ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Sparse approximations for kernel learning vector quantization

Daniela Hofmann, Barbara Hammer*

CITEC center of excellence, Bielefeld University, Germany

Abstract. Various prototype based learning techniques have recently been extended to similarity data by means of kernelization. While state-of-the-art classification results can be achieved this way, kernelization loses one important property of prototype-based techniques: a representation of the solution in terms of few characteristic prototypes which can directly be inspected by experts. In this contribution, we introduce several different ways to obtain sparse representations for kernel learning vector quantization and compare its efficiency and performance in connection to the underlying data characteristics in diverse benchmark scenarios.

1 Introduction

Machine learning techniques have revolutionized the way in which information can automatically be extracted from given data sets in the form of clustering prescriptions, functions, or similar. However, many state-of-the-art techniques such as support vector machines (SVM) essentially act as black boxes. While leading to nearly optimum classification accuracy in many cases, they hardly reveal any insight into why a certain decision has been taken by a given model. In complex settings, however, the latter becomes crucial since human insight is often the only way to further specify a priorly unclear training setting or to substantiate mere observations by causalities as often required in the medical domain, for example. Due to this reason, there is an increasing demand of interpretable models which provide a human understandable interface to their decisions besides excellent classification accuracy [13].

One prominent example is offered by prototype based techniques such as learning vector quantization (LVQ) or generalizations thereof such as proposed in [11, 12]. These techniques rely on prototypical class representatives, and decisions are taken based on the distance of data with respect to these prototypes. Partially, the techniques also provide an inherent low dimensional visualization of their decisions [3]. LVQ methods are restricted to vectorial data only. In recent years, more general data structures are becoming more and more important, such as graphs, trees, sequence data, XML, or the like [5]. Often, these data can be addressed by means of a dedicated similarity measure or kernel.

Several extensions of prototype methods to general distances or kernels have been proposed, see e.g. [8, 6, 1, 10, 7]. In particular relational or kernel approaches have obtained results which are competitive to state-of-the-art alternatives such as SVM [7]. However, one important property of prototype-based techniques is often lost: prototypes are no longer given as explicit representative points in the data space, rather, an indirect representation as a linear combination of an underlying (usually not explicitly given) feature space is used. Thus, interpretability of the models is lost.

*This work has been supported by the DFG under grant number HA2719/7-1 and by the CITEC center of excellence.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

In this contribution, we address the question of how solutions can be approximated by sparse representations of the prototypes which offer a direct interface to human experts. Thereby, we will compare different approaches: a simple approximation of the prototypes by their closest data, an approximation of prototypes by their closest sparse linear combination, and sparse training of LVQ networks. We compare the result for different benchmark data sets.

2 Kernel robust soft learning vector quantization

We rely on a recent kernelized version of a probabilistic LVQ model [12, 7]. Assume data $\xi_k \in \mathbb{R}^n$ are labeled y_k. A robust soft LVQ (RSLVQ) network represents a mixture distribution characterized by m prototypes $w_j \in \mathbb{R}^n$. The labels of prototypes $c(w_j)$ are fixed. σ_j denote the bandwidths. Mixture component j induces $p(\xi|j) = \text{const}_j \cdot \exp(f(\xi, w_j, \sigma_j^2))$ with normalization constant const_j and function $f(\xi, w_j, \sigma_j^2) = -\|\xi - w_j\|^2/\sigma_j^2$. The probability of data point ξ is defined as mixture $p(\xi|W) = \sum_j P(j) \cdot p(\xi|j)$ with prior $P(j)$ and parameters W of the model. The probability of a data point ξ and a given label y is $p(\xi, y|W) = \sum_{c(w_j)=y} P(j) \cdot p(\xi|j)$. Learning aims at an optimization of the log likelihood ratio

$$L = \sum_k \log \frac{p(\xi_k, y_k|W)}{p(\xi_k|W)} .$$

For optimization, usually a stochastic gradient ascent is used which yields update rules similar to LVQ2.1 provided class priors are equal.

Given a novel data point ξ, its class label is the most likely label y corresponding to a maximum value $p(y|\xi, W) \sim p(\xi, y|W)$. For typical settings, this rule can be approximated by the standard winner takes all rule.

Kernelization of this method assumes a fixed kernel k corresponding to a feature map Φ. We set $k_{kl} := k(\xi_k, \xi_l) = \Phi(\xi_k)^t \Phi(\xi_l)$. Prototypes are represented by linear combinations of data

$$w_j = \sum_m \gamma_{jm} \Phi(\xi_m)$$

The cost function of RSLVQ becomes

$$L = \sum_k \log \frac{\sum_{c(w_j)=y_k} P(j) p(\Phi(\xi_k)|j)}{\sum_j P(j) p(\Phi(\xi_k)|j)} .$$

We assume equal bandwidth $\sigma^2 = \sigma_j^2$, for simplicity. Further, we assume constant prior $P(j)$ and mixture components induced by normalized Gaussians. These can be computed in the data space based on the Gram matrix because of the identity $\|\Phi(\xi_i) - w_j\|^2 = \|\Phi(\xi_i) - \sum_m \gamma_{jm} \Phi(\xi_m)\|^2 = k_{ii} - 2 \cdot \sum_m \gamma_{jm} k_{im} + \sum_{s,t} \gamma_{js} \gamma_{jt} k_{st}$ where the distance in the feature space is referred to by $\| \cdot \|^2$. A stochastic gradient ascent of the cost function with respect to the prototypes can be expressed in terms of the coefficients only [7].This adaptation performs exactly the same updates as RSLVQ in the feature space if prototypes are in the linear span of the data. Often, a further restriction to the convex hull takes place.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Note that, unlike vectorial RSLVQ, prototypes are represented implicitly in terms of linear combinations. The inspection of a prototype thus requires to inspect the coefficients γ_j representing the prototype and all data, the latter usually being characterized in terms of pairwise similarities only. Further, an adaptation step has squared complexity caused by the distributed representation of prototypes. Thus, the method does no longer give interpretable results.

3 Approximation of the prototypes

Kernel RSLVQ (k-RSLVQ) yields prototypes which are implicitly represented as linear combinations of data points $w_j = \sum_m \gamma_{jm} \Phi(\xi_m)$. Since the training algorithm and classification depends on pairwise distances only, simple linear algebra allows us to compute the distance between a data point and a prototype based on the pairwise similarity of the data point and all training data only, i.e. the given Gram matrix, as specified above. However, direct interpretability and sparseness of the prototype is lost this way.

Here we propose different ways to arrive at sparse prototype representations, that means linear combinations where only very few coefficients γ_{jm} do not vanish. In such settings, prototypes can be inspected by referring to the very restricted set of data points which contribute to the prototype. We investigate the following possibilities:

1. *Sparse training:* we enhance the cost function of RSLVQ by a term $S(\gamma)$ which prefers sparse solutions of prototypes. A typical choice is the L_1 norm: $S(\gamma) = \sum_{jm} |\gamma_{jm}|_1$. This constraint is weighted with parameter C which is selected according to the given data set.
2. *K-approximation:* we substitute each prototype by its K closest exemplars in the given data set as concerns the distance $\|\Phi(w_j) - \Phi(\xi_m)\|^2$ in the feature space. The latter can be computed based on the kernel.
3. *K-convex hull:* we delete all but the K largest coefficients γ_{jm} in the coefficient vector γ_j. This is then normalized to 1: $\sum_m \gamma_{jm} = 1$.
4. *Sparse approximation:* we approximate a given prototype w_j by its closest sparse linear combination $\sum_m \alpha_{jm} \Phi(\xi_m)$ with small $|\alpha_j|_0$ where the points $\Phi(\xi_m)$ serve as (possibly overcomplete) basis vectors. Since this problem is NP hard, we use a popular greedy approach as offered by orthogonal matching pursuit (OMP) [2]. Since OMP relies on dot products only, we can do it implicitly based on the kernel.

All approximations result in prototypes which are characterized by a small number of data vectors only. The choices differ in the question whether prototypes directly coincide with exemplars or they are given as linear combinations. For the latter, coefficients can become negative for OMP, while 1. and 3. ensure their location in the convex hull of the data.

4 Experiments

We compare RSLVQ and its sparse approximations on a variety of benchmarks as introduced in [4]. Thereby, we particularly want to check whether characteristics of the data allow to infer which approximation is best suited for the given task. The data sets in [4] consist of similarity matrices which are, in general, non-Euclidean. The matrices are symmetrized and normalized before processing. Since the given similarity matrices do not constitute a valid kernel we apply

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

standard preprocessing tools which transfer a given similarity matrix into a valid kernel, as presented e.g. in [4, 9]. We test the two transformations *Spectrum clip:* set negative eigenvalues of the matrix to 0, *Spectrum flip:* negative eigenvalues are substituted by their positive values. In addition, we test the methods on the raw data neglecting its negative eigenvalues. The characteristics of the data are shown in Fig. 1.

- *Amazon47*: This data set consists of 204 books written by 47 different authors. The similarity is determined as the percentage of customers who purchase book j after looking at book i.
- *AuralSonar*: This data set consists of 100 wide band solar signals corresponding to two classes, observations of interest versus clutter. Similarities are determined based on human perception, averaging over 2 random probands for each signal pair.
- *FaceRec*: 945 images of faces of 139 different persons are recorded. Images are compared using the cosine-distance of integral invariant signatures based on surface curves of the 3D faces.
- *Patrol*: 241 samples representing persons in seven different patrol units are contained in this data set. Similarities are based on responses of persons in the units about other members of their groups.
- *Protein*: 213 proteins are compared based on evolutionary distances comprising four different classes according to different globin families.
- *Voting*: Voting contains 435 samples with categorical data compared by means of the value difference metric. Class labeling into two classes is present.

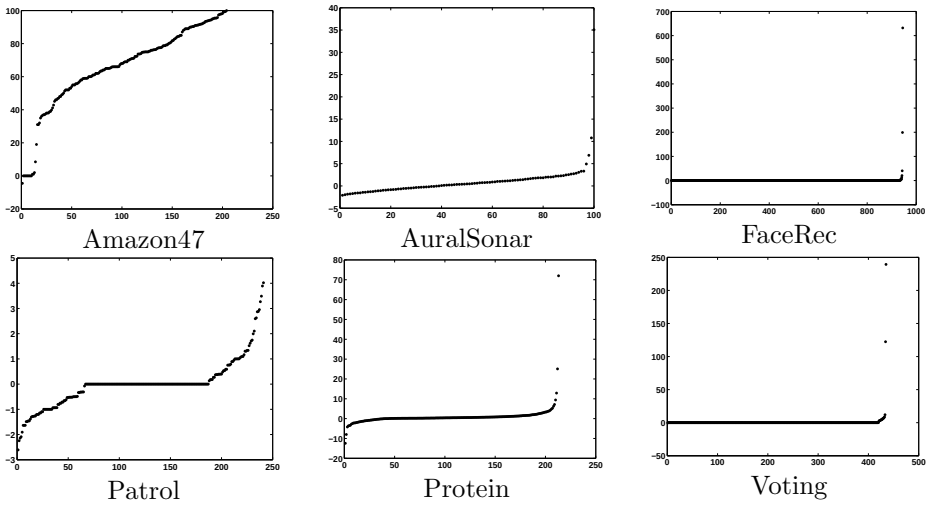

Fig. 1: Characteristic spectrum of the considered similarities. The data sets differ as concerns negative eigenvalues corresponding to non-Euclideanity, and the number of eigenvalues which are different from zero, corresponding to a high dimensional feature space.

For training, we use the same setting as in [4], and we report the results of a 20 fold cross-validation. We compare k-NN and SVM results as reported in [4]

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

with kernel RSLVQ which has been trained with or without sparsity constraint. The settings for standard RSLVQ are taken from [7] as regards parameters. We approximate the solutions obtained using RSLVQ using different values K and different approximation techniques as specified above. If training with sparsity constraint, an appropriate weighting parameter C is determined by binary search such that a desired sparsity is obtained. The parameter C can be very sensitive depending on the data, leading to non-trivial results in a small range only.

	k-NN	SVM	k-RSLVQ		sparse k-RSLVQ		sparsity	
Amazon47	16.95	75.98	15.37	(0.36)	43.40	(0.53)	1.00	72.70
clip	17.68	81.34	15.37	(0.41)	39.92	(0.31)	1.00	72.70
flip	17.56	84.27	16.34	(0.42)	43.18	(0.81)	1.00	72.70
AuralSonar	17.00	14.25	11.50	(0.37)	17.25	(0.78)	11.97	70.08
clip	14.00	13.00	11.25	(0.39)	10.75	(0.30)	12.75	68.11
flip	12.75	13.25	11.75	(0.35)	15.50	(0.76)	12.73	68.17
FaceRec	4.23	3.92	3.78	(0.02)	4.15	(0.01)	1.00	81.88
clip	4.15	4.18	3.84	(0.02)	4.13	(0.02)	1.00	81.88
flip	4.15	4.18	3.60	(0.02)	4.07	(0.02)	1.00	81.88
Patrol	11.88	40.73	17.50	(0.25)	41.67	(0.87)	6.67	72.32
clip	11.56	38.75	17.40	(0.29)	40.00	(0.59)	6.71	72.18
flip	11.67	47.29	19.48	(0.34)	41.56	(0.61)	6.68	72.27
Protein	29.88	2.67	26.98	(0.37)	38.84	(0.74)	22.19	47.77
clip	30.35	5.35	4.88	(0.17)	13.84	(0.38)	13.37	68.54
flip	31.28	1.51	1.40	(0.05)	2.21	(0.10)	13.52	68.19
Voting	5.80	5.52	5.46	(0.04)	5.11	(0.03)	64.35	63.02
clip	5.29	4.89	5.34	(0.04)	5.34	(0.03)	68.67	60.53
flip	5.23	4.94	5.34	(0.03)	5.80	(0.06)	59.92	65.56

Table 1: Results of RSLVQ and sparse training in comparison to alternative classifiers. The standard deviation is given in parenthesis. *Sparsity* refers to the number of nonzero coefficients per prototype in the left column and the loss of density in percent in the right.

The results as reported in Tab. 1 show that the data sets Amazon47 and Patrol do not allow sparse training without deteriorating the classification error. This is mirrored in the high intrinsic dimensionality of these data sets as can be seen from the matrix spectrum. For the other data sets, more than 60% of the coefficients can be set to 0.

Tab. 2 displays the classification error if the learned models are approximated by sparse counterparts. Interestingly, there is no clear favorite which method is best and which degree of sparsity is suited, a larger sparsity not necessarily implying a loss of accuracy (e.g. FaceRec). It seems that sparsity while training does not improve the results of a subsequent sparse approximation in general. Interestingly, some data sets allow an almost lossless representation using one exemplar only (FaceRec, Voting), these data sets happen to be dominated by few large eigenvalues when considering the spectrum. However, the precise approximation technique is vital in the latter case, the optimum relying on the distance only, while the former case allows any type of approximation. Note that all data sets can be accompanied by a sparse model which is almost of the same quality as k-RSLVQ. The optimum approximation technique and whether the technique has an influence at all, however, varies among the data sets.

5 Discussion

We have proposed different ways to arrive at sparse solutions for RSLVQ schemes which open the way towards interpretable prototypes also for kernel RSLVQ.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

	k-RSLVQ					sparse k-RSLVQ				
	k-approximat		k-convex hull		OMP	k-approximat		k-convex hull		OMP
Ama	36.10	41.12	32.02	**15.37**	22.68	43.91	49.25	43.08	43.08	46.12
clip	31.65	43.17	31.45	**15.00**	20.00	42.02	52.43	40.85	40.85	40.84
flip	31.28	45.73	33.15	**16.46**	20.37	43.85	56.59	43.21	43.21	43.57
Aur	25.13	20.00	55.94	25.00	38.00	29.08	26.67	29.81	**17.50**	29.75
clip	24.75	15.00	58.50	23.25	15.00	20.75	16.75	22.25	**11.00**	15.50
flip	24.75	17.62	61.50	19.75	26.00	26.00	21.50	27.50	**15.00**	26.25
Fac	3.70	36.93	3.84	3.78	**3.68**	4.14	37.18	4.15	4.15	4.15
clip	3.76	36.97	3.92	3.84	**3.68**	4.10	37.24	4.13	4.13	4.13
flip	**3.33**	36.98	4.21	3.60	3.60	4.13	37.12	4.07	4.07	4.07
Pat	54.31	25.19	67.98	26.77	48.75	54.36	**24.68**	65.09	40.94	53.85
clip	32.46	**18.86**	38.82	24.38	29.69	28.67	22.86	37.03	40.21	46.25
flip	37.42	**20.60**	40.63	25.42	33.33	29.47	24.23	40.12	41.98	49.90
Pro	55.12	47.53	42.09	**33.14**	52.44	48.76	47.80	44.53	43.95	57.79
clip	22.44	29.38	36.28	27.44	52.09	36.63	33.57	30.12	**14.77**	30.70
flip	23.26	24.88	25.35	3.95	49.07	30.81	28.26	18.84	**3.02**	26.74
Vot	**8.56**	9.48	86.21	82.53	15.57	13.59	15.69	62.82	41.15	15.52
clip	8.65	11.44	86.44	82.76	**5.34**	13.62	13.45	65.34	44.02	**5.34**
flip	7.84	10.03	86.95	82.53	**5.46**	12.82	17.42	65.46	38.39	6.55

Table 2: Results of sparse approximations of the obtained classifiers. For the K-approximation and -convex hull the left column refers to $K = 1$ and for the right value $K = 10$ was chosen. For OMP, the sparsity arises from the problem formulation by setting the approximation quality. The best results are shown in boldface.

Interestingly, it is indeed possible to obtain sparse representations for all data sets within a benchmark suite, however, the optimum method varies. Very simple techniques such as an approximation by the closest prototypes seems to work as well as formal optimization approaches such as provided by OMP.

References

[1] R. Boulet, B. Jouve, F. Rossi, N. Villa. Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257-1273, 2008.

[2] A. M. Bruckstein, D. L. Donoho, M. Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM Review*, Vol. 51(1):34-81, 2009.

[3] K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, M. Biehl. Limited Rank Matrix Learning, discriminative dimension reduction and visualization. *Neural Networks* 26:159-173, 2012.

[4] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, L. Cazzanti. Similarity-based classification: Concepts and algorithms. *JMLR*, 10:747-776, June 2009.

[5] T. Gärtner. *Kernels for Structured Data*. PhD thesis, Univ. Bonn, 2005.

[6] B. Hammer, A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229-2284, 2010.

[7] D. Hofmann, B. Hammer. Kernel robust soft learning vector quantization. In *ANNPR'12*, 14-23.

[8] T. Kohonen, P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8-9):945-952, 2002.

[9] E. Pekalska, R. P. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.

[10] A. K. Qin, P. N. Suganthan. A novel kernel prototype-based learning algorithm. In *Proc. of the 17th International Conference on Pattern Recognition* (ICPR '04).

[11] P. Schneider, M. Biehl, B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942-2969, 2009.

[12] S. Seo, K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589-1604, 2003.

[13] A. Vellido, J. D. Martín-Guerroro, P. J. G. Lisboa. Making machine learning models interpretable. In *ESANN'12*. 2012.