UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
NEURO- UND BIOINFORMATIK

Representation Learning: From Feature Weighting to Invariance

Jens Hocke

Dissertation

Universität zu Lübeck
Institut für Neuro- und Bioinformatik

Lübeck 2016

From the Institute for Neuro- and Bioinformatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. Thomas Martinetz

Representation Learning: From Feature Weighting to Invariance

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by

Jens Hocke
from Winsen/Luhe (Germany)

Lübeck 2016

Jens Hocke
Institute for Neuro- and Bioinformatics
Universität zu Lübeck
Ratzeburger Allee 160
23538 Lübeck, Germany

Board of examiners

First referee:	Prof. Dr. rer. nat. Thomas Martinetz
Second referee:	Prof. Dr.-Ing. Alfred Mertins
Chairman:	Prof. Dr. rer. nat. Stefan Fischer

Date of oral examination: 15.11.2016

Approved for printing. Lübeck, 15.11.2016

Zusammenfassung

Es werden immer größere Datenmengen angesammelt. In vielen Anwendungen sind diese Datenmengen so groß geworden, dass eine automatische Analyse nötig ist. Unser Ziel hier ist die Verbesserung von Klassifikationsraten durch eine Änderung der Repräsentation der Daten. Es werden neue Methoden zur linearen Transformation der Daten ebenso wie neue Architekturen zum Lernen invarianter Repräsentationen beschrieben. Für die invarianten Repräsentationen werden auch Zusammenhänge zwischen einer Theorie zur Invarianz und existierenden Methoden gezeigt. Zusätzlich haben wir ein Modell für die Repräsentationen eines sehr frühen Teils des menschlichen visuellen Systems entwickelt, um ein Wahrnehmungsphänomen zu reproduzieren.

Zunächst betrachten wir den k-NN Klassifikator, welcher wegen seiner Flexibilität und Einfachheit beliebt wurde. Ein Nachteil des Standard-k-NNs ist das festgelegte Distanzmaß. Dieses macht ihn abhängig von der initialen Skalierung der Daten. Um dieses Problem zu beheben, lernen wir lineare Transformationen, um die Daten vorzuverarbeiten. Eine Methode, die wir einführen, skaliert die Dimensionen der Daten entsprechend ihrer Relevanz, wobei auch irrelevante Dimensionen gefunden werden. Die andere Methode lernt eine vollständige lineare Transformation, welche es erlaubt weitere statistische Eigenschaften der Daten zu berücksichtigen. Beide Methoden werden auf künstlichen Daten getestet und dann auf echten Daten untersucht, wo sie gut im Vergleich zu anderen Methoden abschneiden. Im Gegensatz zu den anderen Methoden sind sie unabhängig von der initialen Skalierung der Dimensionen.

Danach betrachten wir Transformationen von Objekten in Bildern und versuchen Repräsentationen zu lernen, die invariant zu diesen Transformationen sind. Wir bauen bei unserer Arbeit auf Tomaso Poggios i-Theorie auf, die eine mögliche Erklärung gibt, wie Invarianz im menschlichen visuellen System entsteht. Wir zeigen Verbindungen zwischen dieser Theorie und bestehenden Methoden zum Lernen invarianter Repräsentationen auf. Dann führen wir eine neue Lernmethode für Repräsentationen ein, dessen Architektur direkt aus der

i-Theorie abgeleitet wurde. Des Weiteren passen wir ein "Convolutional Network" an, indem sich langsam ändernde Ausgaben für Bildsequenzen begünstigt werden, um invariante Repräsentationen zu lernen. In Experimenten testen wir die Invarianzeigenschaften dieser Methoden.

Außer diesen Untersuchungen zum Lernen von Repräsentationen wird auch ein neues Modell für einen frühen Teil des menschlichen visuellen Systems beschrieben. Es modelliert die visuelle Verarbeitung im Auge und ermöglicht die Rekonstruktion eines Bildes aus dem Ausgabesignal unter Zuhilfenahme von Methoden des Compressed Sensing. Bei der Rekonstruktion entstehen Bildfehler ähnlich dem Crowding, einem Defizit visueller Wahrnehmung. Diese Bildartefakte untersuchen wir mit Bildern von Buchstaben, wie es auch oft in psychologischen Wahrnehmungsexperimenten gemacht wird.

Abstract

An ever increasing amount of data is collected. In many domains it has become so large that an automatic analysis is mandatory. We focus on improving classification results by changing the representation of the input data. New linear representation learning methods as well as new architectures for learning invariant representations are described. For the invariant representations connections between a theory of invariance and existing methods for learning invariant representations are drawn. Additionally, we developed a model for the representations in the early human visual system to reproduce a cognitive phenomenon.

First, we consider the k-NN classifier, which has become popular due to its flexibility and simplicity. A drawback of the standard k-NN is a fixed distance measure. This makes it dependent on the initial scaling of the data. To alleviate this drawback, we learn linear transformations to preprocess the data. One method we introduce rescales the data dimensions according to their relevance and also finds the irrelevant dimensions in this process. The other method learns a full linear transformation, which allows this method to take more statistical regularities of the data into account. Both methods are inspected on artificial data and then evaluated on real-world data, where they compare well with competing methods. In contrast to other methods they are independent from the initial scaling of the data dimensions.

Second, we study transformation of objects in images and try to find a representations invariant to these transformations. We base our work on Tomaso Poggio's i-Theory, which gives a possible explanation how invariance emerges in the human visual system. We explain connections between this theory and existing methods for learning invariant representations. Then we introduce a new representation learning method with a network architecture directly derived from the i-Theory. Furthermore, we adapt a convolutional network to learn an invariant representation by encouraging slowly changing outputs for movie data. Experimentally, both methods are tested for their invariance properties.

Besides these investigations into representation learning also a new model for

the representations in the early human visual system is presented. It models visual processing in the eye and allows to reconstruct an image from the output signal using methods from the compressed sensing framework. From this strategy artifacts similar to crowding, i.e., a deficit of visual perception, arise. These effects are analyzed using images of letters, as it is often done in psychological cognition experiments.

List of Publications

[1] J. Hocke and T. Martinetz, "Learning Transformation Invariance from Global to Local," in *Workshop New Challenges in Neural Computation 2015* (B. Hammer and T. Villmann, eds.), vol. 03/2015 of *Machine Learning Reports*, pp. 16–24, 2015.

[2] J. Hocke and T. Martinetz, "Maximum Distance Minimization for Feature Weighting," *Pattern Recognition Letters*, vol. 52, pp. 48–52, 2015.

[3] J. Hocke and T. Martinetz, "Learning Transformation Invariance for Object Recognition," in *Workshop New Challenges in Neural Computation 2014* (B. Hammer and T. Villmann, eds.), vol. 02/2014 of *Machine Learning Reports*, pp. 20–25, 2014.

[4] J. Hocke and T. Martinetz, "Global Metric Learning by Gradient Descent," in *Artificial Neural Networks and Machine Learning - ICANN 2014 - 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15-19, 2014. Proceedings* (S. Wermter, C. Weber, W. Duch, T. Honkela, P. D. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, eds.), vol. 8681 of *Lecture Notes in Computer Science*, pp. 129–135, Springer, 2014.

[5] J. Hocke and T. Martinetz, "Application of Maximum Distance Minimization to Gene Expression Data," in *Workshop New Challenges in Neural Computation 2013* (B. Hammer, T. Martinetz, and T. Villmann, eds.), vol. 02/2013 of *Machine Learning Reports*, pp. 6–7, 2013. Short Paper.

[6] J. Hocke and T. Martinetz, "Feature Weighting by Maximum Distance Minimization," in *Artificial Neural Networks and Machine Learning - ICANN 2013 - 23rd International Conference on Artificial Neural Networks, Sofia, Bulgaria, September 10-13, 2013. Proceedings* (V. Mladenov, P. D. Koprinkova-Hristova, G. Palm, A. E. P. Villa, B. Appollini, and N. Kasabov, eds.), vol. 8131 of *Lecture Notes in Computer Science*, pp. 420–425, Springer, 2013.

[7] J. Hocke and T. Martinetz, "Experience in Training (Deep) Multi-Layer Perceptrons to Classify Digits," in *Workshop New Challenges in Neural Computation 2012* (B. Hammer and T. Villmann, eds.), vol. 03/2012 of *Machine Learning Reports*, pp. 113–115, 2012. Short Paper.

[8] J. Hocke, K. Labusch, E. Barth, and T. Martinetz, "Sparse Coding and Selected Applications," *KI - Künstliche Intelligenz*, vol. 26, no. 4, pp. 349–355, 2012.

[9] J. Hocke, M. Dorr, and E. Barth, "A compressed sensing model of peripheral vision," in *Human Vision and Electronic Imaging XVII* (B. E. Rogowitz, T. N. Pappas, and H. de Ridder, eds.), vol. 8291, pp. 82910Z–82910Z–7, Proceedings of SPIE, 2012.

[10] J. Hocke, E. Barth, and T. Martinetz, "Application of non-linear transform coding to image processing," in *Human Vision and Electronic Imaging XVII* (B. E. Rogowitz, T. N. Pappas, and H. de Ridder, eds.), vol. 8291, pp. 829105–829105–8, Proceedings of SPIE, 2012.

[11] J. Hocke, T. Martinetz, and E. Barth, "Image Deconvolution with Sparse Priors," in *Workshop New Challenges in Neural Computation 2011* (B. Hammer and T. Villmann, eds.), vol. 05/2011 of *Machine Learning Reports*, p. 9, 2011. Abstract.

Contents

1. Introduction

> Everything changes and nothing stands still.
>
> Heraclitus

Data is collected almost everywhere: from scientific measurements of the radiation from distant galaxies, over commercial recording of online shop usage for recommendation systems, to personal tracking of the footsteps made. While large amounts of data are collected and can be stored, they are too large for interpretation by humans. Therefore, computer based approaches are developed. Interpretation may involve assigning labels to data points, finding special data points, or finding trends in the data.

Depending on the data, these tasks can be very challenging. Often, the relevant labels and properties can not be measured directly. Usually, they are inferred indirectly from sets of values that can be measured. However, these values will only partially reflect the labels and properties of interest. Most likely many irrelevant properties are captured by these values. For example in a medical diagnosis health can not be measured directly, it is inferred from several variables. There may be measurements on heart rate, blood pressure, and concentrations of antibodies in the blood available. But these measurements vary from person to person and can also be influenced by the amount of recent physical activity. It is likely in many scenarios, not only in this medical, that one of the underlying variables (physical activity) changes over time and influences many measurements. Therefore, it often seems like every measurement changes over time.

In the last decades considerable effort has been devoted to handle such data and extract useful information. For assigning labels to data points, a process called classification, methods like k-Nearest Neighbors [1], Support Vector Machines [2] and Deep Neural Networks [3] have been developed. They all use training data, a set of data points with known labels, in order to label unknown data. Each of these methods is performing well in various applications, but each also has different disadvantages. k-Nearest Neighbors is simple and the results

are easy to interpret. However, the classification performance depends on the initial scaling of the data dimensions. In contrast, Support Vector Machines and Deep Neural Networks do not depend on the initial data scaling. Instead, it is hard to analyze why a label was assigned to a data point[1]. A problem for all classifiers is the size of the training data set. Particularly, if the data dimension is large, a complex classifier is needed. But, to achieve good error rates on unknown data, the number of training samples needed grows exponentially with the complexity of the classifier, which is known as the curse of dimensionality.

Representations can help to bypass these problems. Often, they are applied to reduce the data dimension by filtering out the irrelevant information. This allows using classifiers with lower complexity. In addition to the application of representations for improving classifiers, it is also possible to process the data to better fit the human cognitive capabilities, thus allowing interpretation by humans. Early, but still wide spread are predefined representations such as Fourier-Transformation, wavelets [4, 5], and SIFT-Features [6]. These representations work well for many tasks. To find a good representation for arbitrary inputs without hand crafting, representations can also be learned. Popular are statistical methods such as Principal Component Analysis [7], Independent Component Analysis [8], and Sparse Coding [9] or Linear Discriminant Analysis [10].

In this thesis, I examine two specific applications of representation learning. Using linear transformations for the representations, I adapt data in order to eliminate the dependence of classifiers like k-Nearest Neighbors on the initial scaling. Thus, the range of possible applications for these classifiers is extended. Here, I contribute two new and robust representation learning methods [11–14].

Second, representation learning for obtaining invariance to image transformations is investigated. Transformations of objects cause highly entangled boundaries between their representations in the pixel space. This is a problem that can be approached using a complex classifier and a huge amount of training data. To handle all transformations that can naturally occur in the image domain the amount of labeled training data needed seems infeasible. I show the close relations between existing methods and the i-Theory [15], a theoretical framework for invariant representations. Then based on recent theoretical findings I propose a new network architecture in order to learn invariant representations [16]. Furthermore, I investigate a network architecture designed to achieve invari-

[1] Assuming the SVM uses a kernel.

ance to multiple transformations simultaneously and that can be trained using unlabeled movie data [17].

Besides this research on representation learning, I studied a phenomenon occurring in human visual perception. Visual object recognition by the human brain in most tasks performs much more reliable than any machine learning method. According to a well accepted hypothesis visual processing in the brain uses several layers of representation, where the input signal is transformed from layer to layer to more abstract representations. Since this approach works so well, it would be useful to understand how these representations are computed. But it is hard to observe this directly. Fortunately, there are visual phenomena showing deficits in the visual system. These deficits may indirectly reveal mechanisms of the visual system or at least restrict the set of possible models.

One visual phenomenon is crowding: an object that can be recognized in solitude becomes unrecognizable in the presence of other objects surrounding it. I propose a new model of crowding using concepts from compressed sensing [18]. It reproduces important properties of crowding using image representations as prior.

In short, the contributions of this thesis are four different representation learning methods including their evaluation, new insights into the connections between invariance learning methods, and a new model for visual crowding.

After this introduction representation learning and related concepts are introduced in more detail in Chapter 2. Then the first main part on linear representation learning follows. It provides an overview on the topic (Chapter 3) and presents my new methods in the chapters 4 and 5. The second part is on transformation invariant representation learning. Important methods and theoretical concepts are summarized and connections to the i-Theory are presented (Chapter 6). Based on theoretical findings a new architecture is introduced in Chapter 7, while Chapter 8 shows my slowness based architecture. The last part is on crowding. Important concepts and models are described in Chapter 9. Followed by the new crowding model in Chapter 10. While each chapter presenting experimental findings contains a discussion, the final chapter provides a broader discussion of all results.

2. Basics

In the introduction we have seen that more and more data is collected in a diverse set of domains. Computational approaches are required to interpret the data, and classifiers in combination with suitable data representations are one essential solution. Here, we introduce terminology and algorithms essential for this thesis.

First, we have a closer look at classification methods, their training process, and under which general conditions they work well and, respectively, which conditions cause problems. Then representations in their different facets are introduced, and we describe how they are often combined with classifiers in order to improve the classification performance. Finally, some basic tools for creating and training representations are shown. This is by no means a comprehensive introduction to the methods. For further insights we refer the reader to the text books [19, 20].

2.1. Classification

Classification refers to the process of assigning class labels to data points, for example a data point may contain several medical measurements and the labels may be different illnesses. In general these labels are used to indicate that data points with the same label belong to the same class. Labels can algorithmically be assigned using classifiers. Mathematically, a classifier is a set of input-output functions $\mathcal{C}$. Each of these functions, identified by P, provides a mapping

$$y = \mathcal{C}_P(\mathbf{x}) \tag{2.1}$$

from some data point $\mathbf{x}$ to the label y. Often integers are used as labels. To get good results for the classification, a classifier providing a suitable set of input-output functions as well as a good input-output function needs to be selected. Suppose the classifier is predetermined, P is then usually estimated based on a set of known data point label tuples $(y_i, \mathbf{x}_i),\ i = 1, 2, \ldots, I$, which are referred to

as training data. This estimation process is named supervised training. Ideally, now the classifier can correctly assign labels to data points it has not seen previously. This, however, is not always true. To measure how well the classifier works, it is tested on data points which were not presented in the training phase. These tuples are referred to as test data. The ratio between correctly classified data points and the total number of datapoints

$$E = \frac{1}{I} \sum_{i} L(y_i, \mathcal{C}_P(\mathbf{x}_i)) \tag{2.2}$$

is called error rate, a wide spread measure for the classification quality. Here, $L(y_i, c(\mathbf{x}_i))$ is a loss function, which returns one if $\mathbf{x}_i$ was misclassified and zero else. If the error rates on the training data and the test data are similar, the classifier is said to generalize well.

There are two main factors which influence the error rate and generalization. First, the classifier needs suitable input-output functions to separate the classes, and, second, the training data needs to be a sufficiently large, representative sample from the data distribution. Some classifiers are versatile, i.e. they can be adapted to many different class boundaries. These classifiers, which are called complex, will have low error rates on many training sets. However, they need a lot of training data to generalize well, whereas classifiers with low complexity require much less training samples for good generalization. This should be considered when selecting a classifier. For a mathematically precise account of these properties consider [2].

Here, we briefly introduce two popular classifiers, the linear classifier and k-nearest neighbors [1]. Different instances of linear classifiers such as perceptrons [21] and support vector machines (SVMs) [2] are available. They mainly differ in the training algorithm. However, for the classification they all separate two classes by a hyperplane (Figure 2.1). This is achieved by projecting all data points $\mathbf{x}$ onto a weight vector $\mathbf{w}$ perpendicular to the hyperplane. Then a bias is added and the threshold function $\sigma(\cdot)$ is applied in order to obtain the label y:

$$y = \sigma(\mathbf{w}\mathbf{x} + \theta). \tag{2.3}$$

The elements of the weight vector $\mathbf{w}$ and the bias θ are the parameters $P = \{\mathbf{w}, \theta\}$ to be learned. To obtain integer class labels the step function $\sigma(\cdot)$ is applied. A drawback of the linear classifier is that it can not separate classes with non-linear class boundaries well, so for many data sets they do not provide suitable input-output functions. Also, they are not designed to handle more then

two classes. Nevertheless, it is possible to adapt linear classifiers to multi-class problems, but this is out of our scope. To allow for non-linear class boundaries so called kernel methods have been developed. They transfer the linear separation of the classes into a non-linearly transformed space and increase the complexity of the classifier.

For non-linear classification boundaries and in multi-class settings the k-nearest neighbors (k-NN) algorithm is more intuitive. It is a particularly simple algorithm that directly stores the known data point label tuples as parameters. The computations are moved to the classification step. For classifying a data point the k nearest training data points are computed. The label occurring most frequently in this neighborhood is assigned to that data point (Figure 2.1).

Since in the training phase tuples are only stored without further processing, it is straight forward to adapt the classifier when new training tuples become available. These tuples are then simply stored as well. Additionally, inspection and analysis of the classification results is possible by looking at the neighborhood. These properties made k-NN a popular choice in many domains. The main problem of this method is that it needs an appropriate distance measure. Often, the Euclidean distance is used, which is not always optimal, since the Euclidean distance requires the data dimensions to be scaled according to their relevance for classification.

2.2. Representations

Representations can help overcome some of the problems that arise in classification. Here, representations are introduced in the narrow scope of this thesis, for a broader discussion the reviews by van der Maaten et al. and Bengio et al. [22, 23] are recommended. All data we collect are representations of underlying information. However, different representations are suitable for different tasks. Therefore, the representation of the data needs to be adapted. For classification a good data representation will remove irrelevant information and reduce the dimensionality, while preserving discriminative information in order to improve generalization. Whereas for visualization, the main focus is on reducing the dimensionality to two or three. Retaining the information is only a secondary objective. Compressing data is another wide spread application of representations with the main goal of reducing the redundancies in the data.

Formally, a representation maps a data point $\mathbf{x}^{\text{original}} \in \mathbb{R}^{D_1}$ with D_1 dimen-

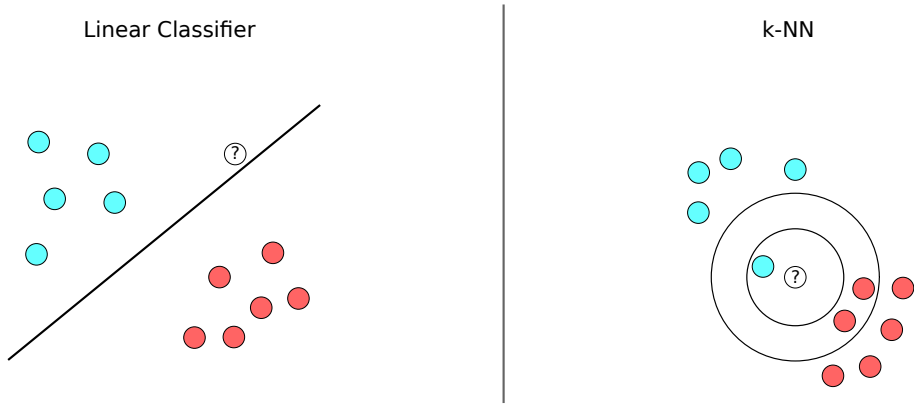


Figure 2.1.: The linear classifier on the left separates two classes (Data points are filled in cyan and red, respectively.) by a line. The label for a new data point (Point with a question mark.) is assigned depending on which side of the line the data point lies. So in this case cyan. In contrast, the k-NN algorithm selects the label based on the k nearest neighbors. If for example k was one, the cyan class label would be assigned, while for $k = 3$ it would be the red class label.

sions from data space to the representation space via a function $\mathcal{F}_P$

$$\mathbf{x}^{\text{new}} = \mathcal{F}_P(\mathbf{x}^{\text{original}}) \tag{2.4}$$

where $\mathbf{x}^{\text{new}} \in \mathbb{R}^{D_2}$ is the data point in the new representation with D_2 dimensions, P is a set of parameters for the mapping function. This concept has been introduced in multiple domains under different names [24]. In harmonic analysis and signal processing it is known as transformation. In learning theory it is called a feature map, while in information theory it is called encoding.

Here, the focus is in the application to classification settings. As mentioned in the previous section, a classifier may depend on the scaling of the data dimensions. This is of course a matter of representation. Another problem is to obtain good error rates for high dimensional data with non-linear class boundaries and a limited set of labeled data points available (Figure 2.2). A complex classifier is needed to separate the classes. However, such a classifier requires many data points to generalize well. If it is possible to reduce the dimensionality and simplify the class boundaries by a change of the representation, a classifier of low complexity may be sufficient and, thereby, also good generalization using only a few training samples may be achieved. Therefore, often the data representation is changed prior to training and classification.

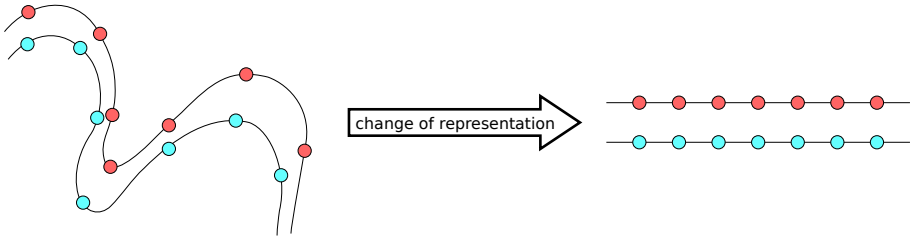


Figure 2.2.: Illustration of a representation change for the data points from two classes (Data points are filled in cyan and red, respectively.). In many datasets the representations for different classes are entangled as visualized on the left. By changing the representation we try to disentangle these classes as shown on the right.

Typically the mapping function $\mathcal{F}_P$ and the parameters P have been selected manually based on experience. Examples for such mappings in the computer vision domain are histograms, edges, and corners. In many applications a frequency representation obtained by the Fourier transformation or Wavelets has been vital. This manual approach works well for many tasks, but experts and time are required. Furthermore, in applications such as visual object recognition in unrestricted environments the results are not satisfactory.

To overcome the limitations of handcrafted mapping functions, learning methods can be applied. They allow an automatic adaption to data. Classical methods are principal component analysis (PCA) [7], independent component analysis [8], sparse coding [9, 25], and linear discriminant analysis [10]. These methods exploit statistical relationships in the data to improve the representation. For example the PCA assumes correlations in the data dimensions. The directions of maximum variance contain most of the information about the data. Therefore, after aligning the data dimensions with these directions, the low variance dimensions can be discarded. A core idea of this method is that the original representation is caused by underlying factors, which are entangled [26]. Of course, the PCA uses a very simple statistical model. If the underlying factors are entangled in a complicated manner, more sophisticated models are needed to completely untangle the factors. It would be optimal if that was achieved. Yet, significant improvements of error rates can be achieved without completely untangling the factors, leaving parts of this task to the classifier.

2.3. Learning Representations

Throughout large parts of this thesis new representation learning methods are developed. In this section some basic concepts for developing these methods are introduced. There are two main problems to be solved. A mapping function $\mathcal{F}_P$ is needed and the parameters P need to be found.

The mapping function depends on the goals pursued by the representation. A rescaling of the data dimensions can be achieved via a linear transformation using a diagonal matrix, where the matrix elements on the diagonal are the parameters to be learned. For disentangling factors non-linear transformations may be required. However, there is no rule how to obtain these transformations. It is subject to scientific research to come up with good mapping functions for various tasks.

After a mapping function $\mathcal{F}_P$ has been selected, we need to measure the quality of the outputs $\mathbf{x}^{\text{new}}$, in order to optimize P. This requires to establish a goal, and a measure to find out to which degree this goal has been achieved. For example, one might want to halve the dimensionality of the data and loose as little information as possible. Then a possible measure for the information loss is the mean squared reconstruction error

$$E(P, X) = \frac{1}{I} \sum_i ||\mathbf{x}_i^{\text{original}} - \mathcal{F}_P^{-1}(\mathcal{F}_P(\mathbf{x}_i^{\text{original}}))||^2 \tag{2.5}$$

averaged over a data set $X = [\mathbf{x}_1^{\text{original}}, \mathbf{x}_2^{\text{original}}, \ldots, \mathbf{x}_I^{\text{original}}]$. The function $E(P, X)$ is called energy function, since it is a positive scalar valued function which indicates good models by small values. Note, it is quite common to use the squared reconstruction error to obtain an energy function. Besides this dimensionality reduction objective from the example, many other objectives, such as independence, sparsity and slowness have been proposed.

Assuming an energy function is available to measure the quality of the outputs from the mapping function $\mathcal{F}_P$, the parameters P can be found by minimizing the energy. Optimal solutions can, however, not be guaranteed for arbitrary energy functions. Local optima make the optimization difficult. This should be considered when designing an energy function. Another pitfall are trivial solutions, which can be avoided by constraints. For the optimization then a subset of the available data, the training data is used. In case also the labels are needed, the optimization is referred to as supervised training, just like in the case of classification. Commonly, no labels are needed, which often allows a much larger

training set to be used by these unsupervised training methods, since the expensive labeling process is not required.

The two optimization methods used in this thesis are linear programming and gradient descent, which are presented briefly in the following. Linear programming can be used if the energy term and the constraints used are linear. It is a well established approach which has been researched for a long time. As a result there are several fast solvers that can guarantee to find the optimal solution. To use such a solver, the term needs to be reformulated into a standard form:

$$\min_{\mathbf{p}} \mathbf{c}^T \mathbf{p} \text{ s.t. } A\mathbf{p} \leq \mathbf{b},\ \mathbf{p} \geq \mathbf{0}, \tag{2.6}$$

where $\mathbf{p} \in \mathbb{R}^D$ is a vector of parameters to be optimized. The optimization term with N constraints is established via $\mathbf{c} \in \mathbb{R}^D$, $b \in \mathbb{R}^N$, and $A \in \mathbb{R}^{N \times D}$. For a more comprehensive introduction to linear programming the reader is referred to [27].

For many problems it is hard to find a good linear energy term. Often it is easier to model problems using non-linear energy terms. If a gradient $\frac{\partial E(P,X)}{\partial P}$ for the energy $E(P, X)$ is available, it can be used for optimization by gradient descent. The main idea is to follow the gradient starting from a randomly initialized P_0 iteratively to a minimum. The steps are given by the gradient and scaled with a learning rate. Thus, the parameters P_t will change according to:

$$P_{t+1} = P_t + \eta \left. \frac{\partial E(P, D)}{\partial P} \right|_{P_t}, \tag{2.7}$$

where P_t are the parameters at iteration step t. This is a very simple approach to optimization that finds good solutions for many energy terms. For large data sets it can be time consuming to compute the entire gradient. Often it is a lot faster to approximate the gradient using only a single data point or a subset of the data points, often referred to as mini-batch. In every iteration then a different point or mini-batch is selected. This approach is called stochastic gradient descent (SGD). Unfortunately, selecting good parameters is crucial for SGD. A good practical guide for the parameter selection was published by Bottou [28]. In the experiments of my thesis I avoided manually selecting parameters such as the learning rate using variational-SGD [29] or the sum-of-functions optimizer [30].

One problem often encountered in SGD optimization is the poor performance on highly correlated data with a large variation in variance. Therefore, often a preprocessing step is applied prior to representation learning. By whitening the

data, gradient steps tend to go to the optimum more directly. The zero component analysis (ZCA) [31, 32] is a particularly popular method for whitening in deep convolutional network domain. Similar to PCA, the data is projected onto the eigenvectors $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N] = V$, which together with the eigenvalues $\text{diag}([\lambda_1, \lambda_2, \ldots, \lambda_N]) = \Lambda$ are obtained from the covariance matrix $C = V\Lambda V^\top$ of the data set. By $\text{diag}(\lambda)$ a diagonal matrix Λ with the elements of λ placed on the diagonal is created. The projected data is rescaled and projected back to the original domain. These three steps can be expressed using the matrix

$$W_{ZCA} = V\Lambda^{-1/2}V^\top, \tag{2.8}$$

where $\Lambda^{-1/2}$ is a diagonal matrix with $[1/\sqrt{\lambda_1}, 1/\sqrt{\lambda_2}, \ldots, 1/\sqrt{\lambda_N}]$ on the diagonal. The matrix W_{ZCA} is then used for a linear transformation

$$\mathbf{x}^{\text{whitened}} = W_{ZCA}^\top \mathbf{x}^{\text{original}} \tag{2.9}$$

to obtain the whitened data point $\mathbf{x}^{\text{whitened}}$ from $\mathbf{x}^{\text{original}}$. In Figure 2.3 the transformation vectors $[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N] = W_{ZCA}$ obtained from a set of natural images are visualized. Clearly, they are local and very similar to each other. Due to this similarity it is possible to use one of the central elements as filter for whitening. Note, this preprocessing step is a form of representation change which is learned.

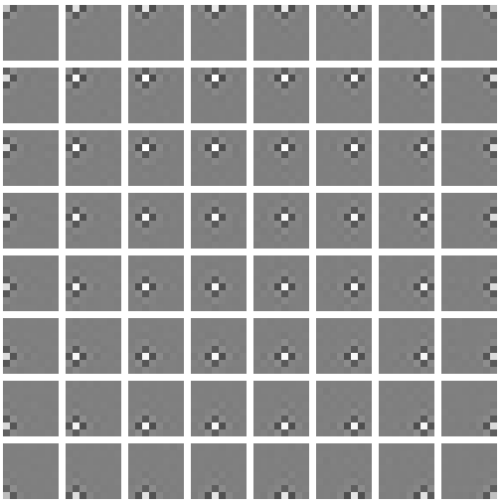


Figure 2.3.: ZCA components obtained from a set of 8×8 pixel natural images. The component vectors were rearranged to form 8 by 8 pixel images.

Part I.

Linear Representation Learning

3. Introduction to Linear Representation Learning

Real world data can be challenging to machine learning methods. But, by change of the representation with a suitable linear transformation, the task can be simplified. Consider the k-nearest neighbor classifier (k-NN) [1]. It can easily adapt to various non-linear decision boundaries without a prior on underlying data distributions. No training is required and, therefore, additional labeled data can directly be incorporated to improve the error rates. These properties make it popular in various applications. However, in the standard form it uses a Euclidean metric, which depends on the scaling of the data dimensions. Hence, a unsuitable scaling will cause bad a performance. This problem can be handled by a linear transformation significantly improving the error rates [33]. The same is true for other machine learning methods such as Learning Vector Quantization (LVQ) [34] and k-Means [35], which depend on a fixed measure for estimating distances. Note, while often the data has a non-linear class boundary, it is not necessary to disentangle and linearize this boundary by a non-linear transformation. The machine learning methods mentioned above are well suited to cope with non-linear class boundaries.

In many applications the data is collected from different domains. For example medical data may contain substance concentrations, blood pressure, and heart rate. The relative scaling of such data is arbitrary, since there is no generally adequate scaling. However, the performance of many machine learning methods, which measure distances between data points, depend on the scaling of the input data and an arbitrary scaling may be disadvantageous. To adjust this, each data dimension $x_\mu, \mu = 1, \ldots, D$ can be rescaled by a weighting factor w_μ. This process called feature weighting will change the distance

$$d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_{\mathbf{w}} = \sqrt{\sum_{\mu=1}^{D} w_\mu^2 (x_\mu - x'_\mu)^2} = \sqrt{\sum_{\mu=1}^{D} (w_\mu x_\mu - w_\mu x'_\mu)^2} \tag{3.1}$$

between pairs of points $\mathbf{x}$ and $\mathbf{x}'$ in the new representation. This distance metric

is called weighted euclidean distance. Besides improving the distances between data, also irrelevant dimensions can be removed by zero weights. This is beneficial for machine learning methods due to an improved signal to noise ratio and a decrease in complexity of the learning methods.

In addition to an arbitrary scaling there are also often correlations and thus redundancies in the data. By a full linear transformation $L\mathbf{x}$ of the data point $\mathbf{x}$ by a transformation matrix L, the dimensionality of the output can be further reduced compared to feature weighting. Therefore, also the signal to noise ratio will increase, while the complexity of follow up methods can be decreased. Since many of the learning methods that benefit from a linearly transformed representation are based on distances between data points, the metric for the distances is changed instead of the representation. This adaption is referred to as metric learning. Often a Mahalanobis distance

$$d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_W = \sqrt{(\mathbf{x} - \mathbf{x}')^T W (\mathbf{x} - \mathbf{x}')} \tag{3.2}$$

with the positive semidefinite matrix W is used. This distance, however, is equivalent to a linear transformation of the data:

$$\begin{aligned} ||\mathbf{x} - \mathbf{x}'||_W &= \sqrt{(\mathbf{x} - \mathbf{x}')^T W (\mathbf{x} - \mathbf{x}')} && (3.3)\\ &= \sqrt{(\mathbf{x} - \mathbf{x}')^T L^T L (\mathbf{x} - \mathbf{x}')} && (3.4)\\ &= \sqrt{(L\mathbf{x} - L\mathbf{x}')^T (L\mathbf{x} - L\mathbf{x}')} = ||L\mathbf{x} - L\mathbf{x}'||_2. && (3.5) \end{aligned}$$

Therefore, metric learning using a Mahalanobis distance can be seen as linear representation learning. Furthermore, it is closely related to feature weighting, since feature weighting can be realized by restricting the transformation matrix L to a diagonal matrix.

For all these methods parameters need to be estimated. The main approach to this are so called filter methods. These use a heuristic to improve the representation. For example for classification it is beneficial if equally labeled data (intraclass) is clustered closely, while differently labeled data (interclass) is far apart.

This chapter reviews important methods for feature weighting and metric learning. In the following two chapters one method for feature weighting and one method for metric learning are introduced. Both with a focus on improving the error rates of k-NN, a classifier still widely applied due to its simplicity and ability to handle non-linear class borders.

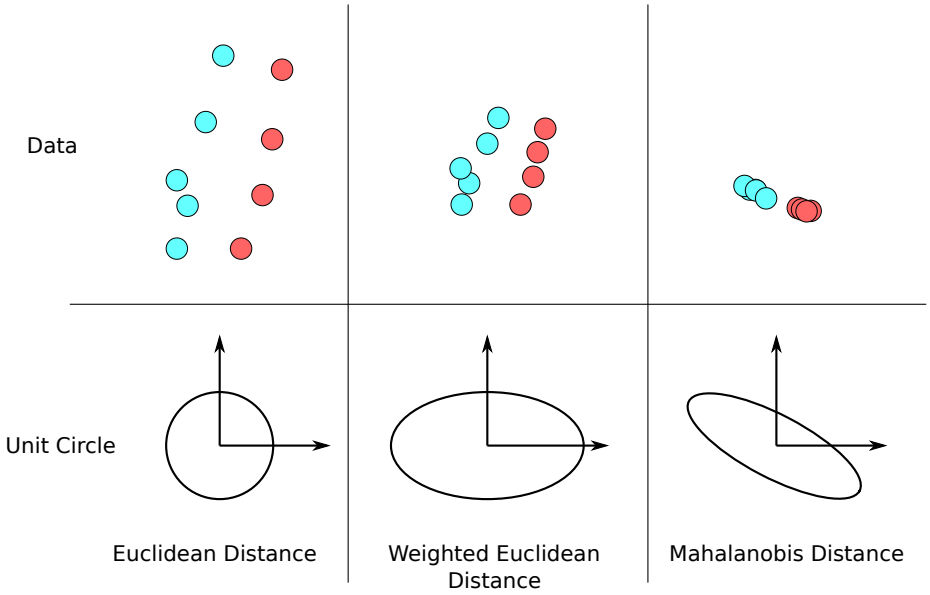


Figure 3.1.: Visualization of the effect of different metrics. The top row shows data in its original representation on the left and two transformed versions next to it. In the middle column only the axes were rescaled. On the right a full linear transformation was applied. These changes of the distances can equivalently be achieved by the weighted Euclidean distance for the middle column and the Mahalanobis distance for the right column. Note, how the ratio of intraclass to interclass distances is improved. The bottom column illustrates the adaption of metrics using unit circles.

3.1. Feature Weighting

Despite useful applications for feature weighting, there are only few methods available. The most simple approach to feature weighting is to rescale every dimension through normalization of the data variance along every dimension. However, this does not take class label information into account, and, therefore, may even decrease the classification performance. The Relief algorithm by Kira and Rendell [36] aims to account for this problem. Two extensions to the concepts from Relief are Simba [37] and I-Relief [38]. Originally, all these methods were developed to select the most important dimensions for classification by learning a weight vector that indicates the relevance of each dimension. Yet, this weight vector has proven to work well for feature weighting. To obtain ad-

ditional methods one could restrict the metric learning methods from the next section to rescaling of the dimensions. Below, the well established methods Relief and Simba are described.

3.1.1. Relief

Relief rescales the dimensions of input data assuming that it is beneficial if equally labeled data is close together and differently labeled data is far apart. The rescaling is done based on local measurements. For every point $\mathbf{x}$ the ratio between the closest equally labeled datapoint $nh(\mathbf{x})$, called nearest hit, and the closest differently labeled data point $nm(\mathbf{x})$, called nearest miss, is changed. This is done iteratively. After the scaling factors $w_\mu, \mu = 1, \ldots, D$ for every dimension μ are initialized with zero, a random datapoint $\mathbf{x}$ is selected. The nearest hit and the nearest miss are determined in the original space. Then the scaling factors are updated by

$$w_{\mu,t+1} = w_{\mu,t} + (x_\mu - nm(x)_\mu)^2 - (x_\mu - nh(x)_\mu)^2, \tag{3.6}$$

for every iteration step t. Note, the update rule is a heuristic that is not derived from a measure that describes the quality of the representation obtained by the weight vectors. Nevertheless, convergence seems to be no problem. Problems could be caused by the fixed nearest hits and misses, making Relief dependent on the initial scaling of the data.

3.1.2. Simba

The main concept of Simba is very similar to Relief. Again the ratio between nearest hit $nh(\mathbf{x})$ and the nearest miss $nm(\mathbf{x})$ are iteratively improved. Simba, however, takes a more principled approach by introducing a cost function. This allows for a gradient descent and, therefore, the representation is guaranteed to improve. The cost function

$$E = \sum_i \theta_i \tag{3.7}$$

uses a difference of weighted distances

$$\theta_i = \frac{1}{2}(||\mathbf{x}_i - nm(\mathbf{x})_i||_w - ||\mathbf{x}_i - nh(\mathbf{x})_i||_w) \tag{3.8}$$

with

$$||z||_w = \sqrt{\sum_\mu w_\mu^2 z_\mu^2}. \tag{3.9}$$

To find the D optimal weights $w_\mu,\ \mu = 1, 2, \ldots, D$ an iterative approach is chosen. First, the weights are initialized with $w_\mu = 1$ and then, for a random data point $\mathbf{x}$ the nearest hit and the nearest miss are determined in the current rescaled space. Then the gradient of the energy function is used to update the weights w_μ

$$w_{\mu,t+1} = w_{\mu,t} + \frac{1}{2}\left(\frac{(x_\mu - nm(x)_\mu)^2}{||x - nm(x))||_w} - \frac{(x_\mu - nh(x)_\mu)^2}{||x - nh(x))||_w}\right) w_\mu. \tag{3.10}$$

After the update the scaling factors are normalized by $w_\mu / \max_j w_j^2$. These steps starting from the selection of a random point $\mathbf{x}$ are repeated. In contrast to Relief, the nearest hit and the nearest miss are redetermined in the rescaled space in every iteration. However, this may cause multiple local optima.

3.2. Metric Learning

Linear representation learning by PCA is often applied for dimensionality reduction. As mentioned above this can easily be turned into a metric. The main drawback, however, is that it does not take label information into account and, therefore, the resulting representation may loose discriminativity. Here, a short review on popular linear metric learning methods, which optimize discriminativity, is given. These are Large Margin Nearest Neighbors (LMNN) [39], Neighborhood Component Analysis (NCA) [33], and Mahalanobis Metric Learning for Clustering (MMC) by Xing et al. [35]. Besides these methods, several non-linear and local metric learning methods have been proposed, which are, however, not within the scope of this work. More detailed reviews can be found in [40, 41].

3.2.1. Large Margin Nearest Neighbors

LMNN [39] is closely linked to the mechanics of the k-NN algorithm. A neighborhood of the k closest equally labeled data points $\mathbf{x}_j,\ j \in \mathcal{N}_i$ is established for every data point $\mathbf{x}_i$. The neighborhood points are identified by the set of indices $\mathcal{N}_i$. By adapting the metric LMNN tries to free this neighborhood from

differently labeled data. Additionally, it tries to minimize the distance between equally labeled data points. This goal can be formulated as a semidefinite program

$$\arg\min_{W}(1-\alpha)\sum_{i,j\in\mathcal{N}_i}(\mathbf{x}_i-\mathbf{x}_j)^\top W(\mathbf{x}_i-\mathbf{x}_j)+\alpha\sum_{i,j\in\mathcal{N}_i,l}(1-h_{il})\xi_{ijl}\ \text{s.t.} \tag{3.11}$$

$$\textbf{(1)}\ (\mathbf{x}_i-\mathbf{x}_l)^\top W(\mathbf{x}_i-\mathbf{x}_l)-(\mathbf{x}_i-\mathbf{x}_j)^\top W(\mathbf{x}_i-\mathbf{x}_j)\geq 1-\xi_{ijl} \tag{3.12}$$

$$\textbf{(2)}\ \xi_{ijl}\geq 0 \tag{3.13}$$

$$\textbf{(3)}\ W\succeq 0, \tag{3.14}$$

and, therefore, an optimal solution can be found. Nevertheless, the authors have adapted an optimizer specifically to this problem to obtain fast convergence. To identify the data points indices $i = 1,...N$, $j = 1,..,k$, and $l = 1,...N$ are used, where N is the number of data points. The parameter α weights the terms for minimizing within class distance and optimizing k-NN performance. In 3.11 a binary indicator function h_{il} is used. It equals one if $\mathbf{x}_i$ has the same label as $\mathbf{x}_l$ and zero else. Slack variables ξ_{ijl} are needed for the problem to be always solvable, since it can not be guaranteed that all neighborhoods can be freed from the differently labeled data. The main drawback of LMNN is that it depends on a good selection of the nearest neighbors by the Euclidean distance.

3.2.2. Neighborhood Component Analysis

Similar to LMNN, the NCA [33] tries to free an area around every data point from differently labeled data. While LMNN uses discrete k-neighborhoods, the NCA approach is stochastic and soft. It optimizes the leave one out classification of k-NN. To avoid a discontinuous function as optimization criterion, k-NN is replaced by a stochastic neighbor selection rule. For each point $\mathbf{x}_i$ the neighbor $\mathbf{x}_j$ has an equal class label with probability

$$p_{ij}=\frac{\exp(-||L\mathbf{x}_i-L\mathbf{x}_j||^2)}{\sum_{i\neq k}\exp(-||L\mathbf{x}_i-L\mathbf{x}_k||^2)}, p_{ii}=0, \tag{3.15}$$

where L is a transformation matrix. By adding the probabilities p_{ij} for all points $\mathbf{x}_j$ from the same class C_i as the point $\mathbf{x}_i$, the probability p_i for correctly classifying $\mathbf{x}_i$ is computed. An objective function

$$E(L)=\sum_i\sum_{j\in C_i}p_{ij}=\sum_i p_i \tag{3.16}$$

is then established by the sum of probabilities for all data points $\mathbf{x}_i$. The derivative of this function is then used to adapt L by gradient descent. The main drawback is that $E(L)$ may have multiple local optima. Therefore, it will not always find the optimal solution.

3.2.3. Mahalanobis Metric Learning for Clustering

An early metric learning method is MMC by Xing et al. [35]. Originally, it was introduced in the context of clustering. But it can also be applied to domains such as k-NN, since MMC is not specifically adapted to any machine learning algorithm. It considers all pairs of data points in a global optimization. The main idea is to minimize the sum of distances for all equally labeled data points $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$, while keeping the sum of the distances for differently labeled data $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$ above a bound of one. This is reached via optimizing

$$\arg\min_{W} \sum_{x_i, x_j \in \mathcal{S}} (\mathbf{x}_i - \mathbf{x}_j)^\top W (\mathbf{x}_i - \mathbf{x}_j) \text{ s.t.} \tag{3.17}$$

$$\textbf{(1)} \sum_{x_i, x_j \in \mathcal{D}} \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top W (\mathbf{x}_i - \mathbf{x}_j)} \geq 1 \tag{3.18}$$

$$\textbf{(2)}\ W \succeq 0, \tag{3.19}$$

where the second constraint enforces a positive semidefinite matrix W. To find W, a gradient descent algorithm is combined with two projections to enforce the constraints. This optimization is guaranteed to converge to the optimal solution.

4. A new Feature Weighting Approach

In the previous chapter feature weighting methods were presented. These methods all have in common that they use local neighbors for their optimization. If the main goal is to optimize k-NN classification performance this is very intuitive, since k-NN is based on local neighborhoods. The neighbors are selected either prior to the optimization process, which makes the resulting scaling dependent on the initial scaling, or the neighbors are updated iteratively. The disadvantage of this iterative approach, as it is taken for example by Simba, is that the optimization problem becomes non-convex, which might cause a decrease in classification performance.

Here, I introduce a new feature weighting method for improved k-NN classification, which avoids these problems of local neighborhood dependent optimization. It becomes independent of the initial scaling by global optimization, and optimal solutions are guaranteed since the global criterion can be implemented by linear programming. After introducing the basic method, it is extended by soft constraints to handle noise and outliers in the training data more robustly. Both methods are then evaluated on artificial and natural datasets. These methods and some of the experiments have been published in [11–13].

4.1. Maximum Distance Minimization

Our goal is to minimize the classification error of the k-NN algorithm by rescaling the dimensions of the data. The dimensions are rescaled by a weighting vector $\mathbf{w}$, which we need to learn. In our approach we assume that equally labeled data points should be close together and differently labeled data points should be far apart for a good k-NN classification error. To achieve this we minimize the maximum distance between all pairs of data points that belong to the same class. Additionally, the minimum distance of data points from different classes is constrained to one. This constraint in combination with the minimization criterion will improve the ratio of distances, while the constraint prevents

the trivial solution $\mathbf{w} = \mathbf{0}$. To avoid ambiguous solutions only positive weights $\mathbf{w} \geq \mathbf{0}$ are allowed. Due to these main ideas, which are illustrated in Figure 4.1, we call our method Maximum Distance Minimization (MDM). Note, it is possible to do the opposite and impose a maximum intraclass distance while maximizing the minimum interclass distance (Minimum Distance Maximization (MDM)). Both approaches are mathematically equivalent.

Given data points $\mathbf{x}_i \in \mathbb{R}^D$ with class labels y_i, $i = 1, ..., N$, we formally solve the following constrained optimization problem:

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\mathbf{w}} \geq 1 \qquad \forall i, j : y_i \neq y_j \tag{4.1}$$

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\mathbf{w}} \leq r \qquad \forall i, j : y_i = y_j \tag{4.2}$$

$$\min_{\mathbf{w}} r \qquad w_\mu \geq 0 \ \forall \mu. \tag{4.3}$$

Here, r is the maximum intraclass distance. This optimizations problem can be formulated as a linear program, with the number of constraints in this formulation growing quadratically with the number of data points. Note, the constraints can always be fulfilled and, therefore, our optimization problem is always solvable despite having hard constraints. For our implementation we used the MOSEK-solver[1], one of the many fast solvers available.

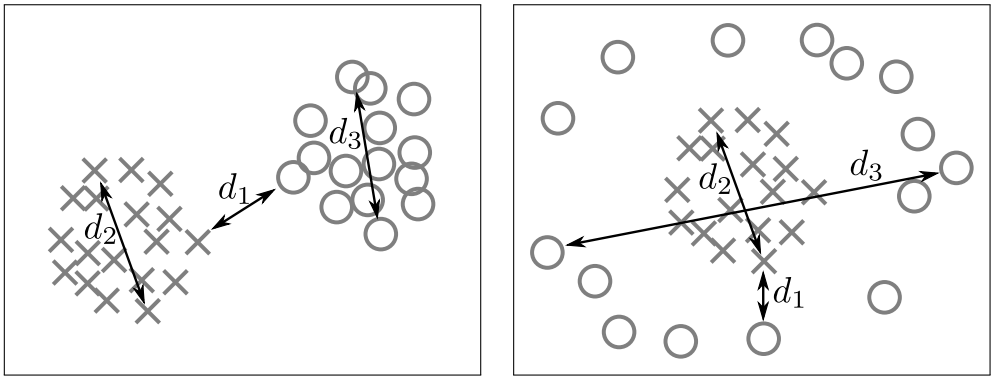


Figure 4.1.: This figure shows two different settings. d_1 denotes the shortest interclass distance. This distance is fixed to one by equation (4.1). The largest intraclass distance for class one (crosses) is d_2 and for class two (circles) d_3. The larger distance of the two (d_2) determines r in equation (4.2) and is minimized.

[1] http://www.mosek.com/

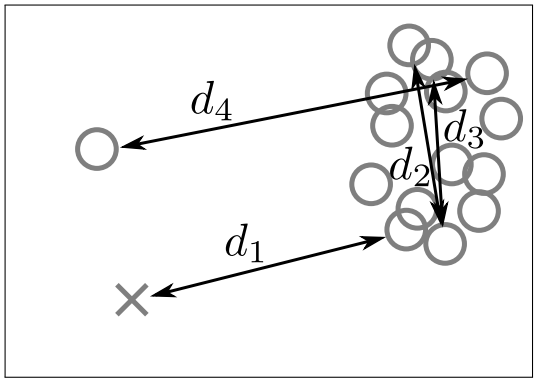


Figure 4.2.: Illustration of the softness effect. d_1 is the largest interclass distance. There is no change in the influence of d_1 compared to standard MDM. However, there is a change for the intraclass distances d_2 to d_4. In a hard setting r will equal d_4 and, therefore, the outlier will have a major effect on the final weight vector $\mathbf{w}$. The smaller C, the more intraclass tuples are allowed to have a distance larger than r. All these tuples will influence $\mathbf{w}$ simultaneously and in that way lower the influence of the outlier. Also the noise effect will be lower, if the final $\mathbf{w}$ depends on several similar tuples like d_2 and d_3.

4.2. Soft Maximum Distance Minimization

Maximum Distance Minimization as introduced in the previous section takes only the most distant data points into account. This hard constraint may make MDM sensible to outliers and noisy data. Here, we extend MDM by introducing slack variables ξ_i for every data point $\mathbf{x}_i$ in order to implement soft constraints. These soft constraints allow a few intraclass distances larger than r and, hence, it reduces the influence of outliers and noise. In Figure 4.2 this soft approach is illustrated. The new optimization problem with the slack variables ξ_i becomes

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\mathbf{w}} \geq 1 \qquad \forall i, j : y_i \neq y_j \tag{4.4}$$

$$||\mathbf{x}_i - \mathbf{x}_j||^2_{\mathbf{w}} \leq r + \xi_i \qquad \forall i, j : y_i = y_j \tag{4.5}$$

$$\min_{\mathbf{w}} r + C \sum_i \xi_i \qquad w_\mu \geq 0 \; \forall \mu, \xi_i \geq 0 \; \forall i. \tag{4.6}$$

By C the use of the slack variables is regulated. Small values of C will allow for a large deviation from the hard constraints. Like the hard MDM version from the previous section, Soft Maximization Distance Minimization can be implemented using linear programming. However, the number of parameters to optimize

Name	Samples	Dimensions	Classes
Iris	150	4	3
Wine	178	13	3
Breast Cancer	683	10	2
Pima Diabetes	768	8	2
Parkinsons	195	22	2
Seeds	210	7	3
Breast Cancer	98	1213	3
DLBCL	180	661	3
Leukemia	248	985	6
Lung Cancer	197	1000	4

Table 4.1.: Description of the datasets. The top datasets are from the UCI repository, and the bottom ones are gene expression data.

grows linearly with the number of data points, and MDM is not parameter free anymore, since C has to be chosen appropriately.

4.3. Experiments and Comparisons

To explore the basic properties of MDM we use artificial data sets. In the visualization in Figure 4.3 the results are shown. The uncorrelated and well separated data distributions allow for perfect discrimination using only one of the two dimensions. MDM assigns zero weight to the non-discriminative dimension. If the distributions overlap, both dimensions are needed for good classification, which is reflected in the MDM results. The correlated data requires metric learning for proper handling. Hence, the small adaptions by MDM are reasonable. Note, for both data sets with overlapping distributions, the dimensions are scaled up. This is due to the interclass term promoting a distance of at least one for the differently labeled points. Finally, the structured non-gaussian data is handled nicely by removing the non-discriminative dimension.

Then MDM was evaluated on real world data using datasets from the UCI repository [42] and gene expression datasets available from the Broad Institute website[2]. Both are described in Table 4.1.

We compared MDM with results obtained with the standard Euclidean dis-

[2]http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi (Full dataset names: Breast-A, DLBCL-B, St. Jude Leukemia, Lung Cancer)

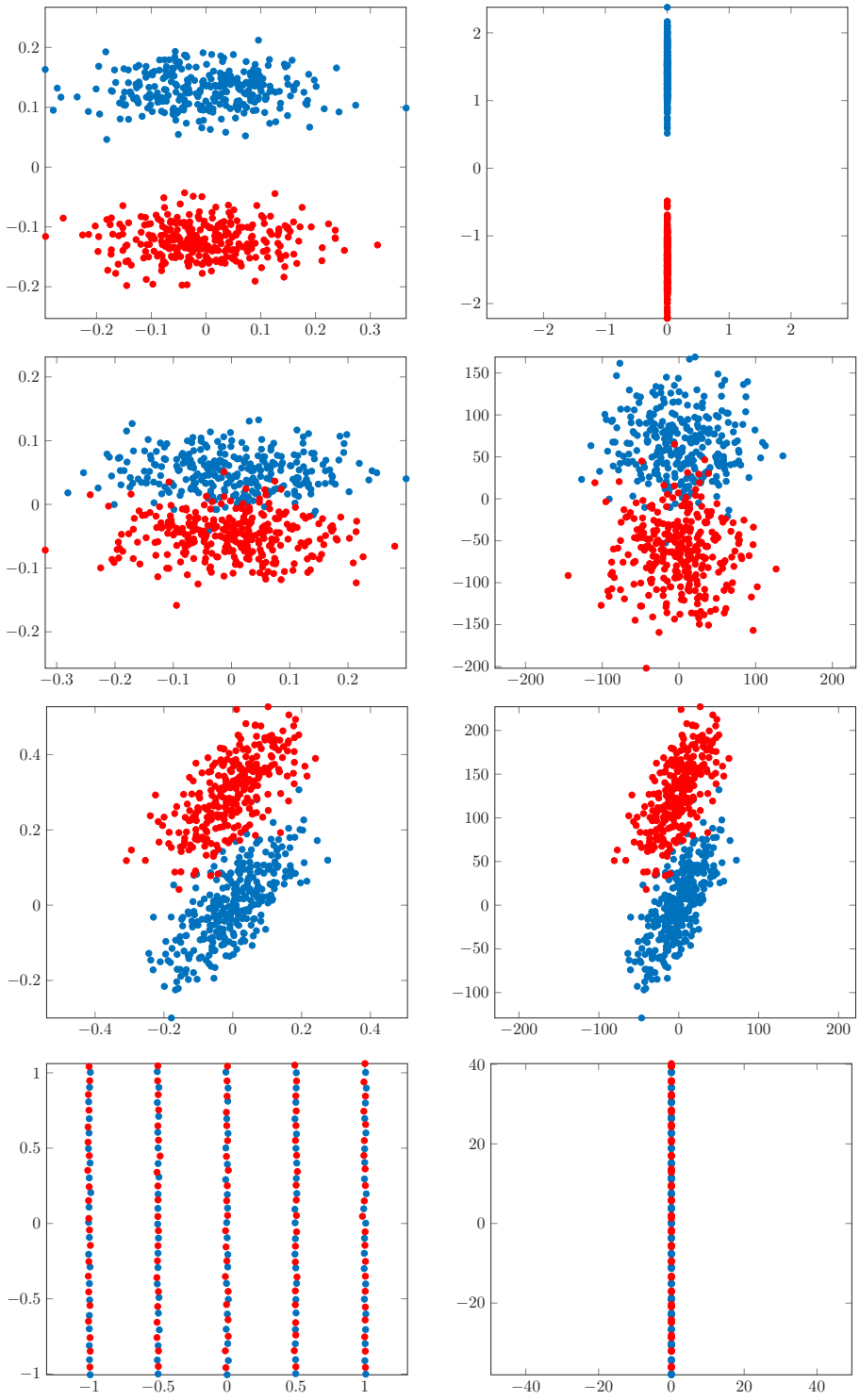


Figure 4.3.: Artificial 2D datasets are depicted in the original scaling on the left and after application of MDM on the right.

tance as well as obtained with the feature weighting algorithms Relief and Simba[3]. As a reference and out of competition we also show the results obtained with LMNN as a complete metric learning method. For the soft MDM, the softness parameter C was selected for each split of the data individually. From the set $\{2^{-x}|x \in \{0,\ldots,10\}\}$, the best C was chosen by 4-fold cross-validation on the training data. For Relief and Simba we need to set the number of training epochs. Here we used one epoch, as this is default in the implementation that we used. Longer training sometimes deteriorated the results. Due to the non-convex optimization, five random starting points were chosen for Simba for every training. For LMNN its parameter α was chosen to be 0.5. The authors described this to be a good choice [43]. To evaluate the classification performance, we split the data into five almost equally large parts and used four of these parts for training and one for testing. The partitioning was used five times for training and testing, with each part being left out once. This was done for ten different splits of the data, so that 50 different test and training sets were obtained. After the weighting was learned on the training set, k-NN with $k = 3$ was used to obtain the error rates on the independent test set.

First, we compared the classification performance on the UCI datasets. Table 4.2 shows the results on the raw data. MDM is clearly superior. The error rates are improved significantly compared to standard k-NN based on the original scaling ("Euclidean"). Only for the iris data, the original scaling is a good choice. Relief and Simba sometimes even worsen the classification performance compared to the original scaling.

In Table 4.3 we see the results after a prior rescaling such that the data distribution is normalized to zero mean and variance one along each dimension. With prior rescaling, Relief and Simba become competitive due to a different initial selection of the neighbors. Obviously, Relief and Simba seem to be very dependent on a good initial scaling. It seems that the initial neighbors more or less remain neighbors during the optimization procedure. But then, obviously, already the initial scaling is a good choice and achieves good results. Even though the other methods in general improve their results on the preprocessed data, MDM remains very competitive. Interestingly, not for all datasets the preprocessing by normalization yields improved results. Especially for the iris data the initial scaling seems to be a better choice. This demonstrates that it is not always clear

[3]We used a implementation by A. Navot and R. Gilad-Bachrach, which is available at http://www.cs.huji.ac.il/labs/learning/code/feature_selection.bak/

	Euclidean	MDM	MDM Soft	Relief	Simba	LMNN
Iris	**3.87(3.32)**	4.33(3.10)	4.00(2.94)	4.00(2.86)	6.27(3.91)	4.00(2.86)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	3.96(0.20)	4.00(0.00)
Wine	30.28(7.25)	2.64(2.81)	**2.13(2.56)**	32.80(6.65)	32.52(7.19)	5.57(3.82)
	13.00(0.00)	12.10(0.65)	12.08(0.67)	13.00(0.00)	13.00(0.00)	12.12(0.63)
Breast	39.20(4.35)	3.50(1.43)	**2.86(1.27)**	39.36(4.30)	39.36(4.30)	4.06(1.34)
Cancer	10.00(0.00)	9.94(0.31)	9.70(0.51)	10.00(0.00)	9.64(0.85)	8.02(0.25)
Pima	29.99(3.45)	27.34(2.52)	**26.43(2.90)**	29.41(3.18)	29.92(3.53)	28.42(3.15)
Diabetes	8.00(0.00)	7.90(0.30)	7.76(0.43)	8.00(0.00)	7.42(0.70)	8.00(0.00)
Parkinsons	14.72(4.96)	10.31(4.85)	**7.59(4.21)**	15.49(4.86)	15.64(4.75)	13.49(4.91)
	22.00(0.00)	21.20(0.88)	20.44(0.73)	22.00(0.00)	22.00(0.00)	21.82(0.39)
Seeds	11.90(3.63)	7.52(3.74)	**7.00(2.98)**	11.71(3.56)	11.86(3.65)	4.86(2.84)
	7.00(0.00)	7.00(0.00)	7.00(0.00)	7.00(0.00)	7.00(0.00)	7.00(0.00)

Table 4.2.: Results on UCI datasets. The top entry is the average test error in percent followed by the STD in parentheses. Below the error rates, the average rank, again followed by the STD, is given. In case of the feature weighting methods, the rank is equal to the number of non-zero weights. The best results obtained with feature weighting are indicated by bold face.

	Euclidean	MDM	MDM Soft	Relief	Simba	LMNN
Iris	5.40(3.92)	4.33(3.10)	**4.00(2.94)**	4.87(3.10)	4.73(2.94)	4.47(3.27)
	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)	4.00(0.00)
Wine	3.88(2.84)	2.64(2.81)	**2.13(2.56)**	3.37(3.21)	3.43(3.03)	2.42(2.12)
	13.00(0.00)	12.70(0.51)	12.96(0.20)	13.00(0.00)	12.54(0.91)	13.00(0.00)
Breast	3.60(1.43)	3.38(1.48)	**2.75(1.36)**	3.35(1.38)	4.04(1.42)	3.38(1.50)
Cancer	10.00(0.00)	10.00(0.00)	9.88(0.33)	10.00(0.00)	9.76(0.62)	9.48(0.79)
Pima	26.73(2.64)	27.34(2.52)	**26.53(2.89)**	26.90(3.54)	27.20(3.45)	26.46(2.67)
Diabetes	8.00(0.00)	8.00(0.00)	8.00(0.00)	8.00(0.00)	5.66(0.85)	8.00(0.00)
Parkinsons	9.13(3.85)	10.31(4.85)	9.59(5.10)	**5.69(3.25)**	7.13(3.92)	5.74(2.91)
	22.00(0.00)	21.50(1.11)	21.16(1.89)	22.00(0.00)	21.92(0.27)	21.96(0.20)
Seeds	8.05(3.00)	7.52(3.74)	**7.00(2.98)**	10.24(3.51)	9.57(3.77)	6.67(3.50)
	7.00(0.00)	7.00(0.00)	7.00(0.00)	7.00(0.00)	6.98(0.14)	7.00(0.00)

Table 4.3.: Results on the UCI datasets after prior rescaling. The dimensions were normalized such that the data distributions have a mean equal zero and a variance equal one. The notation and structure of this table is the same as in Table 4.2.

	Euclidean	MDM	Relief	Simba	LMNN
Breast	**8.07(6.13)**	11.42(7.25)	13.16(7.89)	14.47(7.43)	9.78(7.13)
Cancer	1213.00(0.00)	364.76(62.65)	1213.00(0.00)	1213.00(0.00)	1137.42(2.97)
DLBCL	13.11(5.24)	14.67(5.33)	**12.00(5.62)**	13.28(6.55)	15.44(4.32)
	661.00(0.00)	293.86(34.13)	661.00(0.00)	661.00(0.00)	559.56(1.97)
Leukemia	2.21(2.27)	**1.74(1.96)**	2.18(2.45)	4.12(2.87)	0.69(1.33)
	985.00(0.00)	473.24(55.28)	985.00(0.00)	984.96(0.20)	821.48(7.53)
Lung	**4.37(2.77)**	5.49(3.18)	4.88(2.78)	8.29(4.12)	4.78(2.66)
Cancer	1000.00(0.00)	536.62(78.55)	1000.00(0.00)	999.78(0.46)	870.86(1.87)

Table 4.4.: Results on gene expression data (after prior rescaling). The notation and structure of this table is the same as in Table 4.2.

whether a prior rescaling and which rescaling is beneficial. The main advantage of MDM accounts for this problem, since it is independent of such a prior rescaling. Another interesting result is that although LMNN is much more flexible and complex, it does not perform better, at least on these data sets.

In Table 4.4 we see the results on the gene expression data. They were obtained with the same prior rescaling as used for the UCI data in Table 4.3. The gene expression data are a lot more challenging because the number of data dimensions compared to the number of data points is very large, as shown in Table 4.1. This curse of dimensionality is very challenging for feature weighting and metric learning methods. All methods perform on a similar level as standard Euclidean distance, taking the large standard deviation into account. However, we see a nice feature of our MDM method: MDM remarkably reduces the dimensionality of the data without the use of any parameters. The dimensionality reduction is directly induced by the formulation of the optimization problem. Methods like Relief and Simba, which where specifically designed for this task, need a threshold to be set either by some heuristic or by hand. The dimensionality can be reduced even further if the soft MDM is used, but this comes at the expense of the softness parameter which needs to be set.

4.4. Discussion

We developed a simple feature weighting method that can be implemented using linear programming. The problem of arbitrarily scaled data dimensions, which often cause bad k-NN error rates, is handled by the global optimization of our MDM method. MDM finds weights that yield very competitive k-NN

classification results on standard benchmark problems. For most of the UCI data sets MDM achieves error rates comparable to those of the much more complex metric learning method LMNN. So the additional flexibility of metric learning is not beneficial. A possible reason for this is that the flexibility is not needed for a better representation, or leaning the additional parameters requires more training data for reliable generalization. Hence, for some applications it is better to use feature weighting and keep the interpretability.

What distinguishes our method from commonly used methods is that it uses a global convex optimization criterion, which compares all tuples. This approach avoids predefining which data points are adjacent. Due to the convex optimization problem MDM finds a global optimum. Therefore, the results are almost independent from the initial scaling of the data dimensions. However, since the global optimum is not always a single vector, there may be still some variation in the results. On most of the benchmark data MDM performs well. Of course, for some data sets, such as the Parkinsons data from the UCI data sets, it is beneficial to take local data structures into account. There, both local methods perform better in our comparison (Table 4.3).

In addition to the classification performance, we demonstrated the capability of our method to reduce the dimensionality of a given classification problem on high dimensional data. Contrary to other methods, parameters as for example a threshold are not needed to cut off dimensions with small weights. This makes MDM a straightforward method to use for dimensionality reduction.

A disadvantage of MDM is that the number of constraints grows quadratically with the number of data tuples, which causes a demand of computational recourses for large data sets. Since usually only a small fraction of the tuples is responsible for the final weighting, it should be possible to find an equal solution with much less constraints, i.e. using only a fraction of the tuples.

Since MDM is not specifically adapted to the locally operating k-NN, we expect other distance based machine learning methods to also benefit from our global and robust approach.

5. A new Global Metric Learning Approach

In Chapter 3 metric learning was introduced, which improves the performance of distance based machine learning methods by changing the distances between data points. Linear metric learning either adapts a Mahalanobis distance or a linear transformation to achieve this improvement. Well known methods are NCA [33] and LMNN [39, 43]. They try to free local neighborhoods from differently labeled data to improve the k-NN performance. NCA uses soft transitions and optimizes a projection matrix, while LMNN has hard transitions and operates on the Mahalanobis distance. Another popular method is MMC [35], which globally minimizes intraclass distances using a Mahalanobis distance. Minimum interclass distances are enforced by hard constraints. More details on these methods can be found in Chapter 3.

This Chapter describes a metric learning method we published in [14], which allows for a simple gradient descent optimization. Like NCA it uses soft transitions and optimizes a projection matrix in order to improve the k-NN classification performance. However, it does not use a local model, but a global model similar to MMC and MDM, our feature weighting method presented in the previous chapter.

5.1. Global Metric Learning

Our goal is to improve k-NN classification. Since k-NN classifies data points based on label frequencies in their neighborhoods, good error rates are more likely if the intraclass distances are small and the interclass distances are large. To adapt the distances, the data points $\mathbf{x}_i \in \mathbb{R}^n$ are projected linearly using a matrix $W = (\mathbf{w}_1, \ldots, \mathbf{w}_n) \in \mathbb{R}^{n \times m}$, where the number of output dimensions m can be smaller or equal to the number of input dimensions n. The projection matrix is optimized using a cost function with two parts weighted by a para-

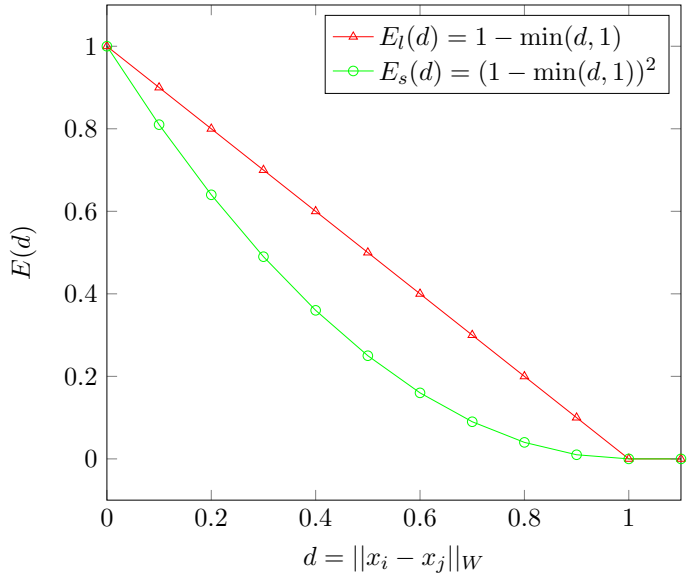


Figure 5.1.: Interclass cost function comparison. The linear function $E_l(d)$ has a constant gradient. Therefore, all interclass pairs influence the projection matrix W equally. For the non-linear function $E_s(d)$, close by pairs have a larger influence due to their steeper gradient compared to the far apart pairs. In addition, the non-linear function is continuously differentiable at the cutoff at $d = 1$.

meter α. The first part of the cost function punishes small distances of pairs from the set of interclass tuples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) : y_i \neq y_j\}$. If the distance one is reached, the cost will become zero due to a cutoff. We chose a squared cost term to penalize close interclass pairs significantly more than far apart pairs and to make a smooth transition at the cutoff. In Figure 5.1 the squared error term is compared to a linear term, where the gradient is constant and also not continuous at the cutoff. The second part punishes large distances of intraclass tuples from the set $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j\}$. We use the distance measure $||\mathbf{x}_i - \mathbf{x}_j||_W = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top W W^\top (\mathbf{x}_i - \mathbf{x}_j)}$ in the cost function

$$E(W) = \alpha \sum_{(i,j)\in\mathcal{D}} \left(1 - \min\left(||\mathbf{x}_i - \mathbf{x}_j||_W, 1\right)\right)^2 + (1 - \alpha) \sum_{(i,j)\in\mathcal{S}} ||\mathbf{x}_i - \mathbf{x}_j||_W^2. \quad (5.1)$$

Due to this purely cost function driven design without any hard constraints, there is always a trade off between minimizing and maximizing distances influenced by many tuples. There is a soft transition from tuples close to their desired distance with little influence to tuples far from their desired distance with large influence. All intraclass tuples are taken into account, making this a global ap-

proach with a Gaussian prior on the intraclass distances. Due to this property we coin this approach Global Metric Learning (GML).

There are many optimizers available that will find good solutions. We use stochastic gradient descent (SGD), because it works fast in case of redundant data. To avoid the need to select a learning rate, we applied variance-based SGD [29, 44]. Interestingly, in the context of visualizing high-dimensional data, Hadsell et al. [45] use almost the same cost term to find a non-linear mapping.

5.2. Experiments

To asses the behavior of GML, it is applied to basic 2D problem sets for visualization. In Figure 5.2 the results are shown. GML works well for the Gaussian problem sets. The uncorrelated perfectly separated data is reduced to one dimension. In case the data has a large overlap this dimensionality reduction, which would be at the cost of classification performance, is not achieved. The correlated and slightly overlapping data again is reduced to one dimension. GML, however, is not able to reduce the expendable dimension in the structured non Gaussian data. This is most likely due to the Gaussian prior introduced by the intraclass term. These results were all obtained with the weighting parameter set to $\alpha = 0.9$ to emphasize the interclass distances. We keep this setting also for the following experiments, even though tuning it for every dataset, e.g. by cross-validation, may be beneficial.

For evaluation on real world data we used the same UCI datasets as in the previous chapter. Additionally, the Balance Scale dataset from the UCI repository was evaluated. It contains 625 data points from three classes obtained in physics experiments on levers. Due to the linear relationships in the 4 dimensional measurements it is well suited for testing learned linear transformations. For each data set 10 different random splits of 50% training data and 50% test data were generated. GML was compared to LMNN, NCA and MMC via the k-NN classification error rates with $k = 3$. After adapting the representations to the training sets, the results were obtained by averaging the k-NN error rates on the test set over the ten splits. As a reference also the Euclidean metric was tested. Besides GML only LMNN has a weighting parameter, which was set to $\alpha = 0.5$ according to the authors advise. NCA and MMC are parameter free.

In Table 5.1 the 3-NN classification error rates after metric learning without preprocessing the data are shown, followed by the standard deviation in par-

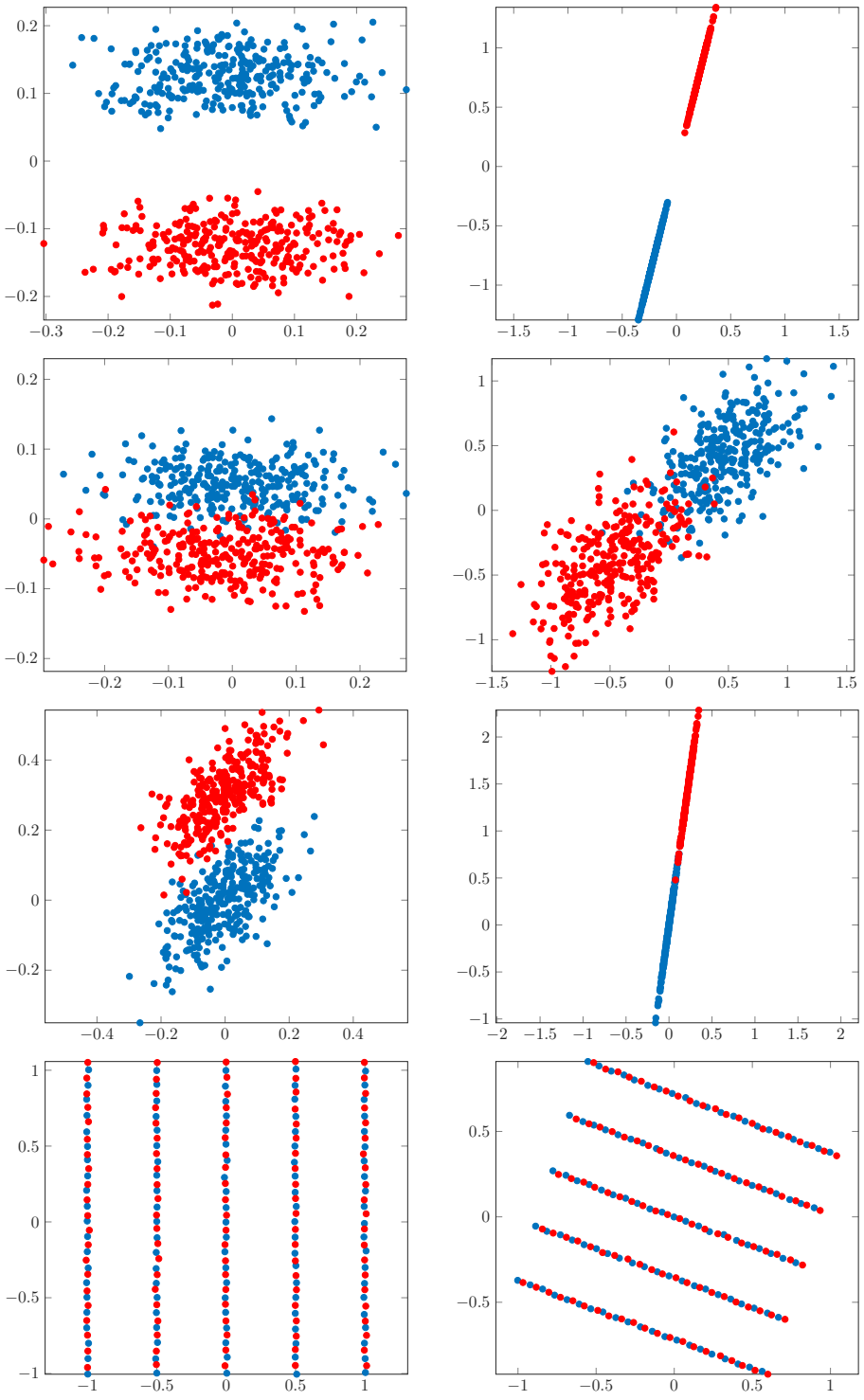


Figure 5.2.: Basic 2D datasets are depicted in the original scaling on the left and after application of GML on the right.

	LMNN	NCA	MMC	GML	Euclidean
Iris	3.07(1.89)	*4.13(1.93)*	**2.27(1.99)**	2.53(1.72)	3.33(1.81)
Wine	5.22(2.03)	*29.78(5.87)*	5.89(3.67)	**2.89(1.75)**	31.00(4.52)
Breast Cancer	3.80(0.57)	*39.71(2.24)*	**3.60(0.44)**	3.68(0.68)	39.68(2.21)
Pima Diabetes	29.14(2.33)	*31.12(1.94)*	**27.19(1.34)**	27.89(1.74)	30.78(1.99)
Balance Scale	*15.50(1.95)*	**7.60(2.87)**	10.00(1.14)	9.33(1.31)	21.73(1.36)
Parkinsons	14.18(2.90)	*17.24(2.47)*	16.63(3.57)	**10.10(4.04)**	16.73(2.46)
Seeds	6.95(2.88)	*7.71(1.76)*	6.86(2.14)	**4.10(1.19)**	11.33(2.64)

Table 5.1.: Classification results after metric learning. There was no preprocessing applied to the data sets. The error rates are given in percent followed by the STD in parentheses. The best results are marked in bold face and the worst in italic.

entheses. We can see that GML performs well on all datasets. In three cases it is slightly outperformed by MMC, and only on the Balance Scale data set it is significantly outperformed by NCA. In the three cases where GML is the best, it is by far the best. NCA is the worst on all other data sets and improves the classification performance only marginally or even deteriorates it compared to the standard Euclidean distance.

When preprocessing is done, the results for some methods change dramatically, as shown in Table 5.2. Here, each dimension is rescaled such that the data distribution has variance one. While all other methods benefit clearly from the preprocessing, there are only small changes in the results of GML and MMC. In fact, due to the global cost function, there should be no change at all. However, the stochastic gradient descent may not always find the global optimum in case of GML, and also the small changes in MMC seem to be due to convergence issues. GML still achieves the best results in three cases, and in the other four cases is never much worse than best one. Note that for all methods except for GML, there is always one dataset where it performs significantly worse than all the others (the worst results are marked in *italic*).

We also tested the feature weighting performance of GML. In Table 5.3 the results for preprocessed datasets are listed. The experimental set up and the preprocessing is the same as for metric learning, however, GML was only used to optimize the diagonal elements of W, leaving the off diagonal elements to zero. For comparison the feature weighting methods MDM, Relief, and Simba were used. Also in this feature weighting scenario GML performs well compared to the other methods and is again the best in three out seven cases. Of course,

	LMNN	NCA	MMC	GML	Euclidean
Iris	2.80(1.93)	*3.20(2.10)*	**2.27(1.99)**	**2.27(1.89)**	3.60(1.55)
Wine	3.00(2.10)	5.67(1.85)	*5.89(3.67)*	**2.67(1.67)**	5.67(1.52)
Breast Cancer	3.65(0.57)	*4.71(0.71)*	**3.60(0.44)**	3.83(0.76)	3.74(0.76)
Pima Diabetes	27.97(1.33)	*29.69(1.77)*	**27.19(1.34)**	27.92(1.76)	27.84(1.93)
Balance Scale	*14.41(1.90)*	**6.71(1.72)**	10.03(1.25)	9.74(1.08)	18.95(0.85)
Parkinsons	**9.49(2.26)**	10.71(2.86)	*15.10(3.36)*	10.71(3.13)	10.00(3.22)
Seeds	6.67(2.06)	7.43(1.33)	*6.86(2.14)*	**4.29(1.63)**	8.29(2.50)

Table 5.2.: Classification results after metric learning on preprocessed data. The dimensions of the datasets were normalized to variance one.

	MDM	Relief	Simba	GML	Euclidean
Iris	**2.93(1.97)**	3.20(2.10)	**2.93(1.51)**	*3.33(2.01)*	3.60(1.55)
Wine	4.00(2.23)	*4.22(2.39)*	4.00(2.11)	**2.67(2.11)**	5.67(1.52)
Breast Cancer	**3.54(0.67)**	4.06(1.02)	*4.12(0.71)*	4.06(0.79)	3.74(0.76)
Pima Diabetes	*28.49(1.87)*	**27.16(1.43)**	27.86(2.04)	28.26(1.64)	27.84(1.93)
Balance Scale	**18.95(0.85)**	19.17(1.31)	19.23(1.24)	*21.31(2.00)*	19.23(0.95)
Parkinsons	10.00(4.13)	9.18(2.50)	*10.82(3.73)*	**8.78(3.01)**	10.00(3.22)
Seeds	8.29(3.56)	9.62(3.06)	*10.19(3.39)*	**7.33(2.50)**	8.29(2.50)

Table 5.3.: Results for feature weighting. In a preprocessing step the dimensions of the datasets were normalized to variance one.

for feature weighting there is the same effect as observed for metric learning: Global methods are robust to the initial scaling, while local methods are effected heavily. Because the best results for the local methods were obtained in the preprocessed setting, we only show those.

As we have seen in the previous chapter, when the metric learning and the feature weighting results are compared (Tables 5.2 and 5.3), most error rates are quite close, showing that often a proper scaling of the dimensions is sufficient for good classification. Only for the Balance Scale and the Seeds datasets there are significant improvements when using the more powerful metric learning. To gain a better understanding we inspect the transformed spaces by t-SNE [46], which embeds the data in a 2D space prioritizing the accuracy of small distances. In Figure 5.3 exemplary results are shown. For both data sets the linear representation learning methods improve the representations. When comparing feature weighting to metric learning, for Parkinsons the results are qualitative similar. Therefore, the additional model complexity of metric learning does not help separating the classes, whereas, the wine data is separated much better by

the metric learning model. Only one data point is in the wrong cluster. Here, the similar results for metric learning are likely due to generalization problems, since there are only few data points in the wine data set. These results should be considered when choosing between feature weighting and metric learning. Additionally, scaling only the original dimensions has the advantage that the dimensions are not mixed, which makes it easier to interpret the results, e.g., extract relevant dimensions.

5.3. Discussion

We introduced a method to learn linear transformations for improved k-NN classification. Similar to our feature weighting approach from Chapter 4 we use a global model. This model with two weighted cost terms for the inter and intra-class distances can easily be optimized by gradient descent. Optimizing the sum of these terms yields a trade off between both. All intraclass pairs are taken into account introducing a Gaussian prior on the distances. This global approach reduces the influence of the initial scaling, and in most cases the global optimum of the energy function is found. This separates GML from well known methods such as LMNN and NCA, which due to their local approach are very sensitive to the initial scaling. A hard constraint as it is used for MMC is avoided, since such a constraint makes the results dependent on few interclass tuples with small distances. Instead GML uses a soft transition from very influential tuples with small interclass distances to a boundary distance with no influence.

While GML is not always the best method on the datasets we used, it was the best method most often (together with MMC), and it never performed much worse than the others. In this sense it was the most robust method. By training only the diagonal of the transformation matrix, GML can do feature weighting, which despite the lower model complexity often gives similar results. Improving the computational complexity of GML is an important next step, since the advantage of scale independence is acquired at the cost of a quadratic increase in computational effort with an increasing number of samples. This restricts GML to data of limited size, such as medical data sets. Often, many of the data points are not relevant for classification and, therefore, the method should take advantage of this.

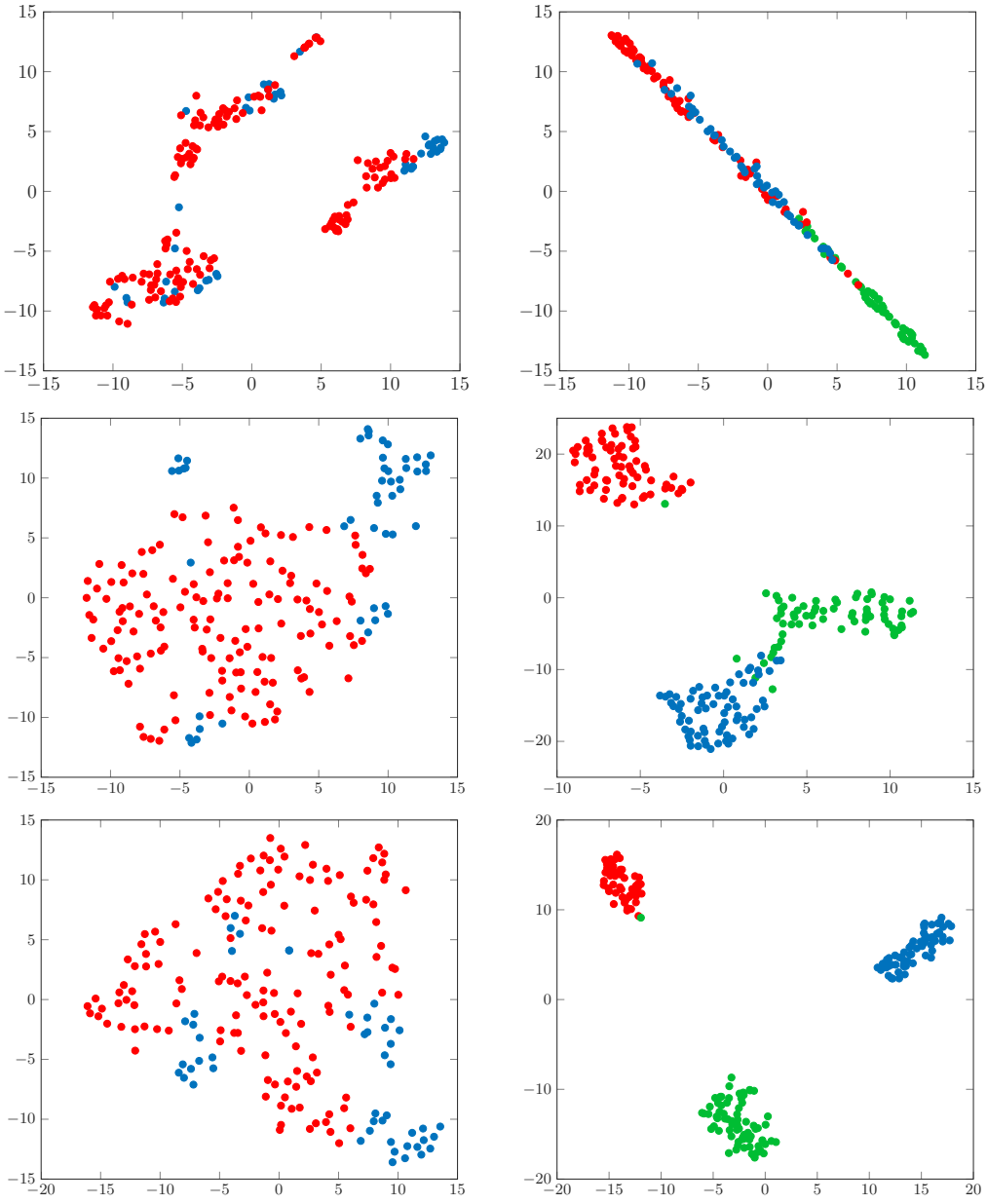


Figure 5.3.: Visualization of the UCI Parkinsons data on the left and UCI wine data the right. The dimensionality was reduced using t-SNE. In the top row the original data is shown. Below the results after feature weighting by GML and in the bottom row after metric learning by GML are shown.

Part II.

Invariant Representation Learning

6. Invariant Representations

When we have a walk outdoors and look around while taking steps forward the view changes slightly. For us it is still easy to recognize objects we know. Also we can learn to recognize new objects from only a few examples. In computer vision these visual object recognition tasks are big challenges. Every step we take will cause dramatic changes to the pixel representation of an 2D image. Even small changes of an object's pose like an animal moving its head as shown in Figure 6.1 are causing too large changes in the image pixels to recognize it as the same animal in all three images.

Figure 6.1.: Three images of a zebra moving the head. This movement causes changes to the pixel representation, which makes generalization for classifiers hard.

Much of the difficultly of object recognition arises from the way images are formed. Light is emitted from different sources and reflected by objects. The reflection depends on the material of the object and, therefore, information about that object is encoded in the light. A camera captures such reflected light by projection onto a 2D sensor plane. This projection contains only partial information on the object. Whenever the camera position or the object moves, different information is captured, and by occlusion this information may be reduced. Additionally, the position of the sensor pixels, where the information is collected,

changes. Besides these camera and object movements also changes of the light sources position, brightness, or color will change the image. For object recognition it would be good to have a representation which is independent of these changes, while encoding as much information about the object as possible. This independence property, called invariance, is topic of the next three chapters and will mathematically be defined in the next section. Our focus here are spatial transformations of the object in the image plane.

State of the art object recognition methods are usually trained on a large number of samples showing every object in different poses, lightning conditions, and from different perspectives. This helps to learn invariance implicitly, even though their optimization criterion is the classification rate on a training set. However, this invariance seems to transfer badly to objects unknown to the classifier, since for every new object a large set of samples needs to be collected for a good classification rate. This works of course for a limited number of objects or categories[1], but it might be impossible or tedious to collect large datasets. In case it is possible to learn a transformation invariant representation for images, the object recognition problem can be handled separately and becomes much simpler. Consider translation, scaling and in-plane rotation. If all these transformations can be applied to an object in an image in 100 discrete steps (Horizontal and vertical translation count as two transformations.), this allows for $100^4 = 10^8$ different images of the same object. In an invariant representation all these images will be the same. Therefore, the complexity of the classification task is reduced significantly and only a few samples are needed for good results, which is much closer to the human object recognition capabilities.

This chapter we begin by introducing basic mathematical concepts in order to define invariance. Then analytic invariant representations as well as existing methods for learning transformation invariant representations are reviewed. As a theoretical tool to understand and analyze invariant representations the i-Theory [15] is described and, finally, we show the close relation between the i-Theory and two representation learning methods. This contribution has previously been published in [17]. Based on the i-Theory and the slow subspace learning presented in this chapter, two new methods will be introduced and analyzed in the following two chapters.

[1] Even up to a thousand of categories as in the ImageNet dataset [47, 48].

6.1. Groups, Transformations, and Invariance

Many image transformations can mathematically be modeled by group structures. These group structures help to understand and analyze the problem of finding an invariant representation. Remember, a group $(G, \circ)$ is a set G together with an composition operation $\circ$ that satisfy four axioms:

- Closure: $\forall g, g' \in G : g \circ g' \in G$
- Associativity: $\forall g, g', g'' \in G : (g \circ g') \circ g'' = g \circ (g' \circ g'')$
- Identity element: $\exists! \, e \in G : \forall g \in G : e \circ g = g \circ e = g$
- Inverse element: $\forall g \in G : \exists g^{-1} \in G : g \circ g^{-1} = g^{-1} \circ g = e$.

In our context G is a set of image transformations (e.g. all in-plane rotations) and the group elements g are transformations with a specific parameter (e.g. rotation by 42 degrees). The definition of groups restricts possible transformations due to the existence of an inverse element, which implies that there is no information lost after applying the transformation. This excludes rotations in three dimensions. However, transformations such as in-plane rotation, translation, and scaling are included [24].

Often the admissible transformations are further restricted by the mathematical model the representation is based on. Widely used, due to their nice algebraic properties, are orthogonal matrix transformations[2] L, that is $L^{-1} = L^\top$. They include transformations like in-plane rotation, cyclic translations, and any pixel permutation.

Next we define invariance and uniqueness, two expressions often used in the following chapters, based on [24]. Suppose we have an image $\boldsymbol{x} \in \mathbb{R}^D$. It can be transformed by the group elements $g \in G$, which are applied as group actions $g(\boldsymbol{x})$. Our goal is to find a mapping μ which should be invariant in the transformation group G. Hence, if we have two images $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$,

$$\exists g \in G : \boldsymbol{x} = g(\boldsymbol{x}') \Rightarrow \mu(\boldsymbol{x}) = \mu(\boldsymbol{x}') \tag{6.1}$$

is true. This invariant mapping is sometimes also called signature. To be useful for discrimination we also need a unique representation, defined by

$$\mu(\boldsymbol{x}) = \mu(\boldsymbol{x}') \Rightarrow \exists g \in G : \boldsymbol{x} = g(\boldsymbol{x}'). \tag{6.2}$$

[2] They belong to the orthogonal group $O(n)$.

Much of the difficulty in representation learning arises due to the problem of retaining uniqueness while achieving invariance.

6.2. Analytic Representations

Historically, invariant representations were found manually. Here, I shortly review integral transformations, the cross-correlation method and moments as the most important examples of invariant analytic representations. In this case analytic means that the representation is obtained by some mathematical framework. This section is based on Woods review [49], which includes a comprehensive discussion of this topic.

A general class of transformations that can be used to find invariant representations are integral transformations. The general 2D integral transformation of a function $f(x, y)$ is

$$g(u, v) = \int_{-\infty}^{\infty}\int f(x, y)k(u, v, x, y)dxdy, \tag{6.3}$$

where $k(u, v, x, y)$ is a function that is often referred to as kernel and $g(u, v)$ is called the response. If a suitable kernel is chosen, an invariant representation can be obtained. One particularly well known example of an integral transformation is the Fourier transformation. Remember, the 2D Fourier transformation is defined as

$$\mathcal{F}(u, v) = \int_{-\infty}^{\infty}\int f(x, y)e^{-i2\pi(ux+vy)}dxdy. \tag{6.4}$$

So the kernel is $k(u, v, x, y) = e^{-i2\pi(ux+vy)}$. The amplitudes $|\mathcal{F}(u, v)| = \sqrt{Re(\mathcal{F}(u, v))^2 + Im(\mathcal{F}(u, v))^2}$ of this complex representation are invariant to shifts in the (x, y)-plane. Several adaptions of the Fourier-transformations have been made to account for different transformations such as rotation and scaling.

Another widely applied approach is cross-correlation. For estimating the transformation parameters (u, v), correlation coefficients

$$C(u, v) = \int_{-\infty}^{\infty}\int f(-x, -y)p(u + x, v + y)dxdy \tag{6.5}$$

with pattern vectors p are computed. Usually some prototype of an object is used as pattern vector. The largest correlation coefficient $C(u, v)$ indicates the

transformation parameters (u, v), which are then used to extract a patch taking the transformation into account. In the form presented here, only translation can be handled. However, an extension to other transformations is possible at a considerable computational effort.

Algebraic methods to find invariant representations can be based on moments

$$m_{p,q} = \int_{-\infty}^{\infty}\int x^p y^q f(x,y)dxdy, \tag{6.6}$$

where $p, q \in \mathbb{N}_0$. The complete set of moments uniquely identify a function $f(x, y)$ and, vice versa, $f(x, y)$ determines uniquely the moments. To achieve invariance to translation the moments can, for example, be centralized

$$m_{p,q} = \int_{-\infty}^{\infty}\int x^p y^q f(x-x_0, y-y_0)dxdy, \tag{6.7}$$

with $x_0 = m_{1,0}/m_{0,0}$ and $y_0 = m_{0,1}/m_{0,0}$ as the center of gravity of $f(x, y)$. By taking quotients or powers of moments other invariances can be obtained.

6.3. Learning Methods

In the previous section methods were presented that are specifically designed to represent images invariant to certain transformations. This works well in case these transformations are known in advance. Here, we are interested in learning invariances, which allows us to adapt to unknown transformations. This section gives a brief summary of some central methods.

6.3.1. Convolutional Networks

An early learning architecture that helps to cope with image translations are convolutional neural networks (CNN) [3, 50]. They are the basis for many competition winning classifiers today in the context of deep learning. Usually, the good classification rates are achieved by using extremely large sets of training data. Similar to the classical multilayer perceptron (MLP) they use several layers of computation, which are trained in a supervised manner.

First we have a look at the MLP. This is a network of so called neurons as basic units. These neurons are arranged in layers, which are stacked. Each neuron in

the network is modeled using a weight vector $\boldsymbol{w}_i$, a scalar bias b and a non-linearity σ (e.g. $\sigma(x) = \tanh(x)$). The activation $y_i^{(l)}$ of a neuron at layer l is

$$y_i^{(l)} = \sigma(\sum_j w_{i,j}^{(l)} \cdot y_j^{(l-1)} + b_i^{(l-1)}). \tag{6.8}$$

In case the input data samples are not vectors, they are reshaped to form vectors that can be used as input $\boldsymbol{y}^{(0)}$ to the network. Starting with random initialization the weight vectors and biases can be trained using back-propagation [51] given a suitable energy function, which could for example be the mean squared error between the output of the network and the labels of a training set.

CNNs have two main processing layers, which are applied in an alternating manner (Figure 6.2). First there is the convolutional layer. The outputs $M_k^{(l)}$ at layer l are called maps. These maps are computed by

$$M_k^{(l)} = \sigma(W_k^{(l)} * Y^{(l-1)} + b_k^{(l-1)}), \tag{6.9}$$

where $W_k \in \mathbb{R}^{U \times V \times D}$ is a convolution kernel and $Y^{(l-1)} \in \mathbb{R}^{I \times J \times D}$ is the output from a previous layer or the input data. The convolution operation is denoted by $*$. Usually, the input is not reshaped as it is done for MLPs. Therefore, the spacial relations in the data are retained and local connections in the network are implemented by convolution. The convolution operation also causes repeated use of these local connection, which is known as weight sharing. If we look at a single map point $m_{i,j,k}^{(l)}$ for $D = 1$

$$m_{i,j,k}^{(l)} = \sigma(\sum_{u,v} w_{u,v,k}^{(l)} \cdot y_{i+u,j+v}^{(l-1)} + b_k^{(l-1)}) \tag{6.10}$$

we see the local and repeated use of connections as well as the similarity to the MLP computations. In contrast to the MLP, the number of parameters is reduced significantly due to these local, repeated connections. The next layer of the CNN does subsampling over regions of these maps as well as over maps. Often the average, the square or the maximum value of these regions is taken. This subsampling is also referred to as pooling. After several alternating convolutional and subsampling layers, fully connected layers like for the MLP are used (Figure 6.3). CNNs are trained like MLPs using backpropagation.

What makes CNNs interesting in the context of learning invariant representations is their ability to learn invariance indirectly if enough transformed versions of the training samples are presented. Any classifier can do this. However, CNNs are particularly well suited. Shifts in the input will cause shifts of the

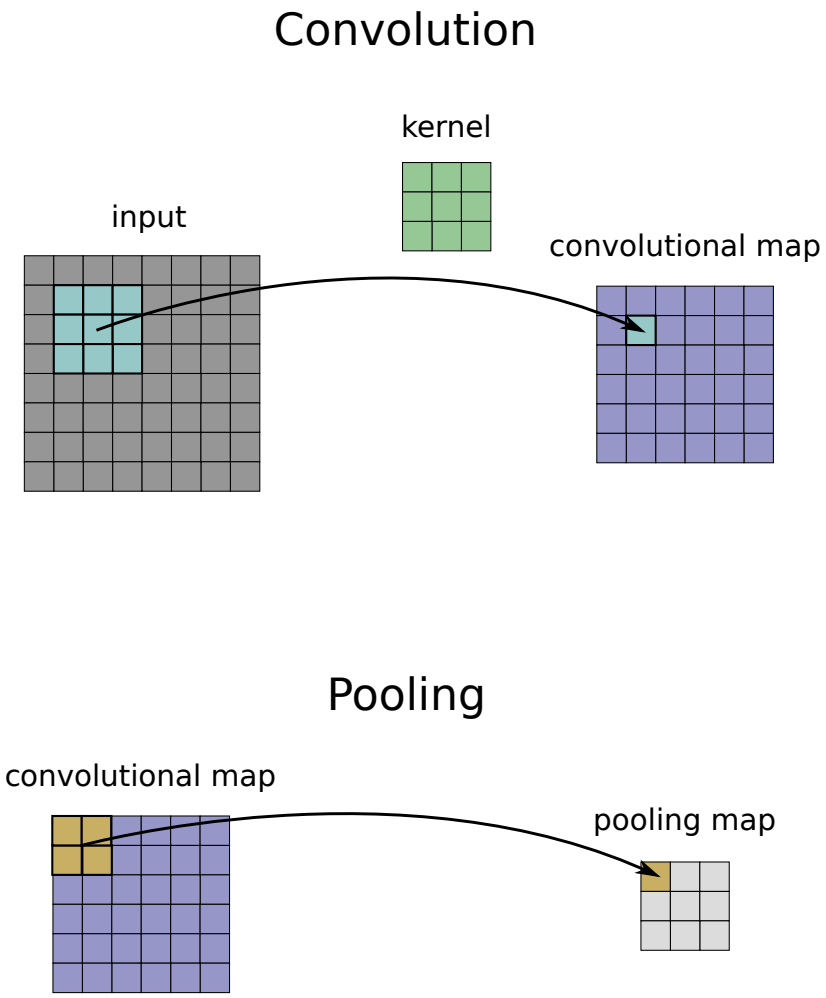


Figure 6.2.: Visualization of convolution and pooling. In the convolution step, a kernel is compared with the input data and the result is stored in the convolutional map. This comparison is repeated for all shifts of the kernel on the input. Note, how the spatial relations are preserved. Next the convolutional map is downsampled by pooling. This is done by some nonlinear function like the average or the maximum over several values from the convolutional map.

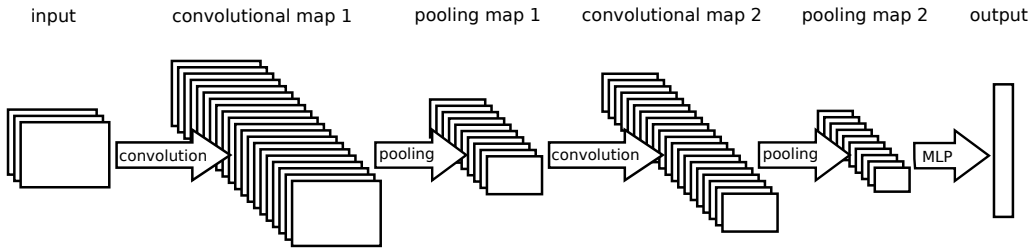


Figure 6.3.: A convolutional network architecture of alternating convolution and pooling steps followed by a multi-layer perceptron. This is the most common architecture, which, however, in practice has more convolution and pooling steps.

representation at each layer due to the weight sharing. This behavior is called equivariance. By subsampling the size of the shifts is reduced from layer to layer. Even though this architecture helps gaining transformation invariance, no guarantees can be given how well this invariance transfers to new classes. Therefore, many training samples are needed for every class. For a more principled approach to learning an invariant representation that generalizes well to unknown classes, an objective function that enforces invariance instead of classification seems mandatory.

6.3.2. Slow Feature Analysis

When a movie camera records two dimensional projections of the real world, a signal that changes fast in its pixel representation is generated. However, the objects in the scene change slowly. Therefore, it seems reasonable to optimize the representation of such a sequence to change slowly and thereby gain invariance to transformations contained in this temporal stream of data.

The Slow Feature Analysis (SFA) [52] is a prominent method exploiting this slowness objective [53, 54]. For some input signal $\boldsymbol{x}(t)$ SFA seeks a function $\boldsymbol{g}(\boldsymbol{x})$, which generates an output signal $\boldsymbol{y}(t) = (y(t)_1, y(t)_2, \ldots, y(t)_J)^\top$ with minimum average change

$$\langle \dot{y}_j^2 \rangle \tag{6.11}$$

for all outputs $y_j, j \in [1, \ldots, J]$ under the constraints

$$\langle y_j \rangle = 0, \tag{6.12}$$

$$\langle y_j^2 \rangle = 1, \tag{6.13}$$

$$\forall j' < j : \langle y_{j'} y_j \rangle = 0. \tag{6.14}$$

Here, angle brackets are used for temporal averaging and $\dot{y}_j$ is the temporal derivative of y_j. The zero mean constraint 6.12 and the unit variance constraint 6.13 prevent trivial solutions, while the decorrelation constraint 6.14 enforces a diverse set of outputs. Generally, this is a difficult problem. By restricting the mapping function $\boldsymbol{g}(\boldsymbol{x})$ for computing $\boldsymbol{y}(t)$, the problem becomes solvable. Only weighted sums

$$g_j(\boldsymbol{x}) = \sum_k w_{jk} h_k(\boldsymbol{x}) \tag{6.15}$$

of functions $h_k(\boldsymbol{x})$ are permitted. These functions, however, can be non-linear. Often $h_k(\boldsymbol{x})$ is chosen to be the quadratic expansion $h_k((x_1, \ldots, x_n)^\top)$ $= (x_1 x_1, x_1 x_2, x_1 x_3, \ldots, x_n x_n, x_1, \ldots, x_n)^\top$. This choice is guided [55] by computational limitations and the fact that linear and quadratic input-output relations have been measured in real neurons [56]. The quadratic expansion increases the data dimension dramatically, though.

6.3.3. Slow Subspace Learning

Subspace models have been introduced to the field of learning invariant features by Kohonen [57] and they have shown to be very useful, because many transformations can be modeled using linear subspaces [57].

A K-dimensional linear subspace can be represented by a set of orthonormal vectors $\boldsymbol{w}_i, i = 1, \ldots, K$. The projection

$$\boldsymbol{x}' = \sum_{k=1}^{K} (\boldsymbol{w}_k^\top \boldsymbol{x}) \boldsymbol{w}_k \tag{6.16}$$

of a data point $\boldsymbol{x}$ on the K-dimensional subspace finds the closest point $\boldsymbol{x}'$ in the subspace to $\boldsymbol{x}$. The length of this point $\boldsymbol{x}'$

$$||\boldsymbol{x}'||_2 = \sqrt{\sum_{k=1}^{K} (\boldsymbol{w}_k^\top \boldsymbol{x})^2} \tag{6.17}$$

in the subspace is used as a feature in many subspace methods. To get a meaningful representation several such features are needed.

Olshausen [58] motivates the use of two dimensional subspaces from a statistical perspective. Sparse coding or independent component analysis can reduce the statistical dependency of basis vectors $\boldsymbol{w}_i \in \mathbb{R}^N$ for images. When a basis $W = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M)^\top \in \mathbb{R}^{M \times N}$ for natural images is learned using one of these methods, Gabor like basis vectors are obtained. An image $\boldsymbol{x}(t) \in \mathbb{R}^N$ at time t in such a basis $\boldsymbol{x}(t) = W\boldsymbol{y}(t)$ is represented by coefficients $\boldsymbol{y}(t) \in \mathbb{R}^M$. When an object in $\boldsymbol{x}(t)$ moves, the coefficients in $\boldsymbol{y}(t)$ will transition among each other. In case pairs of $90°$ phase shifted Gabor functions are used in the representation, circularly symmetric distributions over the coefficients $\boldsymbol{y}(t)$ are cause by the transitions [59], so dependencies are introduced. Olshausen, therefore, suggests to model pairs of dependent coefficients by polar coordinates [58]. Using polar coordinates $M/2$ amplitudes $a_i(t)$ and phases $\theta_i(t)$ are needed to encode $\boldsymbol{y}(t)$. The amplitudes

$$a_i(t) = \sqrt{(\boldsymbol{w}_{i2}^\top \boldsymbol{x}(t))^2 + (\boldsymbol{w}_{i2+1}^\top \boldsymbol{x}(t))^2} \tag{6.18}$$

are equivalent to the length of $\boldsymbol{x}(t)$ projected on 2D subspaces, which are spanned by disjunct pairs of weight vectors $\boldsymbol{w}_{i2}$ and $\boldsymbol{w}_{i2+1}$. They can be used as invariant features encoding image structure. Additionally, phases

$$\theta_i(t) = \tan^{-1} \frac{(\boldsymbol{w}_{i2+1}^\top \boldsymbol{x}(t))^2}{(\boldsymbol{w}_{i2}^\top \boldsymbol{x}(t))^2}. \tag{6.19}$$

can be computed to determine the location of the image structures. Of course, these phases will not be invariant.

Kohonen introduced subspaces in the domain of self organizing maps [57]. Then they have been adopted to representation learning in form of the Independent Subspace Analysis (ISA) [60]. Soon slowness and subspace architectures were combined in [61], minimizing the energy change of the subspaces over time. Newer approaches [62–64] also include sparsity [9].

Here, I present a slow subspace autoencoder model [63] that has been applied successfully in object recognition tasks. It uses an autoencoder term to ensure that the learned representation will encode a diverse set of structures from the input data, while slowness over 2D subspaces encourages an invariant representation in the amplitudes $a_i(t)$. Additionally, this model allows to enforce a

sparse representation. It is defined by the energy terms

$$E_{rec} = \sum_t ||\boldsymbol{x}(t) - W^\top W \boldsymbol{x}(t)||_2^2 \tag{6.20}$$

$$E_{slow} = \sum_t ||\boldsymbol{a}(t) - \boldsymbol{a}(t-1)||_1 \tag{6.21}$$

$$E_{sparse} = \sum_t ||\boldsymbol{a}(t)||_1 \tag{6.22}$$

combined via

$$E = E_{rec} + \alpha E_{slow} + \beta E_{sparse} \text{ s.t. } ||\boldsymbol{w}_i|| = 1. \tag{6.23}$$

Here, α and β are weighting factors. Note, the unit norm constraint on the weight vectors $\boldsymbol{w}_i$. This is necessary to avoid $\boldsymbol{w}_i$ to become a zero vector if large values of α or β are used. This energy model can now be optimized via stochastic gradient descent.

6.3.4. Toroidal Subspace Analysis

Similar to the approaches from the previous subsection the Toroidal Subspace Analysis (TSA) [65] uses 2D subspaces for the learned representation. In contrast to the slowness based subspace approaches, the amplitudes of the subspaces are fixed for pairs of transformed image patches, and the error for encoding one patch in terms of the other is minimized. This requires to additionally model the phase.

The Toroidal Subspace Analysis (TSA) [65] learns invariant representations assuming that images undergo transformations that can be modeled by an orthonormal matrix L, which can be factored $L = WR(\boldsymbol{\varphi})W^\top$. Here, W is an orthonormal matrix that encodes the transformation group, while $R(\boldsymbol{\varphi})$ is a block diagonal matrix

$$R(\boldsymbol{\varphi}) = \begin{pmatrix} R(\varphi_1) & & \\ & \ddots & \\ & & R(\varphi_J) \end{pmatrix} \tag{6.24}$$

with the 2×2 submatrices

$$R(\varphi_j) = \begin{pmatrix} \cos(\varphi_j) & -\sin(\varphi_j) \\ \sin(\varphi_j) & \cos(\varphi_j) \end{pmatrix} \tag{6.25}$$

and the vector $\boldsymbol{\varphi}$ of J elements $\varphi_j \in [0, 2\pi]$, which selects an element from the group (e.g. it acts as a transformation parameter).

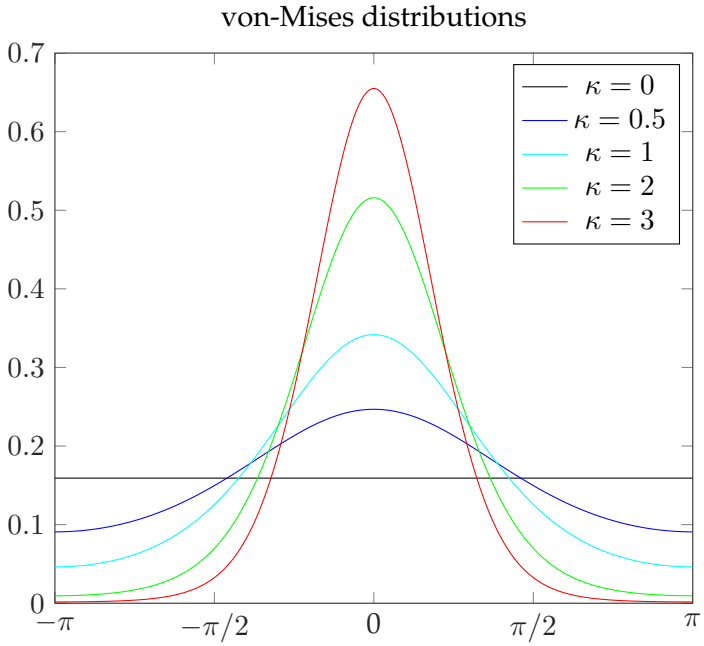


Figure 6.4.: Plot of the von-Mises distribution for different parameters κ and $\mu = 0$.

An image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is related in the TSA model by

$$\boldsymbol{x}_2 = WR(\boldsymbol{\varphi})W^\top \boldsymbol{x}_1 + \epsilon, \tag{6.26}$$

were ϵ is Gaussian noise $\mathcal{N}(0, \sigma^2)$. This relation can be used to express the probability of some $\boldsymbol{x}_2$ given $\boldsymbol{x}_1$ and the transformation parameters encoded in $\boldsymbol{\varphi}$

$$p(\boldsymbol{x}_2|\boldsymbol{x}_1, \varphi) = \mathcal{N}(\boldsymbol{x}_2|WR(\boldsymbol{\varphi})W^\top \boldsymbol{x}_1, \sigma^2). \tag{6.27}$$

The values φ_j are assumed to be marginally independent. Due to their periodicity, the vectors $\boldsymbol{\varphi}$ are modeled by a von-Mises distribution $\mathcal{M}(\varphi|\mu, \kappa)$, which is the circular equivalent of the normal distribution. The parameters μ and κ describe the circular mean and the concentration around this mean (Figure 6.4). Then for a decoupled model the probability of some image $\boldsymbol{x}_2$ given $\boldsymbol{x}_1$ is

$$p(\boldsymbol{x}_2|\boldsymbol{x}_1) = \int_{\boldsymbol{\varphi}} \mathcal{N}(\boldsymbol{x}_2|WR(\boldsymbol{\varphi})W^\top \boldsymbol{x}_1, \sigma^2) \prod_j \mathcal{M}(\varphi_j|\mu_j, \kappa_j) d\boldsymbol{\varphi}. \tag{6.28}$$

From this equation the derivative $\frac{d}{dW} \log p(\boldsymbol{x}_2|\boldsymbol{x}_1)$ for the log likelihood can be derived and W is found by gradient descent. Note, often the transformation

to be learned has much less parameters then J. This will result in coupled φ, which could also be included in the model.

6.3.5. Gated Models

Gated models [66, 67] minimize the encoding distance between pairs of transformed images, like TSA. However, the relation between images $\boldsymbol{x}$ and $\boldsymbol{y}$ is modeled via the product of image coefficients $\boldsymbol{u} = U\boldsymbol{x}$ and $\boldsymbol{v} = V\boldsymbol{y}$

$$z_k = \sum_f W_{kf}(\sum_i U_{fi}x_i)(\sum_j V_{fj}y_j) \tag{6.29}$$

with a vector $\boldsymbol{z}$ of latent variables encoding the transformation parameters. The basis matrices U and V are transformation dependent, while W is a matrix that encodes the correlation pattern between the bases. To impose more structure W is split into $W = TP$, where P is equal to one on all elements P_{ii} and $P_{i(i+1)}$. On all other elements P is zero. This splitted representation causes pairs of coupled weight vectors like the ones from the two previous subsections. Omitting T an invariant representation $\boldsymbol{r}$ for an image $\boldsymbol{x}$

$$r_k = \sum_f P_{kf}(\sum_i U_{fi}x_i)(\sum_j V_{fi}x_j) \tag{6.30}$$

can be obtained [68]. To estimate U, and V, an energy term for this model can be established by an autoencoder [69].

6.4. i-Theory

The i-Theory[3] [15] proposes a theoretical framework of how invariance could emerge in a feedforeward multilayer network with properties similar to the visual cortex. It assumes local modules, which have non-linear units that combine the outputs of linear units. These modules are then combined to a hierarchy. A strong theoretical foundation of this theory allows to derive properties for these modules and the hierarchy, which are beneficial to achieve invariant and unique representations. In the following we will describe this concept as detailed as needed for this thesis, and refer the reader to [15] for a more exhaustive description of the theory.

[3] It used to be called the M-Theory. However, in his newest talks Tomaso Poggio is referring to it as the i-Theory, presumably due to the naming conflict to an identically named theory in physics.

6.4.1. Basic Modules

First, we have a look at orbits. By applying all transformations $g_i \in G$ from the group G to some image $\boldsymbol{x}$ an orbit $O_{\boldsymbol{x}} = \{g_i(\boldsymbol{x}) | g_i \in G\}$ is induced. This orbit is unique for the object in $\boldsymbol{x}$, and it is invariant to the transformations in G. For example the group of in-plane rotations would induce an orbit containing all possible rotated versions of the original image $\boldsymbol{x}_1$. The orbit for some other image $\boldsymbol{x}_2 = g_i(\boldsymbol{x}_1)$ that can be obtained from $\boldsymbol{x}_1$ by rotation would be the same, because for both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ all possible rotated versions are contained in the orbit. Of course for some different image $\boldsymbol{x}_3$ that can not be obtained from $\boldsymbol{x}_1$ by rotation the orbit would be different.

For object recognition we would need to generate and compare the orbit of an unknown object to the stored orbit of a known object. However, it is not clear how to measure the similarity of two obits. One possibility is to use the probability distribution $P_{\boldsymbol{x}}$ induced by the transformations g_i on the image. For these distributions the following holds:

$$\boldsymbol{x}_1 \sim \boldsymbol{x}_2 \Longleftrightarrow O_{\boldsymbol{x}_1} = O_{\boldsymbol{x}_2} \Longleftrightarrow P_{\boldsymbol{x}_1} = P_{\boldsymbol{x}_2}, \tag{6.31}$$

where $\boldsymbol{x}_1 \sim \boldsymbol{x}_2$ denotes that $\boldsymbol{x}_1$ can be obtained from $\boldsymbol{x}_2$. However, these probability distributions are extremely high dimensional making it impractical to obtain them. Therefore, we would like to embed the invariance and discrimination properties of the distributions to a space of lower dimension. For $\boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^D$, the Cramér-Wold theorem [15, 70] ensures that these high dimensional probability distributions can be described by D distributions $P_{\langle g_i(\boldsymbol{x}), \boldsymbol{p}_n \rangle}$ over one dimensional projections $\langle g_i(\boldsymbol{x}), \boldsymbol{p}_n \rangle$, where $\boldsymbol{p}_n, n = 1, \dots D$ are the projection vectors (Figure 6.5). To discriminate a finite number of distributions, empirically a small number of projections $N < D$ is sufficient [15].

Instead of transforming the input image $\boldsymbol{x}$, we can also apply the inverse transformation to the projection vectors $\boldsymbol{p}_n$:

$$\langle g_i(\boldsymbol{x}), \boldsymbol{p}_n \rangle = \langle \boldsymbol{x}, g_i^{-1}(\boldsymbol{p}_n) \rangle. \tag{6.32}$$

By applying the transformations to the templates, we avoid transforming every new image. This allows an invariant and discriminative representation using two layers of computation. The first layer generates all the outputs using scalar products of all weight vectors $\boldsymbol{w}_{in} = g_i^{-1}(\boldsymbol{p}_n)$ with the input $\boldsymbol{x}$, and the second layer quantifies the distributions over the outputs of the first layer. The invariant

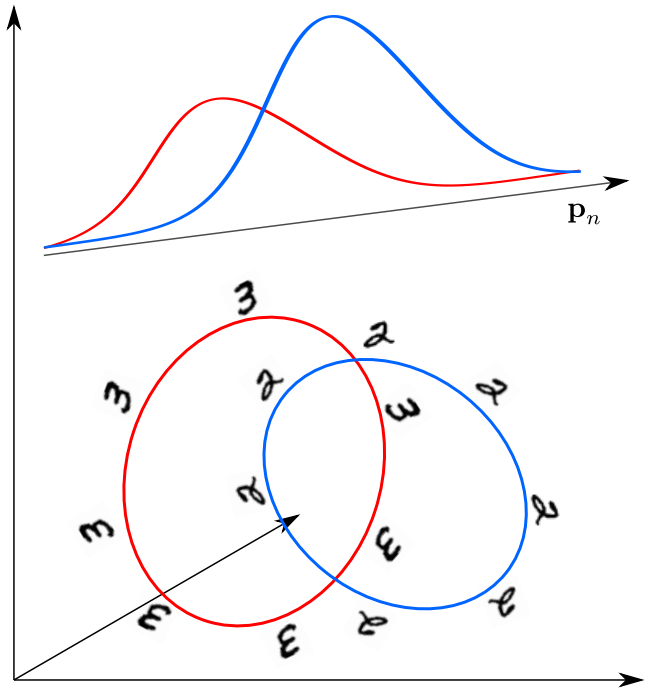


Figure 6.5.: The orbits of two images generated by applying all transformations from a transformation group are projected on a pattern vector $\boldsymbol{p}_n$ and produce distributions.

outputs μ_k^n of these two layers can therefore be calculated as

$$\mu_k^n(\boldsymbol{x}) = \frac{1}{|G|} \sum_{i=1}^{|G|} \nu_k \left(\langle \boldsymbol{x}, g_i^{-1}(\boldsymbol{p}_n) \rangle \right) \tag{6.33}$$

$$= \frac{1}{|G|} \sum_{i=1}^{|G|} \nu_k \left(\boldsymbol{w}_{in}^\top \boldsymbol{x} \right), \tag{6.34}$$

using a set of nonlinear functions $\nu_k, k = 1, \ldots, K$. These nonlinear functions could be powers $\nu_k(\cdot) = (\cdot)^k$, which allow to quantify the distribution by moments, or sigmoids $\nu_k(\cdot) = 1/(1 + e^{-((\cdot)-k\Delta)})$ together with a bias $k\Delta$ could be used to approximate a cumulative distribution function. In case moments are computed often a few are sufficient.

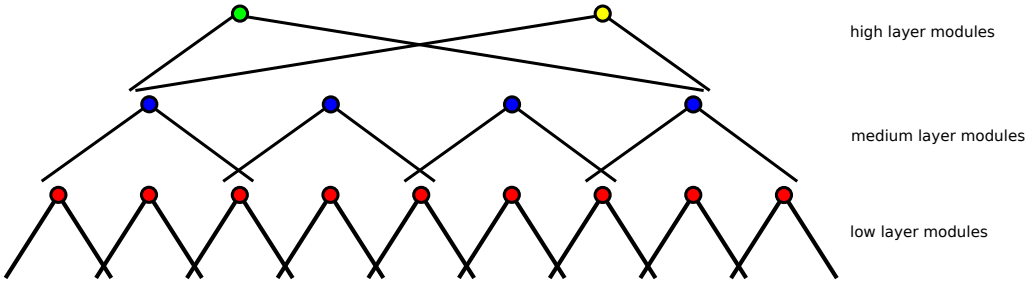


Figure 6.6.: A hierarchy of modules. The bottom modules receive an image as input and produce a representation that is invariant to local transformations. For higher layer modules the range of transformation parameters increases. At the top layer there are class specific modules, which can handle for example head movements in 3D. The output of these layers is fed to some classifier.

6.4.2. Real World Problems

In real world implementations we are limited to finite size vectors $\boldsymbol{w}$. This allows us to process in-plane rotations in the way described above. However, other transformations are only partially visible to $\boldsymbol{w}$. For example translations can move objects out of the observed area or scaling can result in objects larger than the observed area. For a limited range of parameters invariance to these partially observable groups can be achieved, if the pattern is sufficiently local. Local means in case of a single parameter i transformation group that

$$\langle \boldsymbol{x}, g_i(\boldsymbol{p}) \rangle = 0, \ |i| > a \tag{6.35}$$

holds for a given small maximum parameter a. For non-group transformations we relax this requirement to

$$\langle \boldsymbol{x}_C, g_i(\boldsymbol{p}) \rangle \approx 0, \ |i| > a. \tag{6.36}$$

Here, the images $\boldsymbol{x}_C$ are restricted to a class C of similar images. This implies sharply peaked correlations. If this is satisfied, approximate invariance to non-group transformations can be achieved.

So far we considered only a single module representing an isolated object. Global group transformations can be handled by such a module, but background clutter can change the entire representation. This can be avoided by several modules, which together observe the entire image, while each module observes

a local part of the image. The combined signature of these modules is only locally affected by clutter. Additionally, it takes a quite large set of vectors $\boldsymbol{w}_{in}$ for a single module to take multiple transformations into account. Unfortunately, it is in general not possible to factorize transformations (i.e. first obtain translation invariance in one module and then scale invariance in the next module), it is, however, possible to factorize the range of invariance, which can again be done by using multiple local modules [71]. Such a layer of local modules will produce a signature invariant to local transformations. If we want to increase the range of invariance or handle complex transformations, which can approximated locally by simple transformations, we can stack layers to build a hierarchy as shown in Figure (6.6). The range of invariance will increase with each layer. For this hierarchical approach it is necessary that the covariance property is satisfied. That is, the output of each layer transforms according to the same transformation group as the layer input [71], i.e. if an input image is rotated, also the output of the layers is rotated.

6.4.3. Relation to Biology

Hubel and Wiesel proposed that the area V1 in the visual cortex consists of modules of cells performing nonlinear computations on the outputs of linear cells [72]. These linear cells are usually referred to as simple cells, while the nonlinear cells are named complex cells. This is exactly replicated by the modules of the i-Theory. Here, the modules also have two layers. The first layer uses only scalar products, a linear operation. The second layer combines these first layer results in a nonlinear way, which is biologically plausible if specific nonlinearities such as sigmoid functions or quadratic functions are used.

Also it is a widely accepted theory that visual information is processed hierarchically from a lower level region called V1 up to a high level regions such as IT. This is closely resembled by the i-Theory. However, there are also many feedback connections in the visual cortex, which are not used or explained by the i-Theory.

6.5. Relations of the i-Theory to Learning Methods

The i-Theory provides a quite general framework for invariant architectures to which several invariance learning methods can be related. To show these relations in the following subsections, we first consider projections of the orbit.

If we assume that the transformations g_r from the group G have only one parameter r (e.g. degree for rotation) and they are ordered by this parameter, we can assemble a matrix W. This matrix $W = (g_{r_1}^{-1}(\boldsymbol{p}), g_{r_2}^{-1}(\boldsymbol{p}), \ldots, g_{r_N}^{-1}(\boldsymbol{p}))$ is composed of column vectors $\boldsymbol{w}_r = g_r^{-1}(\boldsymbol{p})$. The parameters r_i for the transformations are uniformly distributed $r_i = N/I \cdot (i-1)$ with I being the maximum transformation parameter. In the following line of thinking, we assume one projection vector $\boldsymbol{p}$. Here, we abbreviate g_{r_i} by g_i and w_{r_i} by w_i. The representation $\boldsymbol{y}$ of the image vector $\boldsymbol{x}$ is obtained by

$$\boldsymbol{y} = W^\top \boldsymbol{x}. \tag{6.37}$$

For the transformed image $g_j(\boldsymbol{x})$ we obtain $\boldsymbol{y}' = W^\top g_j(\boldsymbol{x})$. If we observe a single entry y_i of $\boldsymbol{y}$ while applying transformation g_j on $\boldsymbol{x}$

$$\begin{aligned} y_i &= w_i^\top \boldsymbol{x} = w_i^\top g_j^{-1}(g_j(\boldsymbol{x})) && (6.38)\\ &= g_j(w_i)^\top g_j(\boldsymbol{x}) && (6.39)\\ &= w_{i+j}^\top g_j(\boldsymbol{x}) = y'_{i+j}, && (6.40) \end{aligned}$$

we see that the entries of the representation vector shift indices. Only the first or last elements of $\boldsymbol{y}'$, depending on the direction of the transformation, may not be related to $\boldsymbol{y}$. By restricting the applicable transformations G to groups that can be observed completely by W (e.g. cyclic shifts and rotation), a relation to all entries in $\boldsymbol{y}$ can be established, because these group transformations are turned into circular shifts in the representation vector $\boldsymbol{y}$.

In the i-Theory the distribution over the entries in $\boldsymbol{y}$ are quantified, which of course does not change if only the indices are shifted. However, for learning methods it seems beneficial to keep the spacial information of the entries embedded in $\boldsymbol{y}$. The circular shifting of the entries can be described in the frequency domain. A Fourier transform of $\boldsymbol{y}$ will encode $\boldsymbol{y}$ with amplitudes invariant to the group transformation, while the phases encode the transformation parameter. Via the n-dimensional Fourier transform an extension to n parameters is possible.

6.5.1. Relating the i-Theory to the Toroidal Subspace Analysis

Using the above Fourier transform approach we can model the relationship of two images in order to learn the weight vector W and establish a connection to the i-Theory. Let $\boldsymbol{x}(t) = g(\boldsymbol{x}(t-1))$ at time t be a transformed version of an image

$\boldsymbol{x}(t-1)$ in a sequence. From above we know that the Fourier amplitudes will not change. Only the phase will change according to the Fourier shift theorem. Thus, we can reconstruct $\boldsymbol{x}(t)$ by

$$\boldsymbol{x}(t) = W^{-1}F^{-1}R(\phi)FW\boldsymbol{x}(t-1) \tag{6.41}$$

if the vector of phase shifts ϕ encoded in a diagonal matrix $R(\phi)$ is known. Here, the Fourier transform is applied via the matrix F. This reconstruction can be turned in a learning algorithm, where this autoencoder like energy term

$$E = \sum_t ||\boldsymbol{x}(t) - W^{-1}F^{-1}R(\phi)FW\boldsymbol{x}(t-1)||, \tag{6.42}$$

and $R(\phi)$ are optimized in an alternating manner. Since the Fourier transformation is just an unitary transformation, it can be absorbed into W

$$E = \sum_t ||\boldsymbol{x}(t) - W^{-1}R(\phi)Wx(t-1)||, \tag{6.43}$$

where W becomes complex. To avoid a complex difference vector, W and $R(\phi)$ can be turned into real valued matrices. In W each complex row vector W_i is replaced by two real ones $\Re(W_i)$ and $\Im(W_i)$. $R(\phi)$ then becomes a block diagonal matrix as described in equation 6.25. To simplify the computations the transpose instead of the inverse of W can be used

$$E = \sum_t ||\boldsymbol{x}(t) - W^{\top}R(\phi)Wx(t-1)||. \tag{6.44}$$

This is the essence of TSA [65] from Section 6.3.4. Note, by replacing the inverse of W by the transpose, only restricted transformations that can be handled, since $W^{\top}R(\phi)W$ is an orthogonal transformation.

6.5.2. Relating Slow Subspace Learning to the Toroidal Subspace Analysis

Next, we show the close relation between TSA and slow subspace approaches. Since TSA uses 2D subspaces to establish an invariant representation, a close relation to slow subspace approaches operating on 2D subspaces seems natural. Here, we consider the slow subspace autoencoder model presented in Section

6.3.3 omitting the sparsity term. Remember, the energy terms for that model are

$$E = E_{rec} + \alpha E_{slow} \text{ s.t. } ||\boldsymbol{w}_i|| = 1 \tag{6.45}$$

$$E_{rec} = \sum_t ||\boldsymbol{x}(t) - W^\top W \boldsymbol{x}(t)||_2^2 \tag{6.46}$$

$$E_{slow} = \sum_t ||\boldsymbol{a}(t) - \boldsymbol{a}(t-1)||_1 \tag{6.47}$$

with

$$a_i(t) = \sqrt{(\boldsymbol{w}_{i2}^\top \boldsymbol{x}(t))^2 + (\boldsymbol{w}_{i2+1}^\top \boldsymbol{x}(t))^2}. \tag{6.48}$$

And for the TSA we have

$$E = \sum_t ||\boldsymbol{x}(t) - W^\top R(\phi) W x(t-1)||. \tag{6.49}$$

The relation of subspace methods to TSA and the i-Theory can be seen if all energy terms are zero for any pair of group transformed images. Then subspace methods have found a basis W_{slow}, that can reconstruct all sample pairs $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t-1)$ from a sequence, while $\boldsymbol{a}(t)$ does not change for consecutive samples. Only the pairs of activations $w_{i2}^\top \boldsymbol{x}(t)$ and $w_{i2+1}^\top \boldsymbol{x}(t)$ can change over time. This change can be interpreted as an angle change in polar coordinates, which is the only change TSA allows to reconstruct $\boldsymbol{x}(t)$ from $\boldsymbol{x}(t-1)$. From that, we see, any input image can be group transformed using W_{slow} and the matrix $R(\boldsymbol{\phi})$ for the angle change. Therefore, W_{slow} is also an optimal solution for the TSA model. The other way round, an optimal basis W_{TSA} learned by TSA, will always have a fixed $\boldsymbol{a}(t)$, and perfect self-reconstruction is guaranteed via not transformed pairs $\boldsymbol{x}(t)$ and $\boldsymbol{x}(t-1)$. Thus, W_{TSA} is also an optimal solution for the subspace model.

7. Distribution Based Invariance Learning

What is a good framework for learning invariant representations? The previous chapter reviewed several invariance learning methods. None of these methods has shown success on non-affine data, except for the convolutional networks. These convolutional networks, however, are not guaranteed to transfer the invariance property well to unknown classes. For the other methods, possibly the mathematical model for the representation is too restrictive. Here, we propose a new invariance learning method based on the i-Theory [15]. This theory describes a quite concrete model for invariance. It assumes a hierarchy of modules (see Chapter 6.4), where each module works in the same way. These modules even allow approximate invariance to rotation in 3D.

To achieve invariance to a transformation group, the weight vectors in the modules need to resemble an orbit induced by the transformation group. One known approach to finding such weight vectors is to apply the transformations from the transformation group to a random pattern. This, however, requires a method for performing the group transformations on the input [15]. Another approach extracts the weight vectors from data, but it needs a well prepared set of training data and does not learn the weight vectors [73]. In contrast, we try to find the weight vectors by supervised training. The contributions in this chapter have been published previously in [16].

7.1. Learning Method

We follow closely the framework provided by the i-Theory for defining our invariant representation. There are two layers of computation to gain invariance to the transformations g from the group G. For an input image $\mathbf{x}$ the first layer is given by scalar products $\boldsymbol{w}_{in}^{\top}\boldsymbol{x}$ with the weight vectors $\boldsymbol{w}_{in}, i = 1, \ldots I, n = 1, \ldots N$, which generates a projected orbit representation of $\mathbf{x}$. After training,

the weight vectors are supposed to contain N patterns each in I transformed versions that resemble an orbit. The second layer quantifies the distributions of the first layer outputs by moments $m \in M$ out of a set of moments[1] M. So the overall output for the input image $\boldsymbol{x}$ is

$$y_{nm}(\boldsymbol{x}) = \sum_{i}^{I} \left(\boldsymbol{w}_{in}^{\top} \boldsymbol{x} \right)^{m}. \tag{7.1}$$

The quantities I, N, and the set M are parameters to be set. I describes the resolution of the orbit and thus how invariant the representation can be. Larger N and N will allow to discriminate more distributions and thus objects.

By optimizing the weight vectors $\boldsymbol{w}_{in}$, we hope to obtain outputs $y_{nm}(\boldsymbol{x})$ invariant to the transformations $g \in G$. We follow an approach similar to the methods introduced in the Chapters 4 and 5. We improve the ratio between intraclass distance and interclass distance. The main differences is that we assume the same labeled inputs to be from the same orbit and thus we expect that the intraclass distance can be minimized to approximately zero. When the intraclass distances become zero invariance is reached. To implement this idea we establish an energy function. One term of this function will quantify the intraclass distances. Instead of measuring the distance between pairs of input images, the distance to a class prototype is used. This prototype has target values t_{cmn} for every class $c \in C$, moment $m \in M$, and all N projection patterns in the weight vectors. Using the prototype, we can avoid an energy term where the computational complexity rises quadratically with the number K of images per class. We minimize

$$E_{\mathcal{S}} = \frac{1}{K} \sum_{k=1}^{K} \sum_{m \in M} \sum_{n=1}^{N} \left(t_{cmn} - \sum_{i} \left(\boldsymbol{w}_{in}^{\top} \boldsymbol{x}_{k} \right)^{m} \right)^{2} \tag{7.2}$$

to obtain invariance to the transformations in the training set. However, this term can produce trivial solutions. To get outputs useful for classification, the second energy term promotes a minimum interclass distance. Conveniently, this can be done via the prototypes of all tuples of different classes c and c'

$$E_{\mathcal{D}} = \frac{1}{|C|(|C|-1)} \sum_{c,c'} \max\left(1 - ||\boldsymbol{t}_{c} - \boldsymbol{t}_{c'}||, 0\right)^{2}, \tag{7.3}$$

[1] Note, we used moments. In a first attempt we tried histograms with Gaussian and cumulative distribution functions with sigmoids. These are, however, hard to train since their derivative functions are close to zero for many inputs.

with the target vectors $\boldsymbol{t}_c = \left(t_{c,1,1}, t_{c,1,2}, \ldots, t_{|C|,|M|,N}\right)^\top$. Here, $|\cdot|$ identifies the number of elements in a set. To prevent the distances from becoming infinitely large, only those distances smaller than one contribute to the energy. The overall energy is then a combination of $E_\mathcal{S}$ and $E_\mathcal{D}$ weighted by the factor α

$$E = \alpha E_\mathcal{S} + (1-\alpha) E_\mathcal{D}. \tag{7.4}$$

For training, the targets vectors t_{cmn} and the weight vectors $\boldsymbol{w}_{in}$ are initialized randomly. Then the energy term (7.4) is minimized by a gradient optimization to find t_{cmn} and $\boldsymbol{w}_{in}$. We used the Sum of Functions optimizer [30] for the gradient optimization.

7.2. Distance to Center Classification

Invariant representations allow for simple and elegant classification. All images $\boldsymbol{x}^*$ in a class c^* will be represented by outputs $\boldsymbol{y}(\boldsymbol{x}^*)$ that lie exactly on the target vector $\boldsymbol{t}_{c^*}$. If the representation is only approximately invariant, the outputs $\boldsymbol{y}(\boldsymbol{x}^*)$ will be clustered around $\boldsymbol{t}_{c^*}$. Therefore, the class label c for an image $\boldsymbol{x}$ can be determined by the target vector closest to $\boldsymbol{y}(\boldsymbol{x}^*)$:

$$c^* = \arg\min_c ||\boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{t}_c||, \tag{7.5}$$

with $\boldsymbol{y} = \left(y_{1,1}, y_{1,2}, \ldots, y_{|M|,N}\right)^\top$. This strategy we call distance to center classification. In case invariance was not reached, large output clusters may be obtained. Then distance to center classification might not work perfectly.

7.3. Experiments

In this section we test the learning model first on artificial data to see if it is behaving as we expect it and then measure the influence of several parameters. We used binary patches of size 4×4 pixels. Each pixel was randomly set either to one or to minus one with probability 0.5. These patches were shifted using periodic boundary conditions, resulting in 16 transformed versions of every patch (Figure 7.1). Then the capability is tested on handwritten digits from the MNIST dataset [3] (Figure 7.2) as a small real world example. For this dataset we assume that each digit can be generated from some prototype by an unknown nonlinear transformation. For training there are 60000 samples available, while the test set contains 10000 samples.

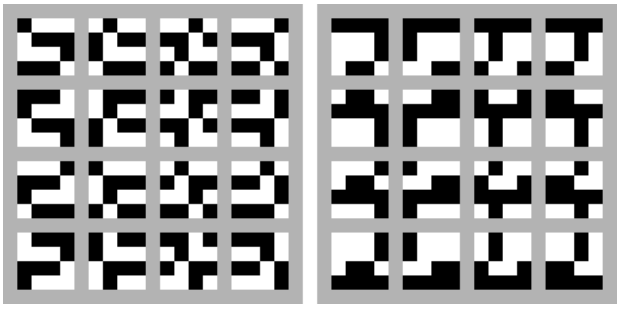


Figure 7.1.: The image shows 2 example patches in all their 16 possible shifts. The circular shifts used are well understood group transformations.

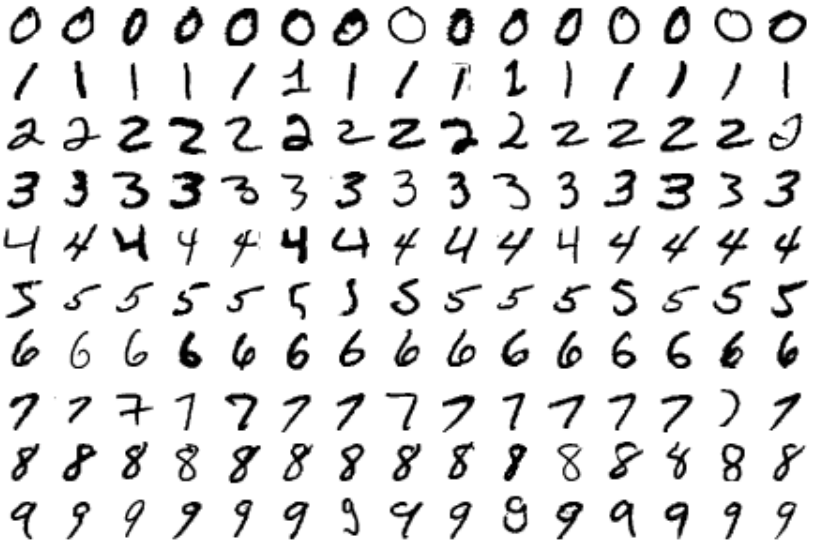


Figure 7.2.: Hand written digits from the MNIST data set. For every row we assume that the digits can be converted into each other by some unknown transformation.

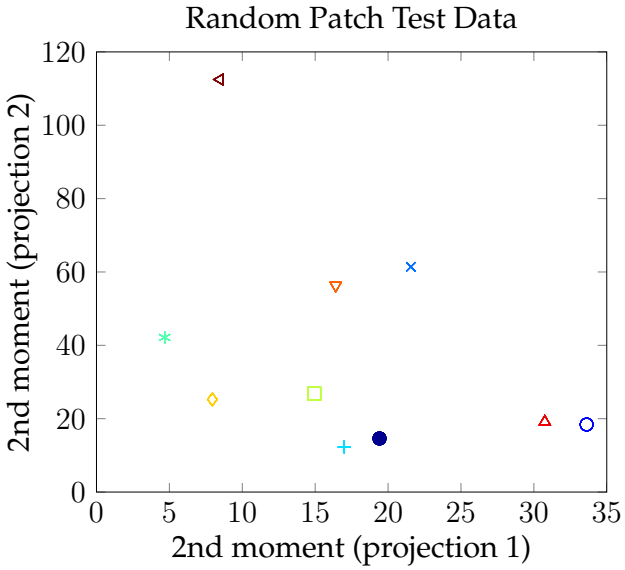


Figure 7.3.: The plot shows the artificial test data in a 2D representation of our model. Each patch is denoted by a different shape and color. Perfect shift invariance is achieved since all shifted versions of a patch fall on the same point. The 2D coordinates for each point were obtained from the second moments of two orbit projections, i.e., $y_{1,2}$ and $y_{2,2}$ from Equation (7.1).

The only model parameter we did not change during the experiments was α, which controls if interclass or intraclass distances are emphasized. It was set to 0.01 for all experiments. This was mainly due to problems of diverging energies for some parameter settings when training on the MNIST data. Additionally, we observed faster convergence on MNIST data.

As a proof of concept we trained the model on the artificial data. For training we generated 100 patches randomly and obtained via the transformations 1600 samples. On these samples we trained the model with $N = 2$ orbit projections and $I = 16$ weight vectors per projection. I is determined by the size of the orbit induced by cyclic translations on 4×4 pixel images in a discrete setting, which is 16. The two orbit projections were chosen for visualization. After training we tested the model using 160 samples obtained from 10 random patches generated like the training images. In Figure 7.3 the results are shown. For each sample $\mathbf{x}$ we obtain two values, the second moment for each projection as described in Equation 7.1. Therefore, each point can be represented in two dimensions. Since all transformed versions of a sample fall on a single point, this representation is

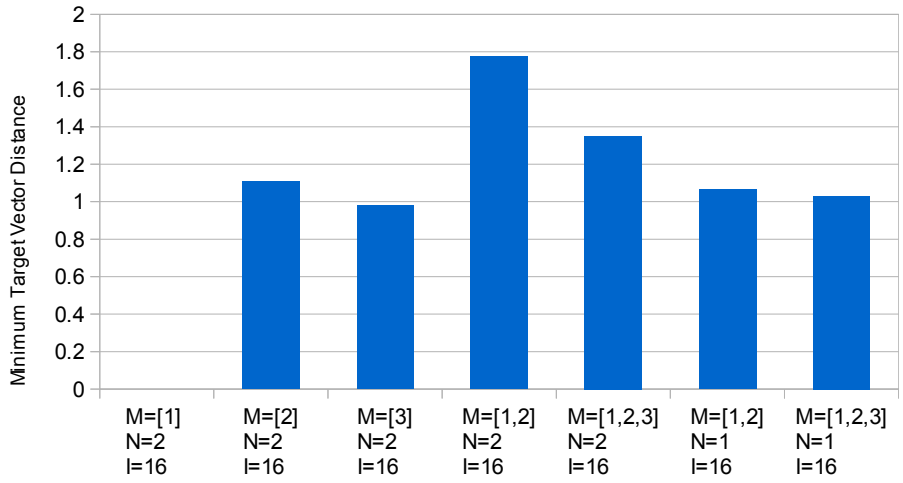


Figure 7.4.: The minimum distance between target vectors averaged over 100 trials. Here, the number I of weight vectors per projection was fixed at 16. The moments M and the number of projections N varies.

perfectly invariant.

The choice of the second moment in the previous experiment was arbitrary. It was motivated by the observation of the authors of the i-Theory that often one moment is sufficient. Here, we explore how different sets of moments influence the results. For this experiment we train the model on different sets of moments using the same training set as before. Then on a test set induced by 10 random patches we measure the min distance $(\boldsymbol{t}_c, \boldsymbol{t}_{c'})$ by $\arg\min_{(c,c')} ||\boldsymbol{t}_c - \boldsymbol{t}_{c'}||$ between all pairs of target vectors $(\boldsymbol{t}_c, \boldsymbol{t}_{c'})$. The results shown in Figure 7.4 are averaged over 100 trials. We see on average mostly a distance of one is reached. Only if the mean value ($M = [1]$) is used, the minimum distances become zero. All other moments, therefore, allow for a discriminative representation that generalizes well to unseen samples.

To see, if the representations learned are invariant, we measured the mean distance $\sum_k ||\boldsymbol{y}(\boldsymbol{x}_k) - \boldsymbol{t}_{c(k)}||/K$ of the data point representations to the target vectors they belong to. This was done using the same setup as for the target vector distances. The the mean distances where all well below 0.02, except if only the third moment was used, which resulted in an average distance of 0.35. So any combination of moments will find an invariant representation, only that third moment seems less useful on its own.

Next, we were interested how the number of weight vectors influence the res-

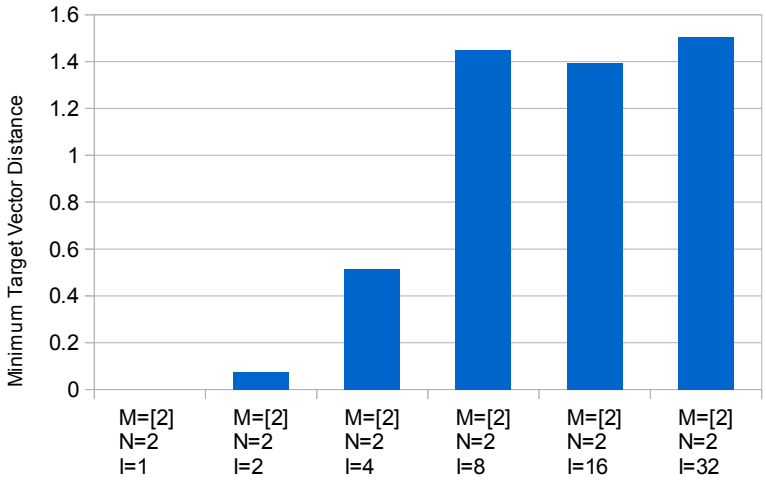


Figure 7.5.: The minimum distance between target vectors averaged over 100 trials. Here, the moment M was fixed at 2, and the number of orbital projections was also set to 2, while the number I of weight vectors per projection varies.

ult. For the artificial data, we know that the entire transformation orbit can be represented by 16 transformed versions of a pattern and, therefore, the projection of the orbit can be done using 16 weight vectors. In this experiment the number of weight vectors varied, while always the second moment and two orbit projections were used. We measured again the distances between target vectors and the distances from data points to the corresponding target vectors in the same set up as before. The average distances from data points to their target vector are all below 0.1, so approximate invariance was always reached. For the distances between target vectors we see in Figure 7.5 a decline of the minimum distances for a small number of weight vectors per projection, as expected. Interestingly, already 4 weight vectors per projection give a reasonable average minimum distance. This is well below the calculated size for the orbits.

To understand why less weight vectors are needed then the size of the orbit suggests, we plotted the learned basis functions. When using only the second moment the resulting weight vectors seem to have little structure. By using the moments $M = [1, 2, 3]$ the set of admissible weight vectors was reduced due to the better quantification of the distributions of the orbit projections. This constraint resulted in more structured weight vectors shown in Figure 7.6. Here, we see for 4 weight vectors per projection that the learned patterns are structured

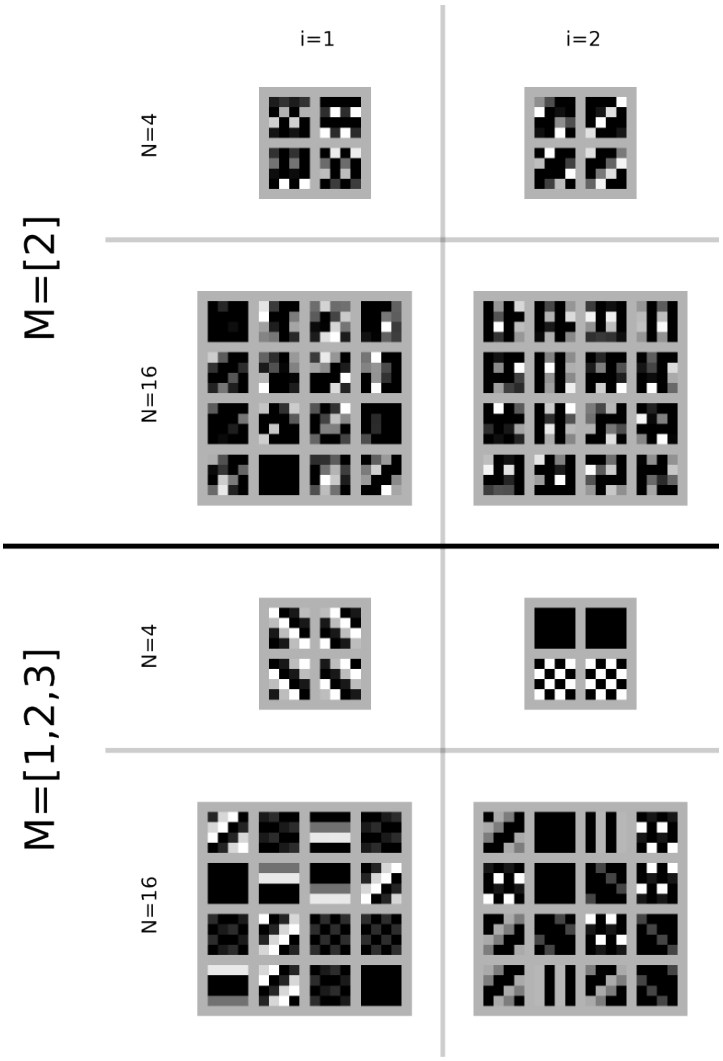


Figure 7.6.: Visualization of the weight vectors for different sets of moments M, number of weight vectors N per orbital projection, and orbital projection index i. Each weight vector was reshaped to the input pattern size of 4×4 pixels.

such that less than 16 weight vectors are needed. For $i = 1$ the patterns have almost constant intensity across the diagonal in one direction, and, thus, only the smaller transformation group of shifts orthogonal to that direction needs to be handled. This can be done using 4 vectors. For $i = 2$, two vectors are not used. The other two vectors have a structure, which is repeated for shifts larger than one. Therefore, only two vectors are needed. Also the larger model with 16 weight vectors per projection does not learn a single pattern in 16 transformed versions. It learns multiple patterns per projection, with similar structures as for the case $N = 4$. These results show that compared to arbitrary patterns, for certain structured patterns less weight vectors are needed to represent the orbits. Therefore, models with few weight vectors per orbital projection work.

Going one step further, we tested invariance to the nonlinear transformations in MNIST. This data is challenging, since the underlying transformations are unknown. We have no hint how to select the number of weight vectors per orbital projection, and, additionally, the number of parameters to learn for the 28×28 pixel images is much larger compared to the 4×4 pixel images from the artificial data.

A small model using two orbital projections with 20 weight vectors each was trained for the visualization shown in Figure 7.7. Each digit can be represented in 2D, because only the second moment was used for quantifying the distributions from the two orbital projections. The visualization of the test data shows ten nice clusters for the digits. Equally labeled digits are often not perfectly aligned, therefore, only approximate invariance to the transformations in MNIST was learned. However, a two dimensional representation may not offer enough degrees of freedom for a dataset as complex as MNIST. To improve separation, the number of projections can be increased. In Figure 7.8, the error rates on the test data for various parameter settings are shown. These error rates are obtained by distance to center classification described in Section 7.2. Clearly, the error rates significantly decrease for a larger number of projections. While the two projections used for visualization yield 16.63% error rate, the best error rate of 2.86% is achieved using 10 projections. However, more projections or weight vectors do not improve the results. Furthermore, we tried to use the moment set $M = [1, 2, 3]$, which, however, increased the error significantly.

In Figure 7.9 the 286 errors made by the best model on the 10000 test samples are shown. These errors have no clear pattern. Even samples easy to classify for a human are misclassified. This suggests that the transformations underlying

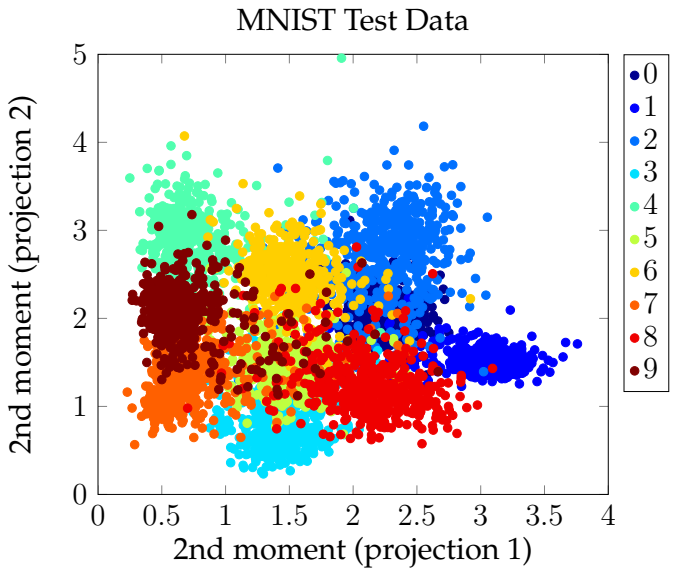


Figure 7.7.: This plot shows the MNIST test data in a 2D representation learned by our model. Each digit is represented by a point colored according to the label. Clearly, they form clusters.

MNIST are not well modeled. However, 17.97% of the samples are misclassified when the distance to center classification is applied in the input pixel space of 768 dimensions. This shows a significant improvement in the organization of the space.

7.4. Discussion

A new method for learning invariant representations was introduced. Using the i-Theory as framework, we hoped to gain invariance to transformations more general than state of the art methods can learn. In the experiments this supervised method showed invariance to periodic boundary translation. Hence, it is capable of learning simple transformation groups, where optimal parameter settings are known. For unknown transformations these parameters are of course not given. Therefore, they need to be found empirically. We tested our model on the MNIST data assuming each digit can be generated from a prototype digit via an unknown transformation. The learned representations were, however, not fully invariant. But they achieved a clustering in a low dimensional space with decent classification performance. Since increasing the number of projec-

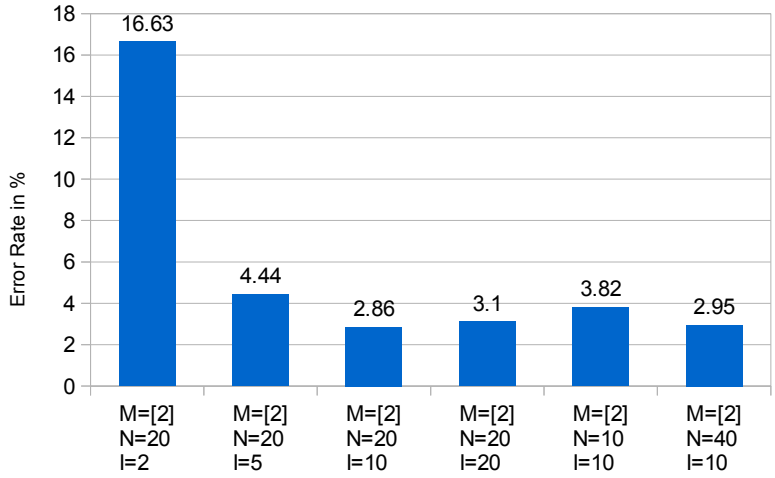


Figure 7.8.: Error rates on the MNIST test data. The moments are denoted by M, I is the number of weight vectors per orbit, and N is the number of orbit projections.

tion vectors per orbit and weight vectors per projection did not improve the invariance, there are either optimization issues or more general problems with the model.

Our analysis of the model showed that several moments are needed to obtain well structured weight vectors. This worked well for the artificial data, but not on MNIST. The supervised approach we developed may cause some of the problems with MNIST, since there are local optima in the energy function. Additionally, this supervised learning requires labeled data, which is not available for many transformations. Therefore, it seems necessary to adapt this learning method to unsupervised learning.

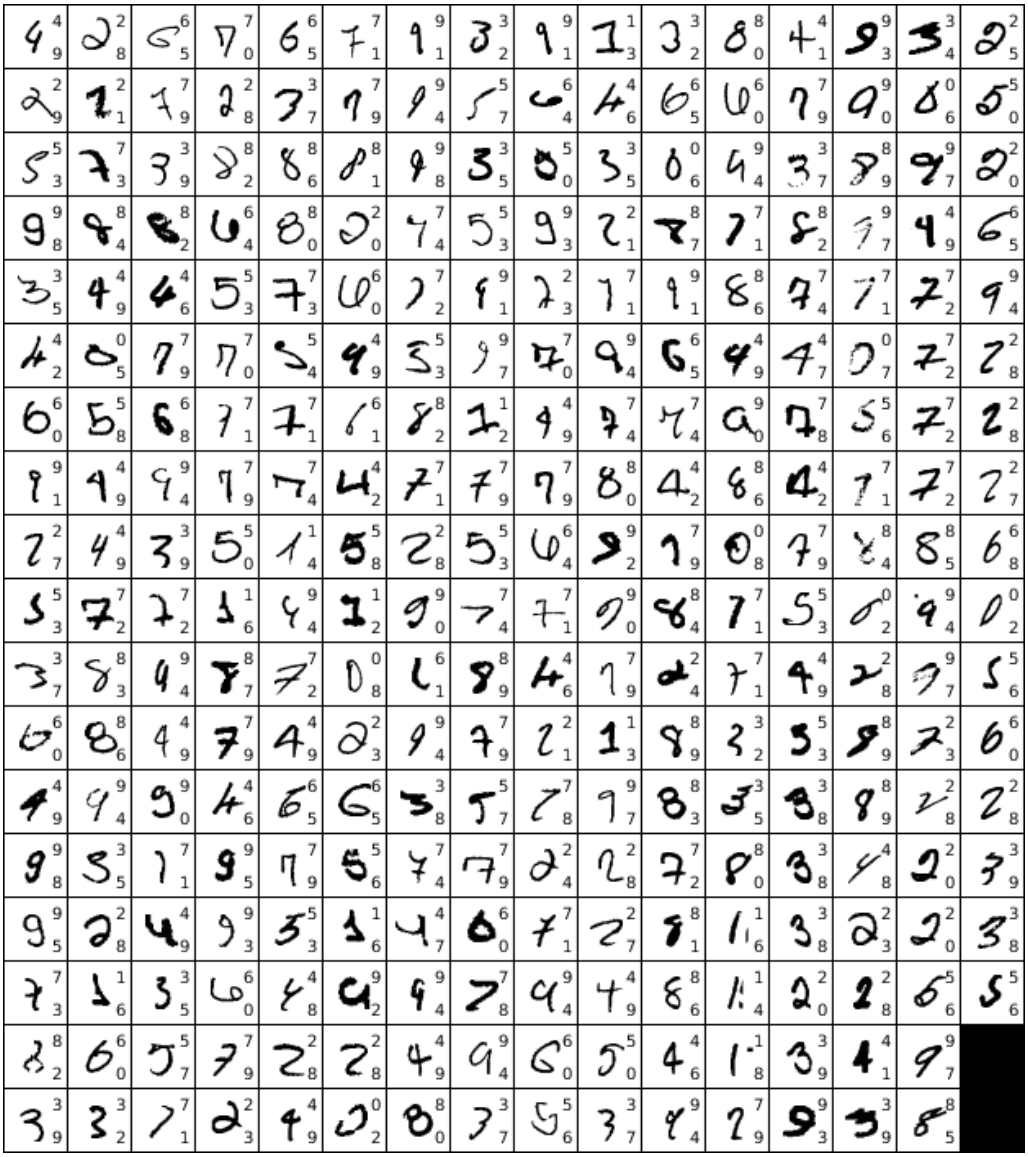


Figure 7.9.: The 286 errors made by a model with the parameters $M = [2]$, $N = 20$, and $I = 10$ on MNIST. At the top right of each digit the correct label, while on the bottom right the wrong label is displayed.

8. Convolutional Slow Subspace Learning

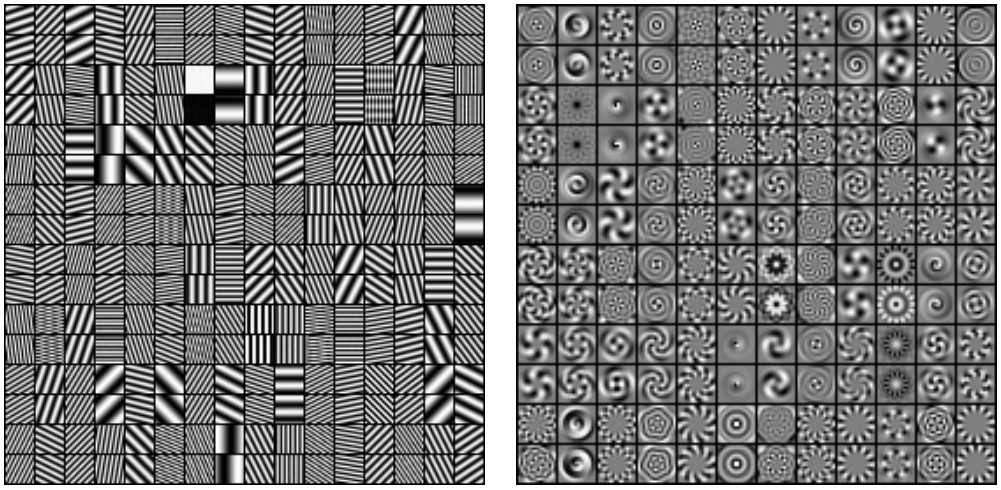
Objects in an image can usually be transformed by multiple transformations simultaneously (e.g. translation and rotation). The unsupervised representation learning methods from Chapter 6 can find representations invariant to many common transformations. However, they seem not to be able to handle more than one transformation group at a time. In this chapter, we examine an hierarchical architecture with a design guided by the i-Theory to approach this problem.

First, we consider the slow subspace autoencoder [63] (Chapter 6.3.3) omitting the sparsity term. We trained this model on pairs of random intensity patches, where one patch is a transformed version of the other patch. These pairs were used as very short image sequences. Figure 8.1a and 8.1b show the weight vectors for a subspace representation invariant to periodic boundary shifts and rotation. Remember, in slow subspace autoencoder pairs of weight vectors $\boldsymbol{w}_{i2}$ and $\boldsymbol{w}_{i2+1}$ are coupled to form amplitude coefficients

$$a_i(t) = \sqrt{(\boldsymbol{w}_{i2}^\top \boldsymbol{x}(t))^2 + (\boldsymbol{w}_{i2+1}^\top \boldsymbol{x}(t))^2} \tag{8.1}$$

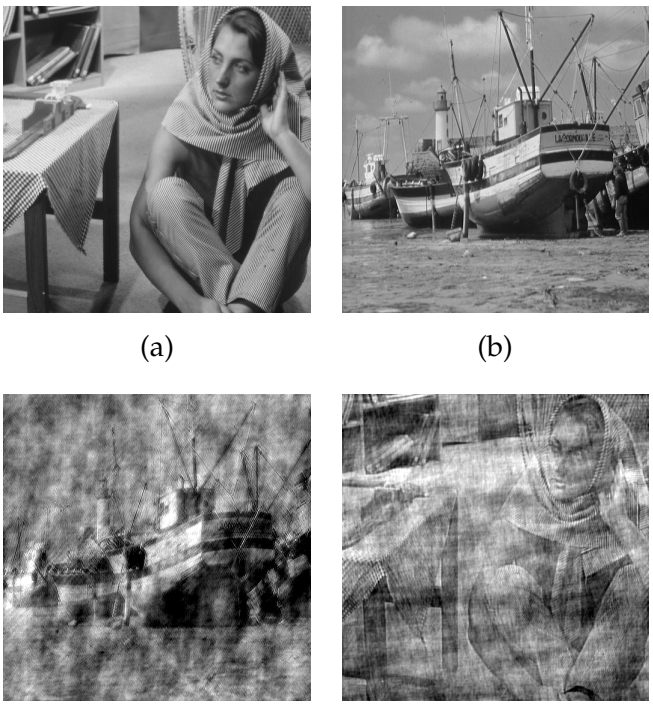
for an image vector $\boldsymbol{x}(t)$ at time t. Therefore, there are pairs of similar weight vectors in Figure 8.1. These pairs can be described by sine-shaped patterns, which in case of the rotation invariant representation are placed on a circle. $\boldsymbol{w}_{i2}$ and $\boldsymbol{w}_{i2+1}$ can be related by a 90° phase shift. These properties allow to represent an image in terms of invariant amplitudes and transformation dependent phases (Chapter 6.3.3).

Note that the weight vectors for the shifts closely resemble a Fourier basis, which usually is also represented by amplitude phase pairs. For the Fourier transformation the amplitudes are known to be shift invariant. However, it is also well known that a large amount of spatial information is encoded in the phases (Figure 8.2). Therefore, invariance is gained at the cost of uniqueness.



(a) (b)

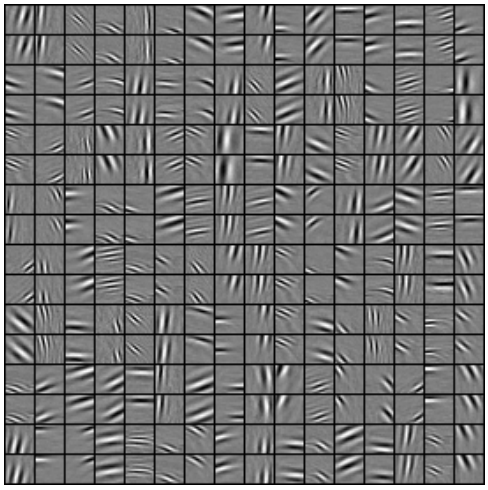
Figure 8.1.: Global bases learned from shifted and rotated patches of random intensities are shown in (a) and (b).



(a) (b)

(c) (d)

Figure 8.2.: When the Fourier phases of image (a) and (b) are exchanged while fixing the amplitude, the image (c) is obtained from (a) and (d) is obtained from (b). This shows how much image information is in the phases.



(c)

Figure 8.3.: A basis learned with a sparsity prior from natural movie sequences.

Since the rotation invariant representation also uses the amplitudes while omitting the phases, it is likely that uniqueness is also lost here.

Another problem of these representations is that factorizing the invariance problem by first gaining invariance to one transformation group in one module and then invariance to a second transformation group in the next module seems impossible [71]. However, factorizing the range of invariance is possible according to the i-Theory [15]. By factorizing the invariance it is restricted to a local window. This can be achieved for example via a sparsity term, since the pixels in natural images are highly correlated only in local neighborhoods. In Figure 8.3 we see the results for the complete slow subspace autoencoder model, which includes a sparsity term. The model was trained on natural movie sequences from the van Hateren data set [74]. Of course, local windows will decrease the invariance [75], but due to the local similarity of most transformations to shifts, a diverse set of transformations can be handled and more spatial information on the input is retained. Additionally, sensitivity to background clutter is decreased, as it only locally effects the representation.

In the following we will describe a similar convolutional subspace architecture we published in [17], which enforces local invariance by its structure. To increase the range of invariance, we add a second layer. Then we test the architecture for invariance to multiple transformations and make use of the convolutional structure to investigate whether redundancy can help to overcome the

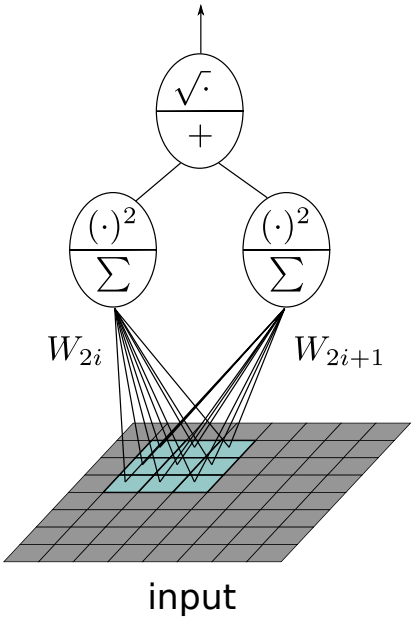


Figure 8.4.: Visualization of a module. Two weighted sums over an area in the input are squared and added. Finally, the root is taken to get a single output value.

loss of uniqueness.

8.1. Convolutional Model

The goal of this convolutional model is to replicate the hierarchical architecture suggested in the i-Theory and additionally to allow for unsupervised training to achieve invariance. To implement the layers of local modules, convolution is used. This will cause reuse of the same local module on all locations of the layer. Using conventional convolution, a highly redundant representation will be generated at the output of this layer due to the large overlap of the inputs for the module. A strided convolution is applied to avoid this redundancy; i.e., for a stride s in all directions of the convolution only every s-th value is calculated.

Each module is represented by filters $W_j \in \mathbb{R}^{U \times V \times D}, j = 1, 2, \ldots, J$, where pairs are coupled like the weight vectors in subspace learning methods (Figure 8.4). The output for an input $X \in \mathbb{R}^{A \times B \times D}$ of a layer is

$$O_i(t) = d_s(Z_i(t)), \tag{8.2}$$

where the amplitude maps $Z_i(t)$ are given by

$$Z_i(t) = \sqrt{\sum_{k=0}^{1} (W_{i2+k} * X(t))^2}. \tag{8.3}$$

In this formal description of the layer output the strided convolution is described by the equivalent of a standard convolution, denoted by $*$, followed by downsampling using the downsampling operator d_s, which takes every s-th value in each direction.

We use an energy model to train this model. A convolutional autoencoder term will encourage a diverse set of filters, while a slowness term ensures an invariant representation. These terms are adapted to this convolutional setting from the slow subspace autoencoder [63], which additionally uses a sparsity term to find local features, whereas this convolutional model finds local features due to the restricted size of the filters. The energies

$$E_{rec} = \sum_t ||X(t) - \sum_j \left(\tilde{W}_j * u_s(d_s(W_j * X(t))) \right) ||_2^2 \tag{8.4}$$

$$E_{slow} = \sum_t ||O(t) - O(t-1)||_1 \tag{8.5}$$

are combined via

$$E = E_{rec} + \alpha E_{slow} \text{ s.t. } ||W_j|| = 1. \tag{8.6}$$

Here, α is scalar for weighting the terms and $\tilde{W}_j$ is obtained from W_j by reversing the order of the elements in the first two dimensions. For the autoencoder, an upsampling operator u_s is introduced that reverses the downsampling by filling in zeros. Note, the autoencoder (8.4) uses no zero padding for $W_j * X(t)$, which decreases the size of the output representation. To recover the original size, zero padding is used for convolution with the upsampled representation. In Equation (8.3) again no zero padding is used.

A multi-layer architecture is then trained layer by layer, bottom up. For training the first layer on sequence images, the images are preprocessed by ZCA filtering (Chapter 2). Using these preprocessed images, filters for the convolutional model were optimized by stochastic gradient descent. After training, the first layer output maps $O_i(t)$ can be computed via (8.2). The next layer is trained on the output of the first layer for unprocessed images. Because these outputs can have many dimensions, the number I of maps is reduced by filtering, and

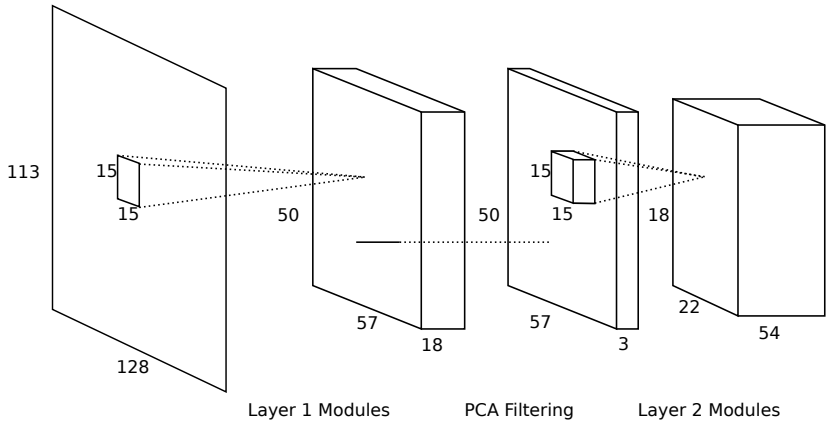


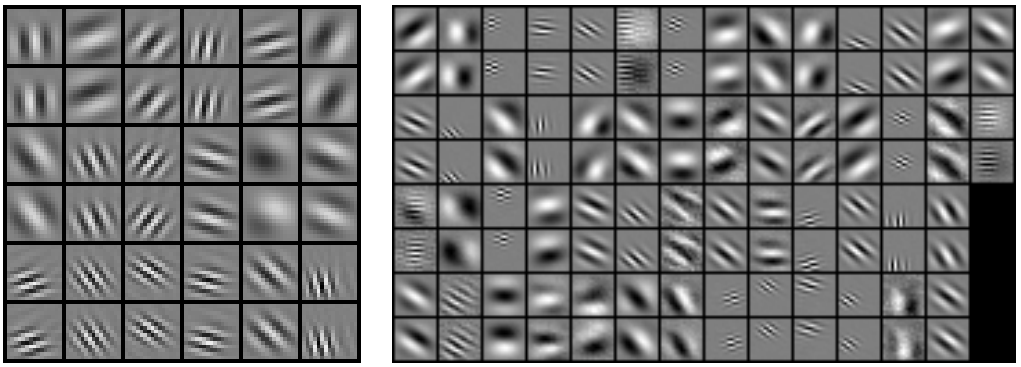
Figure 8.5.: A two layer network using stride two.

therefore also the number of parameters to be learned is decreased. As filters we use the principle components of $1 \times 1 \times I$ patches extracted from the outputs. Note, this filtering retains the spatial structure of the data. Then the data is ZCA filtered and used for optimizing the second layer filters. In my experience, ZCA whitening of the input of the currently trained layer is mandatory to obtain useful filters. Higher layers can be computed analog to the second layer. For computing the outputs of higher layers, only the PCA step is needed, and thus ZCA filtering is omitted.

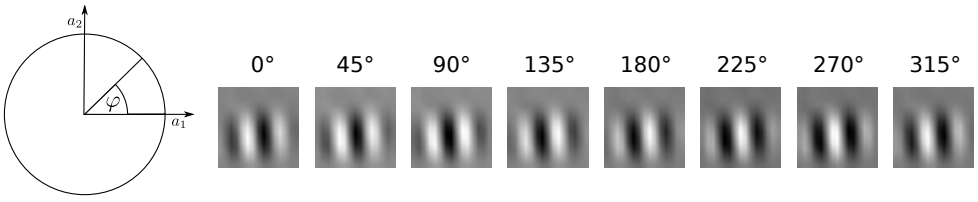
8.2. Experiments

We trained a two layer version of the convolutional model using natural movie sequences from the van Hateren video database [76]. These are gray scale 128×128 pixel movies collected from television containing mostly scenes of animals in the wild. The top 15 pixels are, however, black. Therefore, only 113×128 pixel movies are used for our experiments.

For training the first layer with $\alpha = 50$, the stride was set to 6, the filter size was set to 15×15 pixels and 36 filters shown in Figure 8.6a were trained. Then, using the learned filters, the first layer output was generated using stride 2. To train the second layer, the 18 magnitude maps from the first layer output were reduced via PCA to three maps carrying more than 90% of the variance. For training, the stride of the second layer was set to 6. The filter size was adapted to $15 \times 15 \times 3$ to handle all three maps, and 108 filters were learned with $\alpha = 100$.



(a) (b)



(c)

Figure 8.6.: Layer 1 filters are displayed in (a). In (b) only the top part of the second layer filters is shown. This top part, which is for the first output map, differs from the other parts only in the intensities of the filters. In (c) we see weighted sums $a_1 W_1 + a_2 W_2$ of the first filter pair in the first layer. The vector (a_1, a_2) is from the unit circle. This visualization demonstrates how well shifts can be modeled.

We see the results for the top map in Figure 8.6b.

Clearly, for both layers we obtain Gabor like filters (Figure 8.6). For the second layer the filters are repeated in every map, however, with different intensities. By visual inspection (8.7), the covariance property can be verified. This property is described in the i-Theory (Chapter 6.4.2) as prerequisite for an invariant multilayer architecture. These findings suggest invariance to small shifts in the first layer and an increased invariance to these shifts in the second layer. We tested this translation invariance and also rotation and scale invariance using 100 patches of 64×64 pixels from the van Hateren image database [74]. We measure the change in the output of each layer, as the input undergoes trans-

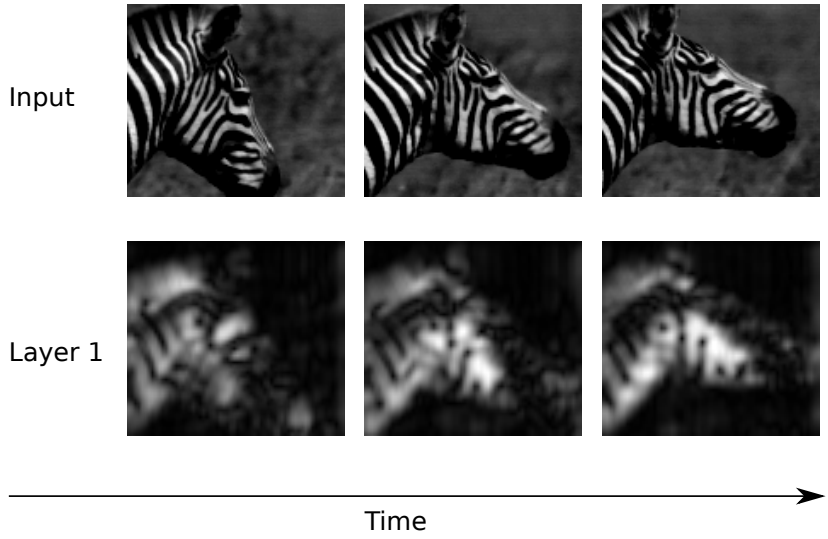


Figure 8.7.: In the top row a stream of input images is shown. Below, an output stream for the first filter pair in layer 1 is shown. Clearly this output stream shows the same transformations and therefore, the covariance property is fulfilled. Note, the layer one output has been generated with stride two. These outputs have been scaled up by factor two for display.

formations. The MSE between the output of the original and the output of the transformed patch is taken and normalized against the largest MSE, assuming the patches are uncorrelated for these transformation parameters. The outputs of both layers were downsampled with stride 3.

The plots in Figure 8.8 validate our assumption of invariance to small shifts. We also see invariance to rotation and scaling, because these transformations can locally be approximated by shifts. And, additionally, the invariance increases from the first to the second layer.

Next, we were interested in the effect of the stride. The strides for both layers were adapted simultaneously. Using the same approach as above, we measured the MSE for different strides on shifted patches. The plots in Figure 8.9 show that the first layer output is not affected, whereas, the second layer is. This is due to the change of the represented area. The larger the stride in the bottom layer the larger the area represented in the second layer.

These findings suggest using large strides. One of the main problems of invariant representations, however, is representing the input uniquely. To test how

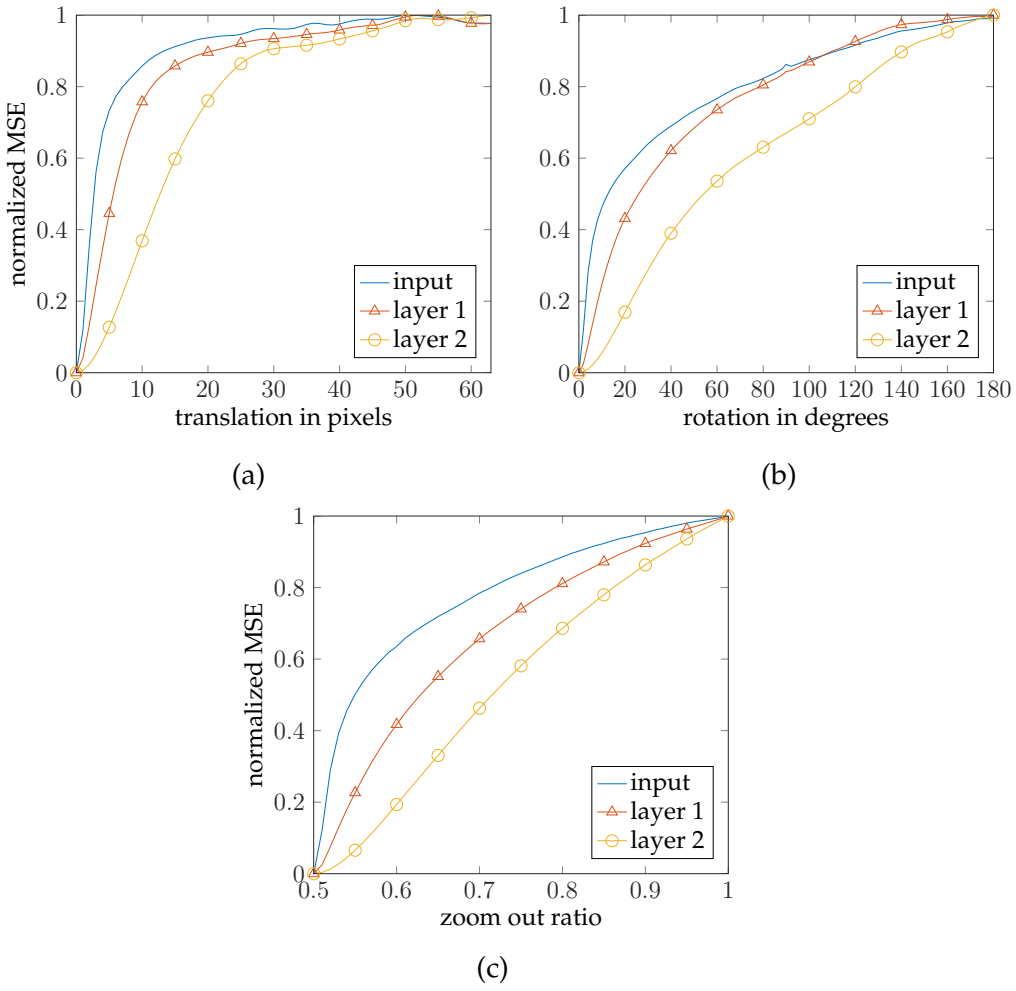
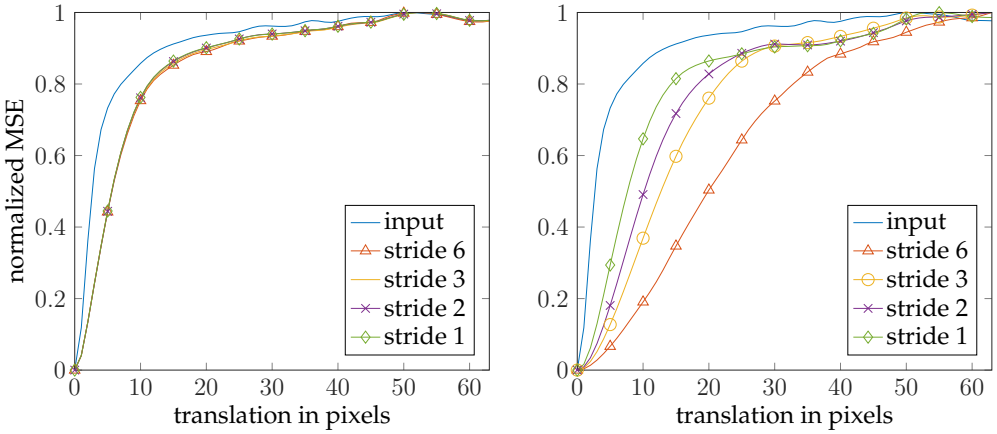


Figure 8.8.: Invariance experiment for varying degrees of shift (a), rotation (b), and scale (c). The normalized MSE in layer 1 and layer 2 is plotted along with results for the unprocessed input patches as reference.



(a) (b)

Figure 8.9.: The normalized MSE in layer 1 (a) and layer 2 (b) depending on the amount of shift is plotted for different strides. As reference also a curve for the input patches is shown.

	Layer 1	Layer 2
Stride 6	1.48	12.88
Stride 3	**1.41**	6.35
Stride 2	1.43	5.01
Stride 1	1.42	**3.28**

Table 8.1.: Results for k-NN classification ($k = 3$) on MNIST. The error rates are given in percent.

well information on fine image structures is retained at each layer, we perform k-NN classification ($k = 3$) on the MNIST [3] dataset. The classification error on the raw images is 3.09%. As we see in Table 8.1, there is a drop in the k-NN classification performance from layer 1 to layer 2, which can be reduced to a certain extend by choosing small stride sizes. This clearly indicates a loss of important information. Interestingly, the first layer error rates are significantly better than on the input images. We think this is due to the small non-affine transformations in MNIST, which may be handled well by the Gabor features.

8.3. Discussion

The architecture presented in this Chapter uses local modules in a hierarchy to achieve invariance to multiple transformations. This structure is motivated by the i-Theory which suggests factorization of the range of invariance to local modules. We trained these modules using a convolutional variant of the slow subspace autoencoder. The resulting representation layer is similar to a layer of i-Theory modules. Indeed, invariance to multiple transformations was measured in the experiments and, as expected, there is an increase in invariance from layer to layer.

However, there is also an information loss which increases from layer to layer and can only be mitigated partially by increasing the redundancy of the layer input. This information loss remains an open problem. Possibly, it can be reduced by learning more object specific representations in the modules. To do this, I think, we need to avoid enforcing orthogonality in the filters and use larger subspaces.

Altogether, the proposed architecture seems to be a good starting point for unsupervised training of deep invariant models. Using only one layer, it is apparently capable of handling some of the non-affine transformations in MNIST, which leads to improved classification results.

Part III.

Crowding

9. Introduction to Crowding in Peripheral Vision

Humans have the impression of high-fidelity perception, although the eye samples only a small fraction of an image mapped on the retina with high resolution. Most of the image is heavily subsampled. Nevertheless, our brain is able to establish this illusion, which, however, is easy to disprove by trying to recognize details of objects seen from the corner of one's eye.

Reduced sampling rates in the periphery of the retina cannot entirely explain the observed recognition deficits. The visual system seems to have deficits when an object, which can be recognized in isolation, is viewed in a cluttered environment. This phenomenon is referred to as crowding. It could be caused by the way images are represented by the visual system. Hence, understanding this phenomenon could help understanding how the best available object recognition system - the human visual system - represents images.

We approach crowding by modeling. To reproduce crowding effects, images are encoded by a simplified model of early vision and decoded from this representation using a sparse image basis as prior. This chapter introduces the crowding phenomenon as well as common theories and models. In the following chapter our model is presented.

9.1. The Crowding Phenomenon

Scenes from the real world are projected by the eye's lens onto the retina. The most central part of the retina is called fovea. Here, the lens maps with high accuracy and densely packed receptors sample the light with a high resolution. With increasing distance to the fovea, which is often referred to as eccentricity, both the quality of the mapped image as well as the sensor density decrease. Vision in this area is called peripheral vision.

We use peripheral vision when we focus on a point and try to recognize a tar-

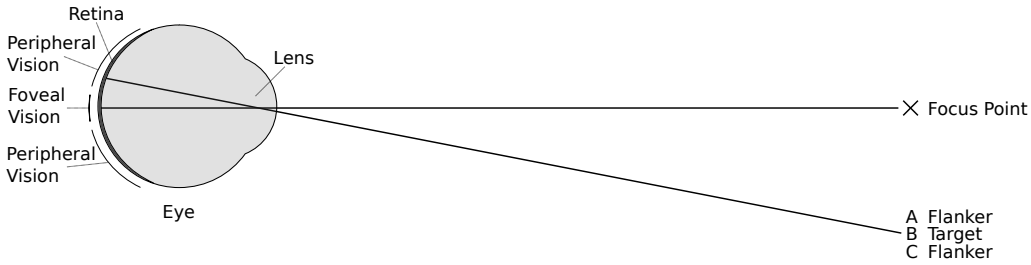


Figure 9.1.: A typical crowding setup is illustrated. On the left side the eye with light falling through the lens on the retina is shown. The retina is colored in dark gray. The central region of the retina performs foveal vision with high sampling rates due to the densely packed photoreceptors. Towards the outer parts of the retina the receptor density decreases. Here, peripheral vision takes place. Note, the extend of these regions indicated on the left is not to scale. The foveal vision region is much smaller. On the right the stimulus is shown. The focus point is projected to the center of the retina, while the targets and flankers are projected to the peripheral part of the retina.

get in the periphery (Figure 9.1). It can be surrounded by other objects referred to as flankers. If in peripheral vision an isolated target can be recognized, but the same target with flankers can not be recognized, we call this phenomenon crowding. An early description is due to Korte [77]. Since this phenomenon has been discovered in 1923, there has been a vast amount of research. Nevertheless, it is still not understood. However, many experiments give cues on the origin of crowding. These results have been reviewed recently [78–80]. Here, we will focus on peripheral crowding, which has strong effects that can clearly be discriminated from other phenomena. There is a debate if crowding is also present in foveal vision [81], since the observed phenomena could also have other causes.

In the following, properties of crowding are reviewed based on [79]. Crowded objects can still be detected, they are visible, but change the appearance and seem to obtain characteristics from flankers [82]. By this effect, identification is impaired. But the impairment is not due to reduced contrast of structures. Rather the structures are perceived as jumbled together and indistinct at seemingly high contrast.

Crowding depends on the spacing between target and flankers. There is a critical spacing, for smaller distances crowding will occur. This critical spa-

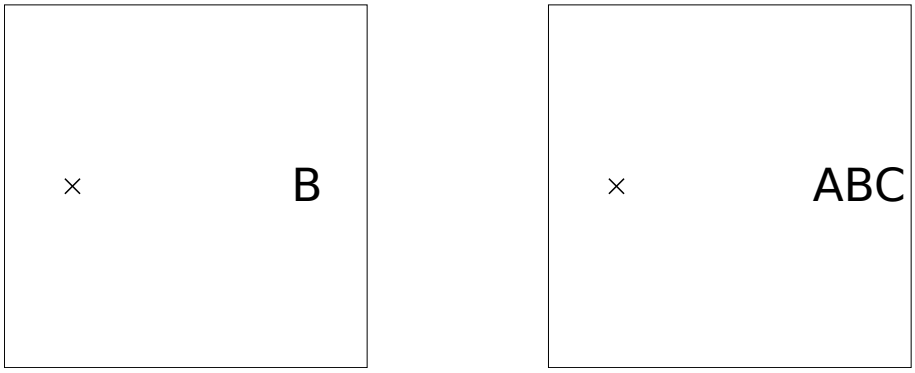


Figure 9.2.: Example stimuli are shown. By focusing on the × the crowding effect can be tested.

cing range increases with eccentricity [83]. The spacing is anisotropic, with a larger influence of horizontally arranged flankers than vertically arranged flankers [84]. Also the effect is not symmetric. Flankers arranged horizontally next to the target have a larger effect if they are placed further away from the fovea compared to flankers with the same distance to the target but closer to the fovea [83].

Even though most demonstrations show spatial crowding only, crowding is not limited to the spatial domain. Temporal flankers, i.e., objects shown in the same location as the stimulus before and after stimulus presentation, can also impair identification.

9.2. Theories of Crowding

A large body of literature has described settings and stimuli, where crowding occurs, as well as targets which increase or decrease the deteriorating effect. A variety of theories try to explain these findings, where the main categories comprise pooling, substitution, and masking models – for a recent review, see Levi [78]. Most theories are descriptive, and only few can directly be implemented as computational models. Both kinds are briefly introduced here.

Pooling models are inspired by classic object recognition models [26, 85]. They assume that local features are measured and integrated over a pooling area, similar to receptive fields, and only the output after pooling is available for recognition. Due to the integration, image information is lost [86]. As noted in [87],

these pooling areas are most likely small in the fovea and become larger towards the periphery. Thus, less neurons are needed to handle the peripheral inputs. While in the fovea a single object is covered by such a pooling area, there are several objects in the peripheral pooling area, which then integrates all features from these objects and causes crowding.

Studies show a high degree of spatial uncertainty in peripheral vision [88, 89]. Substitution models originate from these findings. These models assume that individual features can be detected, but information on the precise feature location is lost, so that features of the target stimulus may be substituted by features of the flankers. This explanation is supported by experiments on the spatial resolution for discrimination tasks. In the fovea, the resolution of individual image features is similar to the resolution of their conjunction. Whereas, in the periphery the resolution of the feature conjunction is much lower than the resolution of individual features [90].

Finally, masking models suggest that strong features of flankers suppress features of the target. This idea extends ordinary masking. In general, masking describes the impairment of the discriminability of a signal by a pattern. This is a well establish phenomenon for signals and masking patterns that overlap. Crowding appears to be just the same, except for the distance between the signal and the masking pattern [91]. Works by [81], however, suggest that masking effects due to inhibition of features are to weak to explain crowding effects.

While all these approaches can capture different aspects of crowding, they do not provide a mathematical framework for simulations. Here, we summarize two pooling models supplying such a framework.

Balas et al. [92] hypothesize that in the visual system statistics of the activations from Gabor-like feature detectors are computed over local pooling regions. These feature detectors are implemented by a complex wavelet transformation. Then statistics, which in some cases are restricted to similar features, summarize the local properties of the input. This model allows to synthesize images with the same statistics and test them on human subjects.

Similarly, Freeman et al. [93] propose a model of Gabor-like feature detectors, which then, in a second stage, are combined nonlinearly and averaged over local regions. This process is only applied to features of neighboring orientation, scale, and position. They can also synthesize images that match the model output.

10. A Compressed Sensing Model of Crowding

10.1. Introduction

Crowding sets a fundamental limit to object perception in peripheral vision as we have seen in the previous chapter. To gain a better understanding of the crowding phenomenon, a computational model is helpful, since it allows for a precise analysis of the causes for degraded perception. Computational models have been proposed before [92, 93]. However, these models make assumptions such as that similar features are combined. Here, we avoid such assumptions.

We hypothesize crowding to be caused by a bottleneck of information transmission from the retina to areas responsible for object recognition in the brain. We present an alternative compressed sensing [94, 95] model for the bottleneck. The work has been published previously [18]. First, a short introduction to the compressed sensing theory is given on which our model is based. Then the new model is presented and analyzed.

10.2. Compressed Sensing

A basic understanding of compressed sensing (CS) [94, 95] is needed to introduce our model. According to the Shannon-Nyquist theorem a signal needs to be sampled at twice the maximum frequency to allow for perfect reconstruction. This is in general true for any signal. However, if prior knowledge about the signal is available only fewer samples may be needed. In the compressed sensing framework [94, 95] random samples are taken. For reconstruction signals that have a sparse representation, such as natural images, are required. This sparsity assumption introduces the prior knowledge, needed to solve a underdetermined systems of equations for reconstructing the input signal.

Mathematically, measurements are acquired by correlating a signal $\mathbf{x}$ with the

sensing patterns θ_k:

$$y_k = \langle \mathbf{x}, \theta_k \rangle, \; k = 1, 2, \ldots, m. \tag{10.1}$$

Usually, a certain degree of randomness is involved in generating the sensing patterns θ_k [94, 96]. This randomness ensures global measurements of the signal as well as maximum differences between the sensing patterns. The sampling of the signal is then done with several sensing patterns, which can be expressed by a measurement matrix Θ with m sensing waveforms θ_k as rows:

$$\mathbf{y} = \Theta \mathbf{x}. \tag{10.2}$$

Note that the problem of solving this equation for the original signal $\mathbf{x}$ given the measurement vector $\mathbf{y}$ is underdetermined, since the dimension of $\mathbf{y}$ is smaller than that of $\mathbf{x}$. Hence, additional information is needed to find a unique solution and thereby recover $\mathbf{x}$. This information is introduced by using a sparse representation $\mathbf{a}$ of $\mathbf{x}$, which is obtained by using a sparse basis Ψ

$$\mathbf{y} = \Theta \Psi \mathbf{a}. \tag{10.3}$$

Sparsity in the coefficients $\mathbf{a}$ is then used as a constraint to find a unique solution to the underdetermined system of equations

$$\arg\min_{\mathbf{a}} ||\mathbf{y} - \Theta\Psi\mathbf{a}||^2 \text{ s.t. } ||\mathbf{a}||_0 < s, \tag{10.4}$$

assuming the signal is s-sparse, i.e. $\mathbf{a}$ has s non-zero elements, in the basis Ψ. This optimization problem can approximately be solved by using the Compressive Sampling Matching Pursuit (CoSaMP) [97] method.

Compressed sensing has been introduced with uniform random sampling, however, particularly good results were obtain with Poisson-disk sampling [98]. The sampling pattern is generated by randomly adding sampling points, while enforcing a minimum distance. Because retinal photoreceptor mosaics are distributed similarly on the retina [99], Poisson-disk sampling is used in our model for biologically plausible sampling.

10.3. Model of Visual Processing

Our model of visual processing considers the signal prior to processing in the visual cortex. Many details of retinal processing are omitted and, therefore, it does not account for all physiological data. We model two stages of convergence, i.e., downsampling.

The first convergence stage starts with the image sampled by the photoreceptors. In the retina the image is blurred and subsampled. This is known to be highly eccentricity-dependent, and varies up to 1000-fold in the human retina [100, 101]. In the fovea the photoreceptors are tightly packed achieving high sampling rates. Besides tight packing, there is also a low level of neural integration[1]. In the periphery, however, there is a low density of cones and, additionally, neural integration due to lateral connections in the retina is high [101]. This stage is modeled by a Gaussian blur and Poisson-disc sampling.

In the second convergence stage, the retinal output is obtained. This output is downsampled, since there are less neurons in this layer than input signals. These neurons are randomly connected to several input neurons. Each of these input neurons can connect to several output neurons. Note that this randomized projection of the input by the output neurons corresponds to what the sensing matrix does in compressed sensing. Therefore, the signal can be transmitted with reduced bandwidth. Again, the sampling rate is eccentricity dependent with larger downsampling in the periphery. The downsampling in this stage is beneficial for the image transmission through the optic nerve from the retina to the visual cortex, since the optic nerve acts as a bandwidth-limited bottleneck. The projections in this stage make this model a pooling model.

For analysis, the input signal is reconstructed from the output of the second stage. This reconstruction is not needed for the brain to access the encoded information. The reason for reconstructing the input is to obtain an intuitive measure of how much information was lost during transmission by visualizing the results. Mathematically, an underdetermined system of equations is solved. To find a unique solution the sparseness of natural signals is used as prior. This is analog to compressed sensing, where a randomly sampled signal is also reconstructed using this prior. Therefore, standard compressed sensing tools can be applied.

10.4. Mathematical Model

To perform experiments we need a mathematical description of our crowding model. An image $\mathbf{x}$ is the input to this model. The first convergence stage then blurrs and sub-samples $\mathbf{x}$, with the blur described by the matrix C being Gaussian. By blurring the image $\mathbf{x}$ we obtain $\mathbf{x}_{\mathrm{blur}}$, which then is sub-sampled via the

[1] For every cone the signal is processed by at least one ganglion cell.

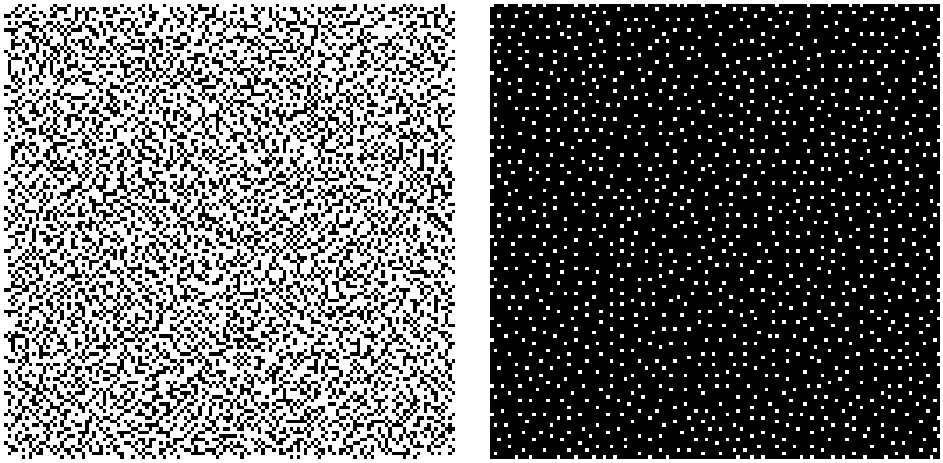


Figure 10.1.: A foveal (left) and a peripheral (right) distribution of the sampling scheme used in the first stage. The Poisson disk radius is 1 pixel for foveal sampling and 3 pixels for peripheral sampling.

matrix $\Phi \in \mathbb{R}^{n \times m}$ ($n < m$). This matrix implements Posson-disk sampling. The resulting image is $\mathbf{x}_{\text{sampled}}$. Combined, the first convergence stage is described by:

$$\mathbf{x}_{\text{sampled}} = \Phi \mathbf{x}_{\text{blur}} = \Phi C \mathbf{x}. \tag{10.5}$$

To generate the Poisson-disk distributed sampling patterns shown in Figure 10.1 we used Bridson's algorithm [102].

In the retina the convergence due to blurring and subsampling increases smoothly from the fovea to the periphery. For the purpose of our experiments, we consider only block-wise constant convergence rates and vary them to simulate foveal and peripheral vision.

Then the samples defined in Equation 10.5 are used as input to a network of neurons. The output neurons randomly connect with multiple inputs. Each connection has random weights for the inputs. This network can be described by a connection matrix Θ, where each output neuron is represented by a row. For every connection to a input neuron there is a non-zero entry in the corresponding column. The value of the entry represents the weight of the connection. Using Θ the activation of the output neurons is computed:

$$\mathbf{x}_{\text{output}} = \Theta \mathbf{x}_{\text{sampled}}. \tag{10.6}$$

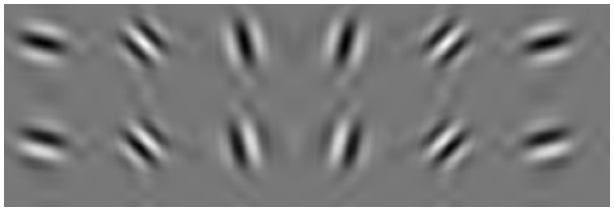


Figure 10.2.: Basis elements from the Dual Tree Complex Wavelet Transform.

In total, our model can be described by the linear transformation

$$\mathbf{x}_{\text{output}} = \Theta\Phi C\mathbf{x}, \tag{10.7}$$

where $\mathbf{x}_{\text{output}}$ contains the information available to later stages of the visual system.

10.5. Input Reconstruction

Inferring the information still available in the output neuron is essential to test the model. We do this by reconstructing an image from the output $\mathbf{x}_{\text{output}}$. Of course, due to convergence, there is a dimensionality reduction in the data, which makes reconstruction an underdetermined problem. However, since the reconstruction problem in compressed sensing (Equation 10.4) is similar to this problem, we can use methods from the compressed sensing framework. Nevertheless, the amount of information that can be encoded by the output neurons is limited. If there is more information in the input signal, some information will be lost. To reconstruct the image, it is represented in a sparse basis Ψ:

$$\mathbf{x}_{\text{output}} = \Theta\Phi\text{C}\Psi\mathbf{a}, \tag{10.8}$$

where $\mathbf{x} = \Psi\mathbf{a}$. As a sparse basis, the Dual Tree Complex Wavelet Transform (DT-CWT) was selected [103, 104]. It is a biologically plausible choice, because the basis elements are similar to the response properties of the simple cells in the primary visual cortex [104]. The elements of this basis, illustrated in Figure 10.2, are oriented Gabor-like functions. Using this representation the CoSaMP algorithm can be applied to approximately solve

$$\arg\min_{\mathbf{a}} ||\mathbf{x}_{\text{output}} - \Theta\Phi\text{C}\Psi\mathbf{a}||^2 \text{ s.t. } ||\mathbf{a}||_0 < s. \tag{10.9}$$

Then, the image $\hat{x}$ can be reconstructed

$$\hat{\mathbf{x}} = \Psi \mathbf{a} \tag{10.10}$$

to visualize the information retained in x_{output} after the two convergence stages of the model.

10.6. Experiments

We used a letter recognition task for testing our model. To simulate the crowding phenomenon qualitatively, 10 images, each with a letter in isolation and a letter flanked by two random letters, were generated. The two letters were placed to the left and to the right of the central letter. The images had a size of 128 by 128 pixels and about 30 pixels high bright letters were placed in the center on a dark background. Foveal and peripheral representations of these letters were obtained using our model. For the fovea only a small Gaussian blur of variance 2 was used and the radius for the Poisson disk sampling was just 1 pixel, which resulted in roughly 10800 samples for the 16384 pixels of the image. In the periphery, the images were blurred with a Gaussian of variance 4 pixels, and the Poisson disk had a radius of 3 pixels, which resulted in about 1750 samples. Convergence in the second model stage, i.e., the number of output neurons, was varied from 1500 to 50, resulting in overall convergence rates of 11 to 328.

To estimate the amount of information represented by the final samples, the input images were reconstructed as described above. The DT-CWT was used with 5 levels of evaluation depth. In Figure 10.3 example reconstructions are shown. One can see that the foveal output neurons have sufficient information encoded to allow for a good reconstruction for both the un-flanked and the flanked letter image. The overall quality of the reconstruction from the peripheral output neurons is of course worse, but the letter 'X' is still recognizable if it is not flanked by other letters. Note that the reconstruction is based on only 1.5% of the input pixels in case of a convergence rate of 66 in stage 2 (250 output neurons). If the letter is flanked by other letters the reconstructed images exhibit typical crowding artifacts in the sense that the letter can be located, but its features are jumbled and recognition is impossible. However, if the convergence rate in stage 2 is decreased to 27 (600 output neurons), also the flanked letter 'X' becomes recognizable.

To compare the input and output images quantitatively, the mean squared

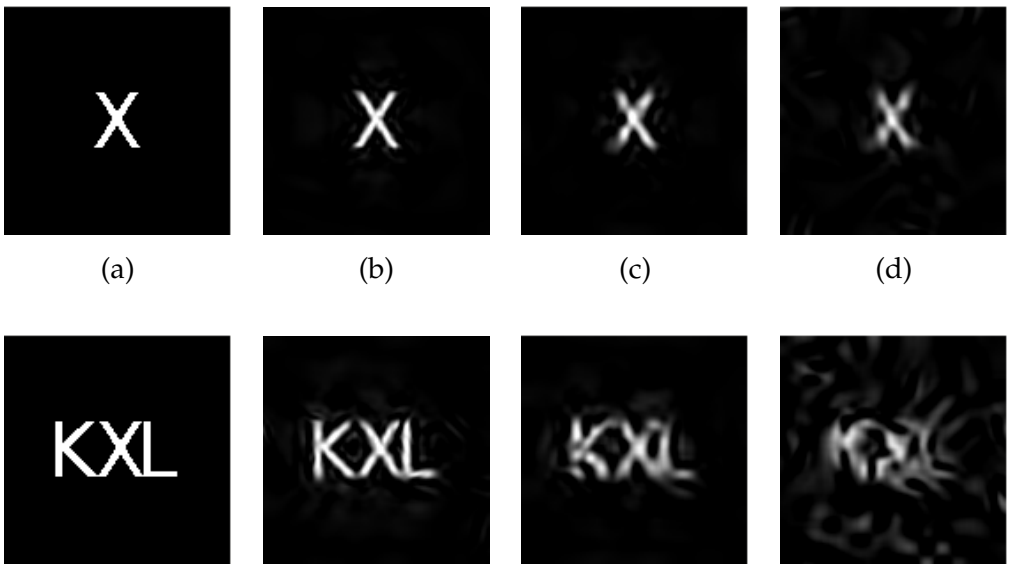


Figure 10.3.: The letter X is shown in isolation (a) and flanked by other letters (e). All other images show reconstructed versions of these input images. The letters can still be easily recognized after reconstruction from a high-bandwidth foveal representation (b, f). These two images correspond to the two arrows in the left plot of Figure 10.4; the convergence rate is 13, i.e. the second stage has 1250 output neurons. The images on the right (c, d, g, h) are reconstructed from the peripheral first stage (see text) with two different convergence rates in the second stage. The four images correspond to the four arrows in the right plot of Figure 10.4. The two convergence rates are 27 and 66, i.e., the second stage had 600 and 250 output neurons, respectively. These four images illustrate the available information at convergence rates where the flanked letter turns unrecognizable while the un-flanked letter can still be read.

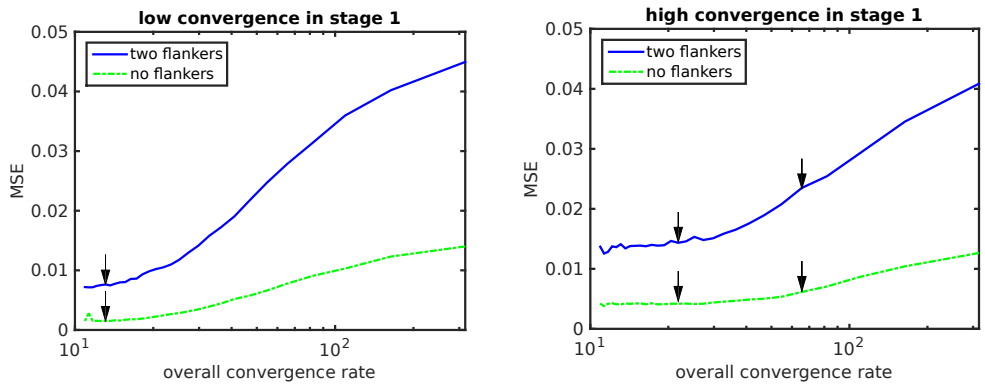


Figure 10.4.: The graphs show the reconstruction errors for a foveal (left) and a peripheral (right) sample of the first stage and as a function of the total convergence ratio (number of input samples in x divided by the number of output samples in x_{output}), i.e., as a function of the number of output neurons in the second stage. The parameters of the first stage are those given in the text. The six arrows correspond to the six reconstructed images shown in Figure 10.3 and the values of the parameters are given in the caption of Figure 10.3.

error (MSE) was used as a difference measure. The MSE was computed only within a bounding box around the central letter to avoid measuring errors due to the flanking letters. We tested the reconstruction quality depending on the number of output neurons, and for the above selected values of the Gaussian blur and Poisson-disc sampling. The errors were measured for all images and averaged over 10 images. Figure 10.4 shows MSE as a function of the overall convergence rate. The parameters of the first convergence stage have been kept fixed at the same two foveal and the peripheral values that were used to create the images in Figure 10.3. Only the number of output neurons in the second convergence stage has been varied. First note that the errors increase with increasing convergence rate and that the two error curves (single letter vs. flanked letter) diverge. This shows us that the crowding effect in our model is due to a limited encoding capacity of the output neurons. However, the first convergence stage does also play a role since the results for the foveal and the peripheral simulation differ. It seems particularly interesting that the critical convergence rate (where the errors start rising) differ for the four curves. Especially in the right plot (peripheral first stage), it becomes evident that a convergence rate optimized for single letters is not optimal for a letter with flankers. In the left plot

(foveal first stage) the difference between the two optima is small. Note, also that the reconstruction errors are in all cases higher for the flanked letter. For both settings the errors of flanked and un-flanked images do not reach the same level. The difference between the errors becomes smaller for the foveal model. Probably the residual errors when many output neurons are used are not due to a limited coding capacity, but due to the under-sampled deconvolution problem. This deconvolution problem sets a limit to the best possible reconstruction. With a full sampling of a sharp image by the receptors and the use of one fourth of the number of pixels as output neurons, a perfect reconstruction is possible for any number of flankers, since that is a standard compressed sensing problem.

10.7. Discussion

We have presented a simple model of low-level vision in the retina with two convergence stages. It makes use of the fact that natural images exhibit redundancies by compressing the image in the convergence stages. This compression is achieved using random sampling similar to compressed sensing of a blurred signal. Thus, the bandwidth needed for transmission through the optic nerve is reduced. The convergence rates vary for both stages over the retina. In the fovea the compression rate is low while it increases with eccentricity in the periphery. This, of course, implies a varying degree of information loss. To explore the consequence of compression in the retinal coding we used compressed sensing method to restore the input signals. For the peripheral part of the model we found clear crowding effects for flanked letters. By visual inspection we can verify the jumbled together appearance, which still has high contrast in many regions. Detection of the letters remains viable, while recognition is impossible. Other properties such as eccentricity dependence and isometry can be explicitly modeled via the shape of the field covered by the second stage neurons.

We analyzed our model for two parameter sets in the first convergence stage. One yielding low convergence rates as found in the fovea, the other resembling the high convergence rates of the periphery. The second stage convergence rate was varied to measure its influence. For both first stage convergence rates, the reconstruction error increases with the convergence rate of the second stage. The increase also depends on weather there are flankers present. Flankers cause a steeper increase of the error. Therefore, when the second convergence stage is optimized for a good representation of the image at minimum bandwidth,

the optima are different for the un-flanked and the flanked object. Un-flanked objects need less bandwidth. These results suggest that evolution has optimized foveal vision to cope with flanked objects to allow for good recognition rates, while for peripheral vision recognizing un-flanked objects is sufficient.

Compared to alternative pooling models, we believe that our model is simple. For example we avoid assumptions such as combinations of similar features, as in the models of Balas et al. [92] and Freeman et al. [93]. Specific to our model is that it models the retina. Although we modeled such an early part of visual processing, we believe later stages may cause crowding effects as well.

In conclusion, we have shown that the compressed sensing framework has the potential to provide new insights regarding visual coding in general and the crowding phenomenon in particular.

11. Summary and Outlook

During the last few years research into classifiers and their enhancement by representations has seen a huge rise. This rise is due to the ever increasing demand for automatical analysis of data, which is collected in a diverse set of domains. Classification already works in many domains. In this thesis, we explored three different problems in representation learning and its application – linear representation learning to bypass the limitations of k-NN, invariant representations for disentangling visual data, and understanding visual phenomena caused by the representations in the human visual system.

Linear representation learning methods can be used to improve the error rate of a classifier, such as k-NN. For k-NN it depends on the scaling of the input data. Of course, this shortcoming can be circumvented by a linear change of the representation. There are several methods already available. However, they tend to depend on the initial scaling of the data. We developed two new methods. Both avoid a dependence on the initial scaling of the data by use of a global optimization term. They globally minimize the distance of equally labeled data, while establishing a minimum distance between differently labeled data.

The first method, MDM, focuses on rescaling the data dimensions. It is also capable of reducing the weight of irrelevant dimensions to zero, thereby performing dimensionality reduction. The global criterion used, does not impose a data distribution as prior, thus avoiding restrictions to a limited class of data sets. This is achieved by solving a linear program, which guarantees to find the optimal solution for the criterion. GML, the second method, uses gradient descent for the optimization. The global energy term used in this method introduces a Gaussian prior. Contrary to MDM it not only rescales the data dimension, but applies a full linear transformation. This allows to reduce statistical redundancies and, thereby, improves the results for several data sets compared to feature weighting methods such as MDM. However, we showed that for many data sets, rescaling, as imposed by the feature weighting methods, is sufficient. In the experiments MDM and GML showed little sensitivity to the initial scaling

of the data sets, making them good choices. In case the data was preprocessed, MDM and GML still find competitive representation with good robustness in the sense that they never performed much worse then alternative methods.

MDM and GML both suffer from a quadratic increase in the number of data pairs, which makes their application to large data sets inefficient. This should be addressed in further development of these methods. At least for MDM this seems possible, because only a small subset of the pairs is relevant to the MDM results. Therefore, it seems reasonable to investigate the change of relevant pairs to decrease the computational effort and memory usage. Moreover, the poor performance of all feature weighting methods in the micro array data, which exhibits very high dimensionality and simultaneously only a limited amount of training samples, demand for better methods. Of course, this setting is in general problematic. But, possibly some improvements can be made by better dimensionality reduction, which is not yet guaranteed to be optimal.

Furthermore, disentangling of visual data was investigated. Due to the 2D projections of scenes in the world, transformations of objects in these scenes cause complicated changes to the pixel representation. Therefore, discriminating objects is a challenging and in general not solved task. I first showed the close relationship between known architectures and the i-Theory. Then, a new architecture based on the i-Theory was introduced, and evaluated. It showed promising results for shifts in an artificial dataset, but for real world data we observed difficulties in the training.

Second, a convolutional network using slow subspace autoencoders was introduced. This network can be trained layer by layer on unlabeled image sequences. Low level layers only provide invariance in a small range. With each layer the invariance range increases. However, simultaneously the information loss increases. This information loss can only be mitigated to a certain degree.

The problem of finding discriminative and transformation invariant representations remains unsolved. Deep neural networks achieve invariance to a certain degree and a remarkable classification accuracy, when supervised training is applied. Yet, they can easily be fooled. Changes barely visible to the human observer cause misclassification [105]. This hints towards discontinuities in these models. It is questionable weather this can be solved by supervised training of a classifier. Hence, our methods directly optimize invariance. The difficulties in training the first model are an interesting problem for further investigation. More promising are, I think, slowness approaches as used in the second method.

In the experiments Gabor-like filters have emerged in training. These filters are also found in the first layer of a well working deep convolutional network after supervised training for classification of images [47]. The architecture of deep convolutional networks is capable of invariant representations of objects. This is not only suggested by the i-Theory, but also by the experiments using Deep Convolutional Inverse Graphics Networks [106], which, however, need well labeled data to achieve invariance in an artificial setting. I suggest to build on our slowness based approach, and increase the size of the subspaces in order to achieve better discriminative properties. Very helpful might be to decorrelate the pooling layer outputs [107], instead of orthogonalizing the filter output. This might allow to pass more information through the pooling layer in such an unsupervised learning approach.

Finally, we applied concepts from compressed sensing to crowding, reproducing this phenomenon of peripheral vision. Our model makes only few assumptions compared to other models. It is placing the phenomenon in early vision prior to passing information through the optic nerve, which is assumed to be an information bottleneck. Some properties of crowding found in psychological studies such as eccentricity dependence and isometry could be explicitly modeled. Nevertheless, we did not model these properties in our model. Also an extension to multi-layer models would be interesting, since this phenomenon could be introduced at multiple stages of representation in the visual system. Although, we were able to reproduce the crowding effect in our model, further phenomena and physiological data are needed to provide sufficient restrictions to draw conclusions regarding the structure of representations in the human visual system.

In conclusion, this endeavor to learn and understand representations, covers a small selection of possible domains. The experiments on the presented learning algorithms show that classifiers benefit from this processing of the data. Especially in domains as complicated as visual object recognition, representation learning seems essential to reach human performance. Moreover, modeling of visual phenomena is an opportunity to better understand representations in the human visual system. This understanding again might help improve representation learning.

Bibliography

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[2] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 71, no. 1, pp. 38–53, 1911.

[5] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[7] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[8] P. Comon, "Independent component analysis, a new concept?," *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[9] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[10] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[11] J. Hocke and T. Martinetz, "Feature Weighting by Maximum Distance Minimization," in *Proceedings of the 23rd International Conference on Artificial Neural Networks*, vol. 8131, pp. 420–425, 2013.

[12] J. Hocke and T. Martinetz, "Application of Maximum Distance Minimization to Gene Expression Data," in *Workshop New Challenges in Neural Computation*, pp. 6–7, 2013.

[13] J. Hocke and T. Martinetz, "Maximum Distance Minimization for Feature Weighting," *Pattern Recognition Letters*, vol. 52, pp. 48–52, 2015.

[14] J. Hocke and T. Martinetz, "Global Metric Learning by Gradient Descent," in *Proceedings of the 24th International Conference on Artificial Neural Networks*, vol. 8681, pp. 129–135, 2014.

[15] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, "Unsupervised learning of invariant representations in hierarchical architectures," *CoRR*, vol. abs/1311.4158, 2013.

[16] J. Hocke and T. Martinetz, "Learning Transformation Invariance for Object Recognition," in *Workshop New Challenges in Neural Computation*, pp. 20–25, 2014.

[17] J. Hocke and T. Martinetz, "Learning Transformation Invariance from Global to Local," in *Workshop New Challenges in Neural Computation 2015* (B. Hammer and T. Villmann, eds.), vol. 03/2015 of *Machine Learning Reports*, pp. 16–24, 2015.

[18] J. Hocke, M. Dorr, and E. Barth, "A compressed sensing model of peripheral vision," in *Human Vision and Electronic Imaging XVII* (B. E. Rogowitz, T. N. Pappas, and H. de Ridder, eds.), vol. 8291, pp. 82910Z–82910Z–7, Proceedings of SPIE, 2012.

[19] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[20] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 4th ed., 2008.

[21] F. Rosenblatt, "The perceptron—a perceiving and recognizing automaton," Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[22] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *Journal of Machine Learning Research*, vol. 10, pp. 66–71, 2009.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.

[24] F. Anselmi, L. Rosasco, and T. Poggio, "On invariance and selectivity in representation learning," *arXiv preprint arXiv:1503.05938*, 2015.

[25] J. Hocke, K. Labusch, E. Barth, and T. Martinetz, "Sparse Coding and Selected Applications," *KI - Künstliche Intelligenz*, vol. 26, no. 4, pp. 349–355, 2012.

[26] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?," *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[27] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.

[28] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, pp. 421–436, Springer, 2012.

[29] T. Schaul, S. Zhang, and Y. LeCun, "No More Pesky Learning Rates," in *Proceedings of the 30th International Conference on Machine learning*, vol. 28, pp. 343–351, 2013.

[30] J. Sohl-Dickstein, B. Poole, and S. Ganguli, "Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, p. 604–612, 2014.

[31] A. J. Bell and T. J. Sejnowski, "Edges are the" independent components" of natural scenes," in *Advances in Neural Information Processing 9*, pp. 831–837, 1996.

[32] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[33] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems 17*, pp. 513–520, 2004.

[34] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Networks*, vol. 15, no. 8-9, pp. 1059–1068, 2002.

[35] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems*, vol. 15, pp. 505–512, 2002.

[36] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the 9th International Workshop on Machine Learning*, pp. 249–256, 1992.

[37] R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin based feature selection - theory and algorithms," in *Proceedings of the 21th International Conference on Machine Learning*, pp. 43–50, 2004.

[38] Y. Sun and J. Li, "Iterative relief for feature weighting," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 913–920, 2006.

[39] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems 19*, 2006.

[40] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.

[41] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[42] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[43] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[44] T. Schaul and Y. LeCun, "Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients," *arXiv preprint arXiv:1301.3764*, 2013.

[45] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 1735–1742, 2006.

[46] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems 25*, pp. 1097–1105, 2012.

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, pp. 1–42, April 2015.

[49] J. Wood, "Invariant pattern recognition: a review," *Pattern Recognition*, vol. 29, no. 1, pp. 1–17, 1996.

[50] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[52] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[53] G. E. Hinton, "Connectionist learning procedures," *Artificial intelligence*, vol. 40, no. 1, pp. 185–234, 1989.

[54] P. Földiák, "Learning invariance from transformation sequences," *Neural Computation*, vol. 3, no. 2, pp. 194–200, 1991.

[55] P. Berkes and L. Wiskott, "Slow feature analysis yields a rich repertoire of complex cell properties," *Journal of Vision*, vol. 5, no. 6, p. 9, 2005.

[56] B. Lau, G. B. Stanley, and Y. Dan, "Computational subunits of visual cortical neurons revealed by artificial neural networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 13, pp. 8974–8979, 2002.

[57] T. Kohonen, "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map," *Biological Cybernetics*, vol. 75, no. 4, pp. 281–291, 1996.

[58] B. A. Olshausen, C. F. Cadieu, and D. K. Warland, "Learning real and complex overcomplete representations from the statistics of natural images," vol. 7446, pp. 74460S–74460S–11, 2009.

[59] C. Zetzsche, G. Krieger, and B. Wegmann, "The atoms of vision: Cartesian or polar?," *Journal of the Optical Society of America A*, vol. 16, no. 7, pp. 1554–1565, 1999.

[60] A. Hyvärinen and P. Hoyer, "Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces," *Neural Computation*, vol. 12, no. 7, pp. 1705–1720, 2000.

[61] C. Kayser, W. Einhäuser, O. Dümmer, P. König, and K. Körding, "Extracting slow subspaces from natural videos leads to complex cells," in *Artificial Neural Networks—ICANN 2001*, pp. 1075–1080, Springer, 2001.

[62] W. Y. Zou, A. Y. Ng, and K. Yu, "Unsupervised learning of visual invariance with temporal coherence," in *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[63] W. Zou, S. Zhu, K. Yu, and A. Y. Ng, "Deep learning of invariant features via simulated fixations in video," in *Advances in Neural Information Processing Systems*, pp. 3212–3220, 2012.

[64] C. F. Cadieu and B. A. Olshausen, "Learning intermediate-level representations of form and motion from natural movies," *Neural Computation*, vol. 24, no. 4, pp. 827–866, 2012.

[65] T. Cohen and M. Welling, "Learning the irreducible representations of commutative lie groups," in *Proceedings of The 31st International Conference on Machine Learning*, pp. 1755–1763, 2014.

[66] R. Memisevic and G. Hinton, "Unsupervised learning of image transformations," in *Computer Vision and Pattern Recognition (CVPR). IEEE Conference on*, pp. 1–8, 2007.

[67] R. Memisevic, "Learning to relate images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1829–1846, 2013.

[68] R. Memisevic, "On multi-view feature learning," pp. 1–8, 2012.

[69] R. Memisevic, "Gradient-based learning of higher-order image features," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1591–1598, 2011.

[70] H. Cramér and H. Wold, "Some theorems on distribution functions," *Journal of the London Mathematical Society*, vol. s1-11, no. 4, pp. 290–294, 1936.

[71] F. Anselmi and T. A. Poggio, "Representation learning in sensory cortex: a theory," *CBMM Memo*, vol. 26, 2014.

[72] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[73] Q. Liao, J. Z. Leibo, Y. Mroueh, and T. Poggio, "Can a biologically-plausible hierarchy effectively replace face detection, alignment, and recognition pipelines?," *CBMM Memo No 3 arXiv preprint arXiv:1311.4082*, 2013.

[74] J. H. van Hateren and A. van der Schaaf, "Independent component filters of natural images compared with simple cells in primary visual cortex," *Proceedings: Biological Sciences*, vol. 265, pp. 359–366, Mar 1998.

[75] J.-P. Lies, R. M. Häfner, and M. Bethge, "Slowness and sparseness have diverging effects on complex cell learning," *PLoS computational biology*, vol. 10, no. 3, p. e1003468, 2014.

[76] J. H. van Hateren and D. L. Ruderman, "Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 265, no. 1412, pp. 2315–2320, 1998.

[77] W. Korte, "Über die Gestaltauffassung im indirekten Sehen," *Zeitschrift für Psychologie*, vol. 93, pp. 17–82, 1923.

[78] D. Levi, "Crowding–an essential bottleneck for object recognition: A mini-review," *Vision Research*, vol. 48, no. 5, pp. 635–654, 2008.

[79] D. Whitney and D. Levi, "Visual crowding: a fundamental limit on conscious perception and object recognition," *Trends in Cognitive Sciences*, vol. 15, no. 4, pp. 160–168, 2011.

[80] M. H. Herzog, B. Sayim, V. Chicherov, and M. Manassi, "Crowding, grouping, and object recognition: A matter of appearance," *Journal of Vision*, vol. 15, no. 6, pp. 1–18, 2015.

[81] D. M. Levi, S. A. Klein, and S. Hariharan, "Suppressive and facilitatory spatial interactions in foveal vision: Foveal crowding is simple contrast masking," *Journal of Vision*, vol. 2, no. 2, p. 2, 2002.

[82] J. A. Greenwood, P. J. Bex, and S. C. Dakin, "Crowding changes appearance," *Current Biology*, vol. 20, no. 6, pp. 496–501, 2010.

[83] H. Bouma, "Interaction effects in parafoveal letter recognition," *Nature*, vol. 226, pp. 177–178, 1970.

[84] C. Feng, Y. Jiang, and S. He, "Horizontal and vertical asymmetry in visual spatial crowding effects," *Journal of Vision*, vol. 7, no. 13, pp. 1–10, 2007.

[85] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[86] L. Parkes, J. Lund, A. Angelucci, J. A. Solomon, and M. Morgan, "Compulsory averaging of crowded orientation signals in human vision," *Nature Neuroscience*, vol. 4, no. 7, pp. 739–744, 2001.

[87] D. G. Pelli, M. Palomares, and N. J. Majaj, "Crowding is unlike ordinary masking: Distinguishing feature integration from detection," *Journal of vision*, vol. 4, no. 12, p. 12, 2004.

[88] C. L. Krumhansl and E. A. Thomas, "Effect of level of confusability on reporting letters from briefly presented visual displays," *Perception & Psychophysics*, vol. 21, no. 3, pp. 269–279, 1977.

[89] D. G. Pelli, "Uncertainty explains many aspects of visual contrast detection and discrimination," *JOSA A*, vol. 2, no. 9, pp. 1508–1532, 1985.

[90] P. Neri and D. M. Levi, "Spatial resolution for feature binding is impaired in peripheral and amblyopic vision," *Journal of Neurophysiology*, vol. 96, no. 1, pp. 142–153, 2006.

[91] S. T. Chung, D. M. Levi, and G. E. Legge, "Spatial-frequency and contrast properties of crowding," *Vision Research*, vol. 41, no. 14, pp. 1833–1850, 2001.

[92] B. Balas, L. Nakano, and R. Rosenholtz, "A summary-statistic representation in peripheral vision explains visual crowding," *Journal of Vision*, vol. 9, no. 12, p. 13, 2009.

[93] J. Freeman and E. P. Simoncelli, "Metamers of the ventral stream," *Nature Neuroscience*, vol. 14, no. 9, pp. 1195–1201, 2011.

[94] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[95] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[96] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, pp. 21–30, March 2008.

[97] D. Needell and J. A. Tropp, "CoSaMP: iterative signal recovery from incomplete and inaccurate samples," *Communications of the ACM*, vol. 53, no. 12, pp. 93–100, 2010.

[98] M. Lustig, M. T. Alley, S. Vasanawala, D. L. Donoho, and J. M. Pauly, "Autocalibrating parallel imaging compressed sensing," in *Proceedings of the 17th Annual Meeting of ISMRM*, p. 379, 2009.

[99] J. Yellott, "Spectral consequences of photoreceptor sampling in the rhesus retina," *Science*, vol. 221, pp. 382–385, 1983.

[100] C. A. Curcio, R. R. Sloan, O. Packer, A. E. Hendrickson, and R. E. Kalina, "Distribution of cones in human and monkey retina: individual variability and radial asymmetry," *Science*, vol. 236, pp. 579–582, 1987.

[101] C. Curcio and K. Allen, "Topography of ganglion cells in human retina," *The Journal of Comparative Neurology*, vol. 300, no. 1, pp. 5–25, 1990.

[102] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," in *ACM SIGGRAPH 2007 sketches*, p. 22, ACM, 2007.

[103] N. Kingsbury, "The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement," in *EUSIPCO: European Signal Processing Conference*, pp. 319–322, 1998.

[104] I. Selesnick, R. Baraniuk, and N. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[105] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[106] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum, "Deep convolutional inverse graphics network," *arXiv preprint arXiv:1503.03167*, 2015.

[107] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," *arXiv preprint arXiv:1511.06068*, 2015.

Danksagung

In meiner Zeit am INB habe ich viele Erfahrungen gesammelt und vieles gelernt; fachlich und persönlich. Einigen Menschen bin ich zu besonderem Dank verpflichtet. Thomas Martinetz danke ich für die Möglichkeit im INB zu Promovieren sowie für die vielen Ideen und Ratschläge. Ebenso danke ich Erhardt Barth, der sich stets Zeit für ein Gespräch genommen hat und hilfreiche Anregungen gab. Des Weiteren möchte ich meiner Kollegin Ingrid für viele spannende Diskussionen und ihre Unterstützung bei Problemen jeder Art danken.

Ebenso danke ich meinen Eltern. Ohne sie wäre ich nicht so weit gekommen. Ihnen, meinen Geschwistern und insbesondere Anna danke ich dafür mich stets ermuntert und in meinem Vorhaben bekräftigt zu haben.

Erklärung an Eides Statt

Ich versichere, dass ich die Dissertation ohne fremde Hilfe angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Weder vorher noch gleichzeitig habe ich andernorts einen Zulassungsantrag gestellt oder diese Dissertation vorgelegt. Ich habe mich bisher noch keinem Promotionsverfahren unterzogen.

(Ort, Datum) (Unterschrift)