK12a].
11. E. Schubert, A. Zimek, and H.-P. Kriegel. "Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection". In: *Data Mining and Knowledge Discovery* (2012). DOI: `10.1007/s10618-012-0300-z`

    This "online-first" publication has since been published in a printed issue, using the same DOI but a different date:

    E. Schubert, A. Zimek, and H.-P. Kriegel. "Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection". In: *Data Mining and Knowledge Discovery* 28.1 (2014), pp. 190–237. DOI: `10.1007/s10618-012-0300-z`

    Contains major portions of Chapter 6.

2013

12. A. Zimek, E. Schubert, and H.-P. Kriegel. *Outlier Detection in High-Dimensional Data.* Tutorial at the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Gold Coast, Australia. 2013
    A later version of the previous tutorial [ZSK12b]; Chapter 4.
13. E. Achtert, H.-P. Kriegel, E. Schubert, and A. Zimek. "Interactive Data Mining with 3D-Parallel-Coordinate-Trees". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD), New York City, NY.* 2013, pp. 1009–1012. DOI: `10.1145/2463676.2463696`
    The visualizations presented in Section 4.6.
14. E. Schubert, A. Zimek, and H.-P. Kriegel. "Geodetic Distance Queries on R-Trees for Indexing Geographic Data". In: *Proceedings of the 13th International Symposium on Spatial and Temporal Databases (SSTD), Munich, Germany.* 2013, pp. 146–164. DOI: `10.1007/978-3-642-40235-7_9`
    The indexing approaches presented in Section 8.3.

2014

15. E. Schubert, A. Zimek, and H.-P. Kriegel. "Generalized Outlier Detection with Flexible Kernel Density Estimates". In: *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA.* 2014
    This publication is based on material from Section 5.1.3 and Chapter 9.

Other Publications, not Part of the Thesis

The following was also published as part of my university career, but have not found explicit mentioning in the thesis since they focus on other topics such as XML and cluster analysis.

1. E. Schubert, S. Schaffert, and F. Bry. "Structure-Preserving Difference Search for XML Documents". In: *Proceedings of the Extreme Markup Languages 2005 Conference, Montreal, Quebec, Canada.* 2005
2. E. Achtert, T. Bernecker, H.-P. Kriegel, E. Schubert, and A. Zimek. "ELKI in Time: ELKI 0.2 for the Performance Evaluation of Distance Measures for Time Series". In: *Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD), Aalborg, Denmark.* 2009, pp. 436–440. DOI: `10.1007/978-3-642-02982-0_35`
3. T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. "Subspace Similarity Search Using the Ideas of Ranking and Top-k Retrieval". In: *Proceedings of the 26th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank), Long Beach, CA.* 2010, pp. 4–9. DOI: `10.1109/ICDEW.2010.5452771`

4. T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. “Subspace Similarity Search: Efficient k-NN Queries in Arbitrary Subspaces”. In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany.* 2010, pp. 555–564. DOI: 10.1007/978-3-642-13818-8_38
5. I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. “On Using Class-Labels in Evaluation of Clusterings”. In: *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD 2010, Washington, DC.* 2010
6. E. Achtert, H.-P. Kriegel, L. Reichert, E. Schubert, R. Wojdanowski, and A. Zimek. “Visual Evaluation of Outlier Detection Models”. In: *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA), Tsukuba, Japan.* 2010, pp. 396–399. DOI: 10.1007/978-3-642-12098-5_34
7. H.-P. Kriegel, E. Schubert, and A. Zimek. “Evaluation of Multiple Clustering Solutions”. In: *2nd MultiClust Workshop: Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with ECML PKDD 2011, Athens, Greece.* 2011, pp. 55–66
8. E. Achtert, S. Goldhofer, H.-P. Kriegel, E. Schubert, and A. Zimek. “Evaluation of Clusterings – Metrics and Visual Support”. In: *Proceedings of the 28th International Conference on Data Engineering (ICDE), Washington, DC.* 2012, pp. 1285–1288. DOI: 10.1109/ICDE.2012.128
9. X. H. Dang, I. Assent, R. T. Ng, A. Zimek, and E. Schubert. “Discriminative Features for Identifying and Interpreting Outliers”. In: *Proceedings of the 30th International Conference on Data Engineering (ICDE), Chicago, IL.* 2014

Acknowledgements

My deepest gratitude to my supervisor, Prof. Dr. Hans-Peter Kriegel as well as my senior colleagues Dr. Arthur Zimek and Dr. Peer Kröger, who mentored me throughout my research, and whose experience was of immense benefit to me when publishing my results; but it was also them who encouraged me to pursue a career in research and gave me the opportunity to do so by providing the position.

My time here at the university has been an inspiring, productive and creative time. This is due to the support of the whole research group here in Munich: Dr. Elke Achtert, Dr. Thomas Bernecker, Dr. Franz Graf, Dr. Tobias Emrich, Susanne Grienberger, Matthias Jakob, Gregor Jossé, Prof. Dr. Hans-Peter Kriegel, Dr. Peer Kröger, Franz Krojer, Markus Mauder, Johannes Niedermayer, Dr. Eirini Ntoutsi, Dr. Marisa Petri, Dr. Matthias Renz, Klaus Arthur Schmid, Dr. Matthias Schubert, Prof. Dr. Volker Tresp, Florian Verhein, Michael Weiler, Dr. Arthur Zimek and Andreas Züfle which all contributed to a great working, teaching and research environment.

Unfortunately I did not have the chance to collaborate with all of them on scientific publications; but at least with some of them I did. I also have to thank my coauthors Prof. Mike E. Houle, for the interesting and inspiring collaboration on high-dimensional data; Ines Färber, Dr. Stephan Günnemann, Dr. Emmanuel Müller and Prof. Dr. Thomas Seidl for the collaboration on cluster evaluation, and their input on outlier detection. Prof. Ricardo J. G. B. Campello and Prof. Mario A. Nascimento for their scientific input each time we met and support at the ABBY-Net summer school in Canada.

There are too many students to name all of them, that helped me with my research; I hope I have been a good mentor for your thesises in return. Dr. Elke Achtert, Dr. Arthur Zimek for starting the ELKI project, which allowed me to reuse much of their code, extend it, and make exhaustive comparisons for many algorithms. Susanne Grienberger took good care of the administrative work, while Franz Krojer provided me with all the hardware, network and computing needs I could imagine; even a small Hadoop cluster I could not yet put to full use due to the lack of large enough and still interesting data.

I also have to thank Prof. Dr. Jörg Sander for his interest in my work, and for agreeing to act as second referee on this thesis, and the interesting (although too rare) discussions we have had on the subject of outlier detection and clustering when we met. I hope that we will have a chance to collaborate in the future.

My teaching duties for the university – tutoring classes and mentoring students – have been an important experience for me, that I do not want to miss anymore. When you explain approaches to a group of students, you have to understand them in more detail than they will need to know, and I must admit that I have often understood all the details only when I had to prepare a new exercise and prepare a detailed solution for it. When reviewing literature or lecture notes, we tend to skip over the tiny details that can make a huge difference. The only other approach to learn methods at this detail is to implement them yourself. While Arthur Zimek pushed me repeatedly to add new functionality to ELKI for comparison with our approaches, which led to a lot of work, but it was all very insightful because this requires you to go into all the details of the methods.

Computer science is not the only thing I have been teaching the last years. I spent a considerate amount of time in the evenings teaching swing dancing (mostly Lindy Hop). I can only express my deepest gratitude to all the dancers I have met, because they helped me keep balance in life, and clear my mind after a long day of thinking. Dancers, fortunately, are particularly uncomplicated: you meet, you dance, and then you have new friends. Often inspiration comes when you are least focused, and I'm sure that dancing helped me a lot; but this is also where I learned teaching. Christine von Scheidt deserves credit for teaching me not only to dance, but also to teach and relax when presenting in front of people.

Finally, but not least, there is also my family and girlfriend who always gave me care, love and a home to return to and relax; but who also tolerated me working late, and sometimes even on weekends when I could not let go. Here I learned to appreciate every day, even when the program code might not yet work as desired. Fortunately, I nevertheless managed to visit my parents occasionally, and often got to see my nieces and nephews there, too.

Erich Schubert
Munich, October 2013.