International Journal of Neural Systems, Vol. 0, No. 0 (April, 2000) 00–00
© World Scientific Publishing Company

Efficient kernelized prototype based classification

F.-M. Schleif*
Dept. of Techn., Univ. of Bielefeld, Universitätsstrasse 21-23
33615 Bielefeld, Germany
E-mail: schleif@informatik.uni-leipzig.de

Thomas Villmann
Faculty of Math./Natural and CS, Univ. of Appl. Sc. Mittweida, Technikumplatz 17
09648 Mittweida, Germany
E-mail: villmann@hsmw.de

Barbara Hammer
Dept. of Techn., Univ. of Bielefeld, Universitätsstrasse 21-23
33615 Bielefeld, Germany
E-mail: bhammer@techfak.uni-bielefeld.de

Petra Schneider
University of Birmingham, School of Clinical & Experimental Medicine
Birmingham B15 2TT, United Kingdom
E-mail: p.schneider@bham.ac.uk

Received (to be inserted
Revised by Publisher)

Prototype based classifiers are effective algorithms in modeling classification problems and have been applied in multiple domains. While many supervised learning algorithms have been successfully extended to kernels to improve the discrimination power by means of the kernel concept, prototype based classifiers are typically still used with Euclidean distance measures. Kernelized variants of prototype based classifiers are currently too complex to be applied for larger data sets. Here we propose an extension of Kernelized Generalized Learning Vector Quantization (KGLVQ) employing a sparsity and approximation technique to reduce the learning complexity. We provide generalization error bounds and experimental results on real world data, showing that the extended approach is comparable to SVM on different public data.

1. INTRODUCTION

The dramatic growth in data generating applications and measurement techniques has created many high-volume and high-dimensional data sets. Most of them are stored digitally and need to be efficiently analyzed to be of use. Clustering and classification methods are very important in this setting and have been extensively studied in the last decades [17,35,33,1,24,36,10]. Challenges are mainly in the timely, memory efficient and accurate processing of such data also in the case of non linearly separable data with multiple thousand items.

Kernelized learning vector quantization (KGLVQ) was proposed in the approach[28] as an extended approach of Generalized Learning Vector Quantization (GLVQ) [29] with the goal to provide modeling capabilities for learning vector quantizers and to improve the performance in classification tasks. While the approach was quite promising it

*corresponding author

has been used only rarely due to its complexity. One challenge is the storage of a large kernel matrix and additionally the storage and update of a combinatorial coefficient matrix Ψ, implicitly representing the prototypes. This makes the approach inapplicable already for data with a comparably small number of items.

Data analysis using kernel methods is an active field of research [3,6,25] offering solutions for the analysis of complex problems. The involved kernel matrix needs, in its *original* form, quadratic space with the number of samples and involves usually cubic time complexity which can be quite demanding for large problems. This pose a big challenge on practical applications.

Modern approaches in discriminative kernel based learning like Sequential Minimal Optimization[27] and other try to avoid the direct storage and usage of the full kernel matrix or restrict the underlying optimization problem to subsets thereof[27,35,37]. For the KGLVQ approach such a strategy has not been proposed so far.

The Nyström-Approximation of Gram matrices constitutes a classical approximation scheme [41,22], permitting the estimation of the kernel matrix by means of a low dimensional approximation. We will employ this method for the approximation of the distance calculations based on the kernel matrix as the key element of our accelerated kernelized GLVQ (AKGLVQ). A further issue with kernel methods is the model complexity by means of the stored data points. In case of the well known support vector machine (SVM) [38], these are the so called support vectors (SV). The number of SVs can become quite large for complex problems. Novel learning methods for the SVM try to shrink this value [19].

In case of KGLVQ the prototypes are implicitly modeled by a coefficient matrix over all data points which is typically dense. This however is not necessary for most data sets.

Sparsity is a natural concept in the encoding of data [26] and can be used to obtain compact sparse models. This concept has been used in many machine learning methods [21,16] and different measures of sparsity have been proposed [26,16]. Taking this into account we propose to integrate a sparsity constraint into KGLVQ allowing the explicit control of the sparsity of the coefficient matrix.

Both optimization concepts, Nyström and sparsity, are used to improve the complexity of KGLVQ such that it becomes applicable for large data sets.

In Sec. 2 we present a short introduction into kernels and give the notations used throughout the paper. Subsequently we present the KGLVQ algorithm and its approximated variant AKGLVQ by means of the Nyström approximation and the additional sparsity constraint. We show the efficiency of the novel approach for experiments on artificial and real life data. Finally, we conclude with a discussion.

2. PRELIMINARIES

We consider a set of vectors $\mathbf{v}_i \in \mathbb{X}^{\mathbb{D}}$ with $\mathbb{X}^{\mathbb{D}} \subseteq \mathbb{R}^{\mathbb{D}}$, $\mathbb{D}$ denoting the dimensionality and $|\mathbb{X}| = N$ the number of samples. Further we introduce prototypes $\mathbf{w}_j \in \mathbb{W}^{\mathbb{D}}$, with $|\mathbb{W}^{\mathbb{D}}| = M$ which induce a clustering of $\mathbb{X}^{\mathbb{D}}$ by means of their receptive fields consisting of the points $\mathbf{v}$ for which $d(\mathbf{v}, \mathbf{w_j}) \leq d(\mathbf{v}, \mathbf{w_l})$ holds for all $j \neq l$ and d denoting a distance measure, typically the Euclidean distance. Further we introduce $c(\mathbf{v}) \in \mathcal{L}$ as the label of input $\mathbf{v}$, and $c(\mathbf{w})$ as the label of the prototype $\mathbf{w}$, respectively. $\mathcal{L}$ denotes the set of labels (classes) with $\#\mathcal{L} = N_{\mathcal{L}}$. Let $\mathbb{W}_c = \{\mathbf{w_l} | c(\mathbf{w}_l) = c\}$ be the subset of prototypes assigned to class $c \in \mathcal{L}$.

We also introduce two special notations for the prototype which is closest to a given point $\mathbf{v}_i$ with the same label: $\mathbf{w}^+$ or a different label: $\mathbf{w}^-$. The corresponding distance d_i^+, d_i^-:

$$
\begin{aligned}
d_i^+ &= d(\mathbf{w}^+, \mathbf{v}_i) \text{ with } \mathbf{w}^+ \in \mathbb{W}_c,\ c = c(\mathbf{v}_i), && (1)\\
\mathbf{w}^+ &:= \mathbf{w}_l : d(v_i, w_l) \leq d(v_i, w_j), \{\mathbf{w}_j, \mathbf{w}_l\} \in \mathbb{W}_c && (2)\\
d_i^- &= d(\mathbf{w}^-, \mathbf{v}_i) \text{ with } \mathbf{w}^- \notin \mathbb{W}_c,\ c = c(\mathbf{v}_i) && (3)\\
\mathbf{w}^- &:= \mathbf{w}_l : d(v_i, w_l) \leq d(v_i, w_j), \{\mathbf{w}_j, \mathbf{w}_l\} \notin \mathbb{W}_c
\end{aligned}
$$

Equation (2) is sometimes also referred as the *winner takes all* (wta) rule restricted to the $\mathbf{w}$ of the same class as $\mathbf{v}$.

Complex data are often not linearly separable in the Euclidean space and it was suggested to map the data $\mathbb{X}$ into a high dimensional Hilbert space $\mathbb{H}$ using a mapping function $\phi : \mathbb{X} \to \mathbb{H}$ to separate the data in a linear manner [33]. The explicit definition of an appropriate mapping ϕ can be complex for the high-dimensional feature space. As pointed out in [33] this explicit formulation is often not necessary, if we are able to express the calculation in our learning algorithm by means of inner products. If we have a positive semi-definite inner product function $\kappa(\mathbf{v}, \mathbf{v}')$, fulfilling the Mercer conditions we can expand it by means of its eigenvalues and eigenfunctions:

$$\kappa(\mathbf{v}, \mathbf{v}') = \sum_i^{\infty} \lambda_i \phi_i(\mathbf{v}) \phi_i(\mathbf{v}') = \langle \phi(\mathbf{v}), \phi(\mathbf{v}') \rangle_{\mathcal{F}} \quad (4)$$

Now we can express the inner products in the feature space based on the kernel function κ calculated in the Euclidean space using e.g. a Gaussian kernel $\kappa(\mathbf{v}, \mathbf{v}') = \exp(-\|\mathbf{v} - \mathbf{v}'\|^2/\sigma^2)$ [8]. The calculation in the GLVQ can be done based on inner products such that (4) is applicable as used to derive the KGLVQ[28].

3. ALGORITHM

Learning vector quantization (LVQ) is a supervised learning scheme. It was introduced as a generic concept for intuitive prototype-based classification algorithms [20]. Several variants were developed to improve the standard algorithms [13,29,34]. LVQ algorithms are based on the empirical risk minimization (ERM) principle and describe the data space by means of prototypical representants (vectors), which are in general elements of the original data space. The main benefit, beside of its good generalization performance [14], is the direct access to the model constituents by means of the prototypes. The prototypes can be directly inspected and provide human interpretable information about typical aspects of the represented data classes.

Generalized Learning Vector Quantization (GLVQ) is an extension of the standard LVQ providing a cost function [29] recently extended in two kernelized variants [28,31]. It is a margin optimization method [7] and can inherently deal with multi class data. Moreover, it is effective also under different distance measures and objectives [11]. The kernelized variants of GLVQ, namely KGLVQ and differentiable kernelized GLVQ (D-KGLVQ) are effective extensions of the original GLVQ concept but suffer from its high complexity or limitations regarding the kernel choice [31]. Subsequently, we briefly review the concepts of GLVQ and KGLVQ which will be extended, by two optimization techniques, yielding AKGLVQ later on.

3.1. *Standard GLVQ*

The cost function for GLVQ is given as

$$E = Cost_{GLVQ} = \sum_i^N \mu(\mathbf{v_i}) \quad \mu(\mathbf{v_i}) = \frac{d_i^+ - d_i^-}{d_i^+ + d_i^-} \quad (5)$$

which is optimized with respect to the free parameters (here the prototypes), by stochastic gradient descent. Note that the classifier function $\mu(\mathbf{v})$ is positive if the vector $\mathbf{v}$ is misclassified and negative otherwise.

The learning rule of GLVQ is obtained taking the derivatives of the above cost function with respect to the parameters $\mathbf{w}$. Using $\frac{\partial \mu(\mathbf{v}_i)}{\partial \mathbf{w}^+} = \xi^+ \frac{\partial d_i^+}{\partial \mathbf{w}^+}$ and $\frac{\partial \mu(\mathbf{v}_i)}{\partial \mathbf{w}^-} = \xi^- \frac{\partial d_i^-}{\partial \mathbf{w}^-}$ with[†]

$$\xi^+ = \frac{2 \cdot d_i^-}{(d_i^+ + d_i^-)^2} \quad \xi^- = \frac{-2 \cdot d_i^+}{(d_i^+ + d_i^-)^2} \quad (6)$$

one obtains for the weight updates [12]:

$$\triangle \mathbf{w}^+ = \epsilon^+ \cdot \xi^+ \cdot \frac{\partial d_i^+}{\partial \mathbf{w}^+} \quad \triangle \mathbf{w}^- = \epsilon^- \cdot \xi^- \cdot \frac{\partial d_i^-}{\partial \mathbf{w}^-} \quad (7)$$

with $\epsilon^{+/-}$ as learning rates, which are typically in the range of 10^{-5}.

3.2. *Kernelized GLVQ*

We now briefly review the main concepts used in Kernelized GLVQ (KGLVQ) as given in the paper of Qin[28]. The KGLVQ makes use of the same cost function as GLVQ but with the distance calculations done in the kernel space. Under this setting the prototypes cannot explicitly be expressed as vectors in the feature space due to lack of knowledge about the feature space. Instead Qin[28] models the feature space as a linear combination of all images $\phi(\mathbf{v})$ of the datapoints $\mathbf{v}$. Thus a prototype vector may be described by some linear combination of the

[†] Divisions including vectors are used element-wise throughout the paper.

feature vectors: $\mathbf{w}_j = \sum_{l=1}^N \psi_{j,l}\phi(\mathbf{v}_l)$, $\psi_j \in \mathbb{R}^N$ is the corresponding coefficient vector. The distance in feature space for a given $\phi(\mathbf{v}_i)$ and $\mathbf{w}_j$ is computed as:

$$\begin{aligned} d^2_{i,j} &= \|\phi(\mathbf{v}_i) - \mathbf{w}_j\|^2 = \|\phi(\mathbf{v}_i) - \sum_{l=1}^N \psi_{j,l}\phi(\mathbf{v}_l)\|^2 \\ &= k(\mathbf{v}_i,\mathbf{v}_i) - 2\sum_{l=1}^N k(\mathbf{v}_i,\mathbf{v}_l)\cdot\psi_{j,l} \qquad (8) \\ &\quad + \sum_{s,t=1}^N k(\mathbf{v}_s,\mathbf{v}_t)\cdot\psi_{j,s}\psi_{j,t} \end{aligned}$$

The update rules of GLVQ can be modified by substituting the Euclidean distance by Equation (8) and taking derivatives with respect to the coefficients $\psi_{j,l}$. The detailed equations are available in [28], a simplified version for the coefficient update is given later on. The final model consists of the pre-calculated kernel matrix and the combinatorial coefficient matrix for the ψ coefficients.

3.3. *Approximation of the kernel matrix by Nyström*

As pointed out in the paper of Zhang[41], different strategies have been proposed to overcome the complexity problem caused by the kernel matrix K in modern machine learning algorithms. One promising approach is the Nyström approximation.

It originates from the numerical treatment of integral equations of the form $\int \mathcal{P}(y)k(x,y)\phi_i(y)dy = \lambda_i\phi_i(x)$ where $\mathcal{P}(\cdot)$ is the probability density function, k is a positive definite kernel function, and $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ are the eigenvalues with $\phi_1, \phi_2, \ldots$ the respective eigenfunctions of this integral equation. Given a set of i.i.d. samples $\{x_1,\ldots,x_q\}$ drawn from $\mathcal{P}(\cdot)$, the basic idea is to approximate the integral by the empirical average

$$1/q\sum_{j=1}^q k(x,x_j)\phi_i(x_j) \approx \lambda_i\phi_i(x)$$

which can be written as the eigenvalue decomposition: $K\phi = q\lambda\phi$. $K_{q\times q} = [K_{i,j}] = [k(x_i,x_j)]$ is the kernel matrix defined on X, and $\phi = [\phi_i(x_j)] \in \mathbb{R}^q$. Solving this equation we can calculate $\phi_i(x)$ as

$$\phi_i(x) \approx 1/(q\lambda)\sum_{j=1}^q k(x,x_j)\phi_i(x_j)$$

which is costly. To reduce the complexity, one may use only a subset of the samples which is commonly known as the Nystöm method.

Suppose the sample set $V = \{\mathbf{v}_i\}_{i=1}^N$, with the corresponding $N\times N$ kernel matrix K. We randomly choose a subset $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^q, \mathbf{Z} \subset V, q << N$ of landmark points and a corresponding kernel sub matrix $\mathbf{Q}_{q\times q} = [k(\mathbf{z}_i,\mathbf{z}_j)]_{i,j}$. We calculate the eigenvalue decomposition of this sub matrix: $\mathbf{Q}\phi_z = q\lambda_z\phi_z$ and obtain the corresponding eigenvector $\phi_z \in \mathbb{R}^q$ and the eigenvalue $q\lambda_z$. Subsequently we calculate the interpolation matrix $\hat{\mathbf{K}}_{N\times q} = [k(\mathbf{v}_i,\mathbf{z}_j)]_{i,j}$ to extend the result to the whole set V. We approximate the eigen-system of the full $K\phi_K = \phi_K\lambda_K$ by [39]:

$$\phi_K \approx \sqrt{\frac{q}{N}}\hat{\mathbf{K}}\phi_Z\lambda_{\mathbf{Z}}^{-1}, \lambda_K \approx \frac{N}{q}\lambda_{\mathbf{Z}}.$$

K can be subsequently reconstructed as

$$\begin{aligned} K &\approx \left(\sqrt{\frac{q}{N}}\hat{\mathbf{K}}\phi_Z\lambda_{\mathbf{Z}}^{-1}\right)\left(\frac{N}{q}\lambda_{\mathbf{Z}}\right)\left(\sqrt{\frac{q}{N}}\hat{\mathbf{K}}\phi_Z\lambda_{\mathbf{Z}}^{-1}\right)' \\ &= \hat{\mathbf{K}}\mathbf{Q}^{-1}\hat{\mathbf{K}}' \end{aligned}$$

To integrate the Nyström approximation into KGLVQ we only need to modify the distance calculation between a prototype $\mathbf{w_j}$ and a data point $\mathbf{v_i}$ which can be expressed using the Nyström approximation. In KGLVQ the prototypes are expressed by means of a linear combination of the datapoints in the feature space as shown in [28]. Hence it is sufficient to update the coefficients of this linear combination. The original update equation for the coefficient matrix in KGLVQ read as:

$$\psi^{t+1}_{\pm,\mathbf{r}'} = \begin{cases} [1 \mp \epsilon\cdot\frac{4\cdot d_i^{\mp}}{d_i^{\pm}+d_i^{\mp}}]\cdot\psi^t_{\pm,\mathbf{r}'} & \text{if } \mathbf{v}_{\mathbf{r}'} \neq \mathbf{v}_i \\ [1 \mp \epsilon\cdot\frac{4\cdot d_i^{\mp}}{d_i^{\pm}+d_i^{\mp}}]\cdot\psi^t_{\pm,\mathbf{r}'} & \\ +\epsilon\cdot\frac{4\cdot d_i^{\mp}}{d_i^{\pm}+d_i^{\mp}} & \text{if } \mathbf{v}_{\mathbf{r}'} = \mathbf{v}_i \end{cases}$$

with $t+1$ indicating the coefficient ψ after the update. A single prototype update has a complexity of $O(N^2)$, due to the double sum in (8). The index or superscript $\pm$ corresponds to the prototype with the same $(+)$ or different $(-)$ label as the data point $\mathbf{v_i}$ as already defined previously. The point

$\mathbf{v}_i$ is the current point used in the iterative gradient descend optimization. The index r' refers to the considered datapoint in the linear combination (the column index of Ψ). The KGLVQ update above is almost identical for the AKGLVQ but the distance calculations are done using the Nyström approximation with Equation (9):

$$\begin{aligned} d_{\cdot,i} &= K(i,i) - 2 \cdot T_{\cdot,i} + \operatorname{diag}(\Psi \cdot T') && (9) \\ \text{with } T_{j,\cdot} &= ((\psi_j \cdot \hat{\mathbf{K}}) \cdot \mathbf{Q}^{-1}) \cdot \hat{\mathbf{K}}' && (10) \end{aligned}$$

where diag provides diagonal elements of the associated matrix. Using Nyström-approximation, the complexity in AKGLVQ is reduced to $O(q^3 + qN)$, caused by the SVD (for some recent work on SVD see[18]) to calculate the inverse of the matrix in the Nyström-approximation and the remaining distance calculation costs [39].

3.4. *Sparse coefficient matrix*

In the paper of Olshausen[26], sparsity has been found to be a natural concept in the visual cortex of mammals. This work motivated the integration of sparsity concepts into many machine learning methods to obtain sparse and efficient models. Here we will integrate sparsity as an additional constraint on the coefficient matrix Ψ such that the amount of non-zero coefficients is limited. This leads to a more compact descriptions of the prototypes, by means of a smaller linear mixture model. The used sparsity measure is the one as given in Olshausen[26]. The sparsity $\mathbb{S}$ of a row of α is measured as

$$\mathbb{S}(\psi_j) = -\sum_{l=1}^{N} S\left(\frac{\psi_{j,l}}{\sigma}\right) \qquad (11)$$

with σ as a scaling constant. The function S can be of different type, here we use $S(x) = \log(1 + x^2)$. We extend the energy function of the KGLVQ by an additional term:

$$E_{AKGLVQ}(\gamma) = E_{KGLVQ}(\gamma) - \beta \mathbb{S}(\psi_{\mathbf{j}}) \qquad (12)$$

The updates for the coefficients of $\mathbf{w}_i$ are structurally similar to those given in the standard KGLVQ using the Nyström formula to approximate the Gram matrix but include the additional term

$$\frac{\partial \mathbb{S}}{\partial \psi_{j,l}} = -\frac{2/\sigma^2 \cdot \psi_{j,l}}{1 + (\psi_{j,l}/\sigma)^2},$$

we restrict the coefficients to be $\psi_{j,i} \in [0,1]$ and bound them by $\sum_i \psi_{j,i} = 1$.

The effect of the sparsity constraint in AKGLVQ on the UCI iris data [4] with one prototype per class is shown in Figure 1. Both models achieve an accuracy of $\approx 90\%$ using a linear kernel. The sparsity constraint effectively helps to reduce the necessary memory of the matrix Ψ. Yet, the associated parameters have to be chosen adequately to balance sparsity and classification accuracy. The sparsity constraint could also be used to speed up the algorithm, by explicit omit operations involving multiplications with zero. This, however, requires a very careful and efficient implementation of the sparsity handling which is not easily accessible within the used runtime Matlab. During the classification step a sparse matrix Ψ can significantly limit the number of distance calculations necessary to map a new item in the feature space and to calculate the distance to a prototype. In the worst case with a dense matrix Ψ we get linear complexity $O(M \times N)$ to calculate the inner products for a new point, whereas a sparse matrix Ψ will typically scale in constant complexity $O(k \times M)$, assuming e.g. a k-approximation of the prototypes.

3.5. *Generalization ability of KGLVQ*

It has been shown in the approaches [7,32] that generalization bounds for LVQ schemes can be derived based on the notion of the hypothesis margin of the classifier μ, independently of the input dimensionality. Rather the margin, i.e. the difference of the distance of points to its closest correct and wrong prototype, determine the generalization ability. This fact makes the algorithm particularly suitable for kernelization: essentially, the generalization ability transfers directly to the kernel version because of the fixed implicit embedding into the feature space. Thereby, large margin bounds are of particular interest due to the usually high dimensionality of the feature space. Bounds which depend on the number of free parameters would likely yield very weak bounds in such cases. For GLVQ as a large margin approach, a straightforward transfer of the bounds as provided in the approaches [7,32] based on techniques as given in the article [2] is possible.

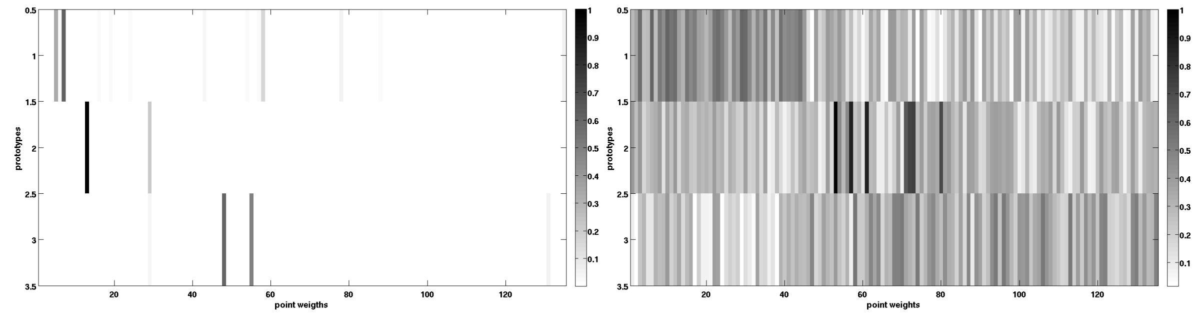

Figure 1: Effect of the sparsity constraint for the UCI iris data shown by means of the Ψ-matrix (normalized for better comparison). With sparsity (left), without sparsity right. Dark values indicated high loaded or high lighted data points for the considered prototype in the Ψ matrix. Data points with very low values over all prototypes can be safely removed from the model.

For convenience, we shortly review the setting as formalized e.g. in the derivation [32]. For simplicity, a classification by a kernelized prototype-based network into two classes is considered. We label prototypes corresponding to the two classes with + and −, respectively. Classification takes place by a winner takes all rule (2), i.e., taking the kernel into account, a data point is mapped to the class

$$f : \mathbf{v} \mapsto \operatorname{sgn}\left(\min_{\mathbf{w}^+} \|\Phi(\mathbf{v}) - \mathbf{w}^+\| - \min_{\mathbf{w}^-} \|\Phi(\mathbf{v}) - \mathbf{w}^-\|\}\right) \tag{13}$$

where sgn selects the sign of the term. A trainable KGLVQ network corresponds to a function f in this class with M prototypes. We can assume that data $\mathbf{v}$ are bounded in size. Thus, also the images $\Phi(\mathbf{v})$ and the possible location of prototype vectors are bounded in size, we refer to the bound by B.

As usual, generalization bounds aim at limiting the generalization error $E_P(f) = P(f(\mathbf{v}) \neq c(\mathbf{v}))$ where P refers to a (probably unknown) probability distribution P. The margin of the classification is obtained by dropping the sign in (13) leading to the related function M_f. For a fixed positive value of the margin ρ and the associated loss

$$L : \mathbf{R} \to \mathbf{R}, t \mapsto \begin{cases} 1 & \text{if } t \leq 0 \\ 1 - t/\rho & \text{if } 0 < t \leq \rho \\ 0 & \text{otherwise} \end{cases}$$

a connection of the generalization error and the empirical error on m samples

$$\hat{E}^L_m(f) = \sum_{i=1}^{m} L(c_{\mathbf{v}} \cdot M_f(\mathbf{v}))/m \tag{14}$$

can be established with probability $\delta > 0$ simultaneously for all functions f using techniques of [2]:

$$E_P(f) \leq \hat{E}^L_m(f) + \frac{2}{\rho} R_m(M_{\mathcal{F}}) + \sqrt{\frac{\ln(4/\delta)}{2m}}$$

$R_m(M_{\mathcal{F}})$ denotes the so-called Rademacher complexity of the class of functions implemented by KLVQ networks with function M_f. The quantity can be upper bounded, using techniques of [32] and structural properties given in [2], by a term

$$\mathcal{O}\left(\frac{N^{2/3}B^3 + \sqrt{\ln(1/\delta)}}{\sqrt{m}}\right)$$

The quantity B depends on the kernel and can be estimated depending on the data distribution. Thus, generalization bounds for KGLVQ with arbitrary kernel result which are comparable to generalization bounds for GLVQ. Note that the only difference as compared to the derivation as provided in [32] consists in the fact that data are implicitly embedded in the feature space such that B depends on the given data points and the kernel.

4. EXPERIMENTS

We analyze our approach using artificial and real life data. The simulated data shall be considered as a toy data set to show the possibility to deal with non-linear separable data distributions, which is a typical application field of kernel methods. Subsequently we provide some analysis for very well known standard test data, followed by more complicated data sets which can not be processed by KGLVQ under reasonable time and memory settings. It should be pointed out that KGLVQ can be applied to very different types of problems, ranging from life-science data [40] to e.g. image processing tasks [23], as long as a valid kernel can be provided.

4.1. *Simulated data*

We start with the non-linear separable ring data set (DS1) and an RBF kernel in the distance measure. The data consist of 800 data points with 400 per ring in 2 dimensions as shown in Figure 2. The first ring has a radius of $r = 10$ and the second $r = 4$, points are randomly sampled in $[0, 2\pi]$. The data set has been normalized in $N(0, 1)$. We also analyzed the ring data using the additional sparsity constraint. In the original model 53% of the weights, averaged over the prototypes are almost 0 (values $\leq 1e - 5$). In the sparsity approach we used σ^2 as the variance of the data scaled by 0.01 and a $\beta = 1$ and obtained a much sparser model with now 75% of the points close to zero.

4.2. *Small sample size data*

Now we present a comparison for 3 benchmark datasets taken from the UCI repository [4], namely the breast cancer data (wdbc), a diabetes study (pima) and the heart data set, used to predict a heart disease. All these data sets are two class examples with $N < 1000$, details are given in Table 1.

We analyze the performance of KGLVQ, AKGLVQ and SVM using the recently proposed Extreme Learning Kernel (ELM) [9]. The ELM kernel is actually a defacto parameter free kernel with the same classification performance as the RBF kernel with optimal σ [9], it has been fixed to $1e10$ in this study. SVM models are obtained by use of a Sequential Minimization Optimization (SMO) optimizer as proposed in [27] and the ELM kernel.

All AKGLVQ and KGLVQ models are obtained with 1 prototype per class, using $C = 100$ cycles and with a nyström approximation of $q = 0.1 \times N$ of the original kernel matrix for the AKGLVQ variant, sparsity was switched *off*. The value of the nyström approximation is not so critical but should be not lower than 10% to keep sufficient approximation accuracy. It has mainly an influence on the runtime performance as long as the data space is sufficiently densely sampled.

The results of AKGLVQ compare favorable in comparison to KLVQ or SVM but especially the runtime is significantly improved with respect to KGLVQ see Table 1. We find that the prediction performance of AKGLVQ and KGLVQ are quite similar, and both are competitive to SVM. KGLVQ however is not really applicable for larger data sets due to the costly distance calculations using the full kernel.

4.3. *Complexity and runtime analysis*

The original KGLVQ algorithm employs a full, quadratic kernel matrix in the distance calculations and is optimizing the $M \times N$ coefficient matrix Ψ. The selected underlying iterative optimization scheme is gradient descend. The optimization is done for C cycles as an upper limit and often independent of the data chosen as $C = 100$. The number of prototypes is typically chosen independent to the real data set size and in general much smaller than N, such that the memory complexity of Ψ is linear in $O(M \times N)$. Taking this into account KGLVQ has a memory complexity of roughly $O(N^2)$. Each distance calculation involves matrix/vector operations with a N^2 matrix which has to be done for all N data point and for C cycles. Hence the runtime complexity is in the range of $O(N^3)$.

The AKGLVQ algorithm provides two approaches to optimize runtime and memory complexity, namely the Nyström approximation and the sparsity constraint as pointed out before. Using the Nyström approximation the memory complexity of the kernel matrix is reduced to $O(q \times N)$. Hence the necessary memory to store the kernel matrix as well as the number of matrix operation is directly reduced depending on q. For most data sets it is reasonable to set q to a small fraction of N e.g. 10%. The memory complexity of the matrix Ψ is unaffected. This leads to an estimated linear memory consumption of $O((q + M) \times N)$. To obtain the two matrices of the Nyström approximation a (pseudo) inverse has to be calculated and for the Nyström based distance calculation additional multiplications by a $q \times N$ matrix

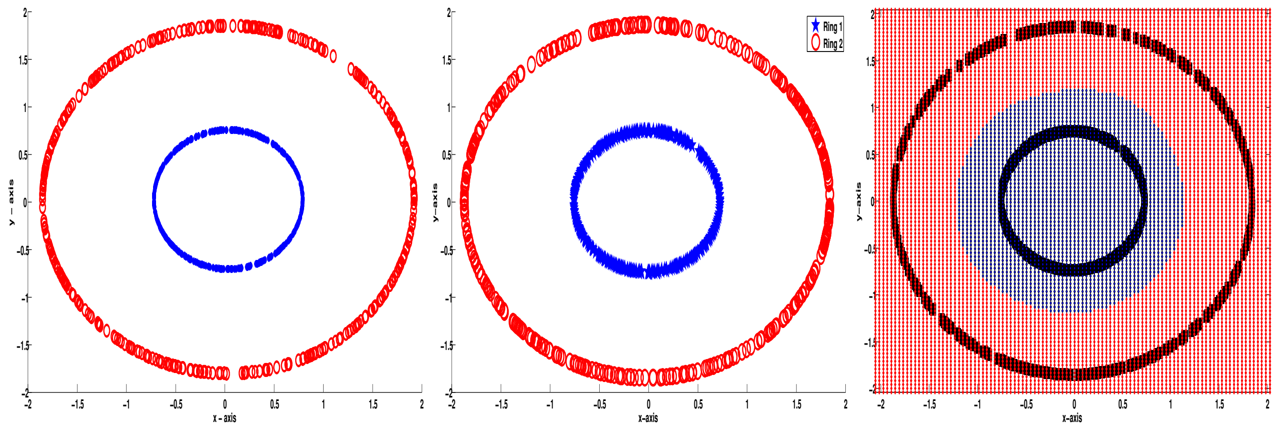

Figure 2: Ring data set (1st plot), KGLVQ model (2nd plot), the outer ring is shown in red and using 'o' while the inner ring is plotted in blue $\star$. The 3rd plot shows the cluster boundaries of the model from the 2nd plot. The model was calculated without sparsity. It can be clearly seen that the AKGLVQ with an rbf kernel successfully separated these two clusters and also the cluster boundaries are very well approximated with a large margin between the two rings.

	Dim	Size	KGLVQ		AKGLVQ		SVM	
				#PT		#PT		#SV
Breast Cancer	32	569	92.97±01.87	2	92.27±03.43	2	97.71±01.45	512
Diabetes	8	768	71.88±04.79	2	71.56±06.19	2	76.42±04.20	691
Heart	13	270	81.85±05.91	2	81.11±06.16	2	84.07±08.38	243

Table 1: Generalization accuracy and model complexity (averaged) for the datasets. The AKGLVQ makes use of the Nyström approximation with 10% of the distances, sparsity has been switched off. The memory used to store the kernel matrix for AKGLVQ is $\approx 95\%$ less then for KGLVQ and a speedup of 2 to 7 could be observed in average. The generalization of AKGLVQ is almost the same like for KGLVQ and is also quite good compared to SVM. #PT refers to the number of prototypes, whereas #SV provides the number of support vectors in the final model.

from both sides are necessary. This leads to a linear runtime complexity of $O(q^3 + qN)$. The full runtime complexity of AKGLVQ is however quadratic because the operations have to be done for all N data points, so we finally get a quadratic setting of $O(N^2)$. A runtime analysis of the *ring* data set with a maximum number of 3000 point is depicted in Figure 3.

By employing the new sparsity measure it is also possible to reduce the complexity of the model and to reduce the amount of memory necessary to store Ψ. The associated parameter β can be estimated by a cross-validation scheme on a sub set of the data using a grid search within a reasonable range of $[0, \ldots, 50]$. In Figure 4 the effect of the sparsity approach with respect to prediction accuracy on a test set and the time complexity is shown. The accuracy and memory complexity is given in % whereas for the time complexity the maximal necessary time is normalized to 1 to allow for better comparison. Using the sparsity constraint and sparse matrices the initial amount of necessary memory is higher than without sparsity due to the overhead caused by the management of sparse matrices. The sparsity constraint is not a hard control parameter for the memory complexity of the model hence it is not possible to provide theoretical guarantees of the memory consumption.

Analyzing Figure 4 we observe that the prediction accuracy is smoothly decreasing with increased β. The optimal β value is around $\beta = 19$ with around 79% accuracy and 40% less consumed memory. An analysis of the other data sets showed that, as ex-

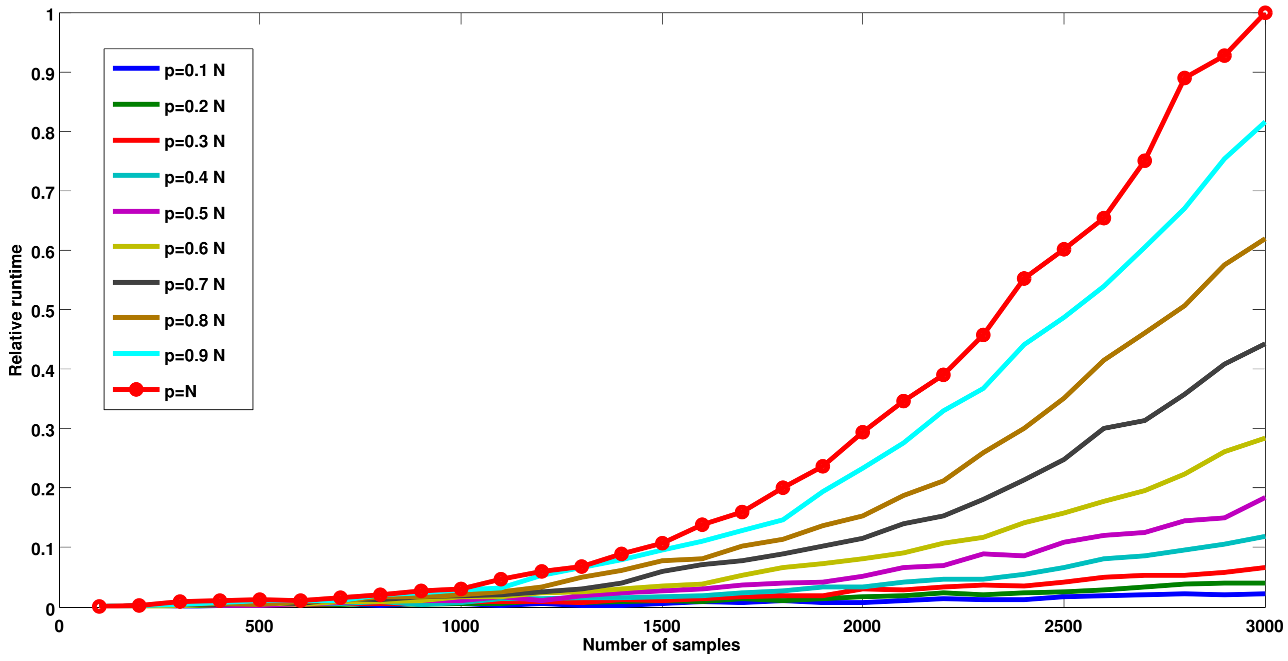

Figure 3: Runtime analysis of AKGLVQ using an extended ring data set for different values of q in the Nyström approximation. The number of samples is changed within $100 - 3000$ on the x-axis, with the relative runtime given on y. The different curves are obtained by changing the Nyström approximation from 10% - 100%. The KGLVQ curve is given with o on the sampling points.

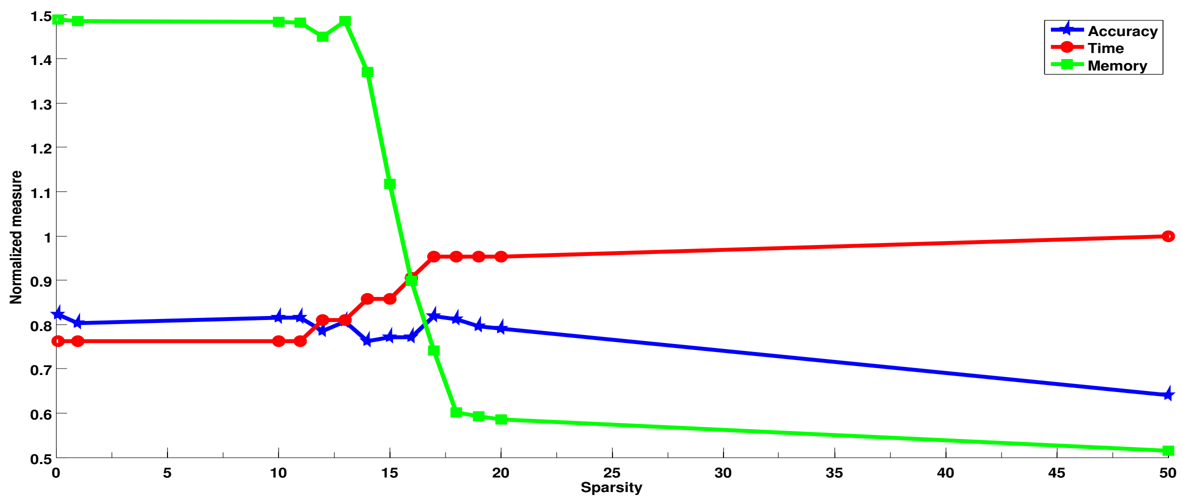

Figure 4: Complexity analysis of AKGLVQ using the sparsity constraint for the heart data set (profiles are similar for the other data sets). The measures are calculated from the observed training model, the memory consumption refers to the matrix Ψ only.

pected, the β parameter is data set specific. It should be optimized based on an independent test set and used to balance accuracy and model complexity.

As an overall observation we found that the runtime of the algorithm is increased by around 20% using the sparsity measure due to additional normalization steps and the effort to manage the sparse matrices. Considering the prior analysis it is obvious that the sparsity constraint can not directly used to speedup the learning time or to reduce (continuously) the consumed memory. Instead it should be used to simplify the final model. This is especially relevant if we focus on interpretable models and a small number of non-vanishing coefficients provides easier access to the interpretation of the prototypes, analyzing the linear combination.

4.4. *Medium sample size data*

In a further study we analyze the accelerated KGLVQ and SVM on medium and large data sets with multiple thousand samples. Thereby we make use of the UCI *spam*-data set[4] which contains measurements to predict if a obtained email is to classify as spam. Further we use the *usps*-data set, containing 16×16 gray-scale images of handwritten digits, provided in[15] and the CMU *faces*-data from the UCI database [4], which contains 640 b/w images of people taken with varying pose, expression, eyes presentation and size. The later two are multiclass data sets. The standard KGLVQ can not any longer be applied for these data under valid settings, without significant sub-sample selections on the training data, which has a negative impact on the results. For the USPS data we took a commonly used subset of 2000 samples randomly sampled. All results are obtained in a 10-fold cross validation using the ELM kernel with 100 cycles for AKGLVQ. Multiclass classification for SVM was done using a 1 vs rest scheme. Results are shown in Table 2.

We observe that the AKGLVQ was quite efficient in modeling the given problems but the prediction performance is significantly lower than the one obtained by SVM also the overall runtime is worse than that of SVM which is typically a magnitude faster than AKGVLQ. However we would like to point out again that our objective is to improve kernel based prototype methods, namely KGLVQ rather to compete directly with SVM. The most interesting property of prototype classifiers may not be the prediction accuracy, although it is quite good in general, but more the interpretability and other aspect as shown in the following.

The higher runtime complexity of AKGVLQ is expected because it is an online learning algorithm in contrast to SVM. AKGLVQ still scales quadratic as pointed out before, if no additional techniques like active learning [30] are employed, which however does not provide guarantees. On the other hand this also allows an easy retraining in case of novel data which is not directly accessible using SVM approaches. Interestingly the quite good prediction results of AKGLVQ are already obtained with very few prototypes leading to compact models. An increase of the number of prototypes up-to a factor of 10 does not change the prediction accuracy significantly. SVM however has used at least 10% of the data or like for the USPS data the whole training data set, making the model very complex.

The KGLVQ and its approximated variant are prototype based methods and the prototypes are constructed by means of a linear combination of the data points in the coefficient matrix Ψ. By analyzing the coefficient matrix Ψ of the models shown in Table 2, it is possible to identify items from the original data set which are considered to be most characteristic and important for the voronoi cell generated by a specific prototype, this is in contrast to SVM models because their model parameters are extreme points rather prototypes. For the USPS dataset which consists of digit images and the faces data set with images of faces it is possible to obtain direct reconstructions of the prototypes which can be easily visualized and interpreted. In Figure 6 the visualizations of the digits $'0','2','8'$ are shown using either the median of the class, the prototype reconstruction or the median support vector reconstruction for support vectors of the corresponding class.

Figure 5 shows different digits of the USPS data set [5]. The Figure has been regenerated as described in [5] for the public available USPS data set using the NeXOM algorithm, which is a specific variant of neighborhood embedding, comparable e.g. to Multi-

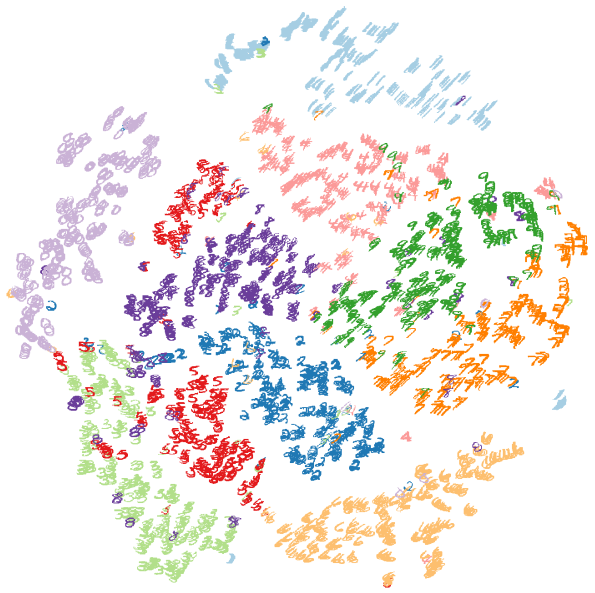

Figure 5: Different USPS symbols in a two dimensional projection using the NeXOM algorithm.

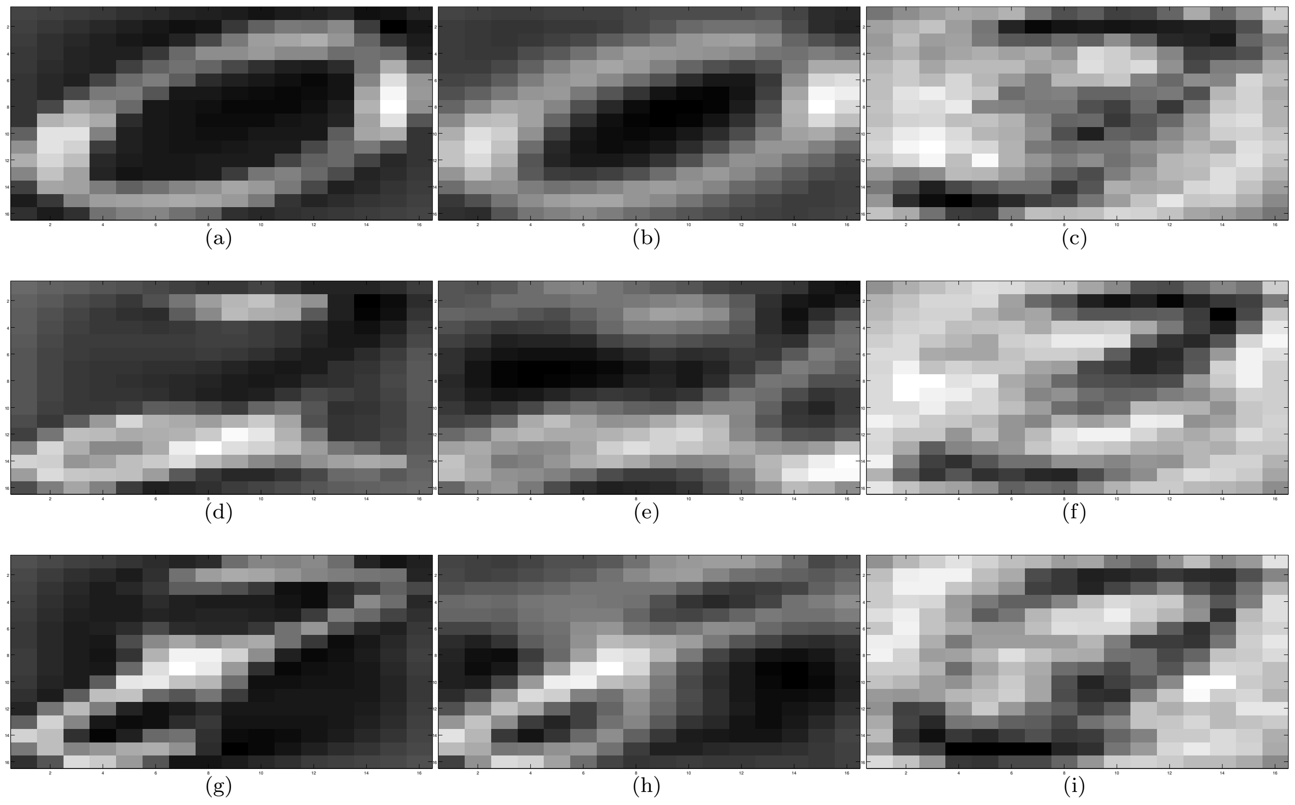

Figure 6: Reconstruction of digit representants. The first row shows reconstructions of the digit ′0′, the second of the digit ′2′ and the last of the digit ′8′. The plot (a) is always the median image of the corresponding class, (b) the learned prototype representation, (c) the support vector reconstruction.

	#C	Dim	Size	AKGLVQ	AKGLVQ-Complexity		SVM	SVM-Complexity	
					ø-T	#PT		ø-T	ø-#SV
Spam	2	57	4601	86.57 ± 02.64	130.73	0.43% (2)	91.92 ± 02.01	00.56	33.13% (469)
USPS	10	256	2000	81.70 ± 01.55	18.95	2.22% (40)	91.35 ± 02.46	00.32	100% (1800)
Faces	20	15360	640	81.09 ± 06.39	01.20	3.47% (20)	94.84 ± 03.83	00.67	10.95% (63)

Table 2: Generalization accuracy and model complexity (averaged) for the small datasets. $\o - T$ refers to the mean runtime in minutes, #PT denotes the number of prototypes and #SV the number of support vectors, respectively. Note that the complexity values are calculated with respect to the training data.

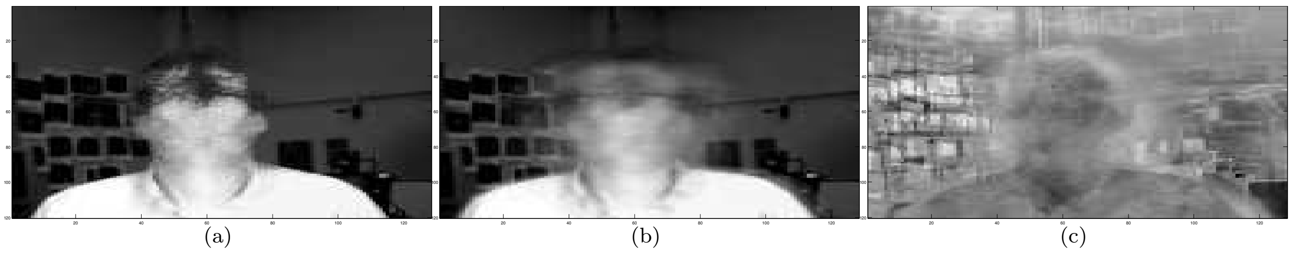

Figure 7: Reconstruction of faces representants for one of the faces classes. The plot (a) is the median image of the corresponding class, (b) the learned prototype representation, (c) the support vector reconstruction.

Dimensional Scaling (MDS)[15][‡] Analyzing the USPS data for the digits $'2'$ and $'8'$ we find that the crossings of the arcs are quite well defined independent of the specific writer but the position of the arcs in the outer regions of the digit differ. This is reflected by the median reconstructions, plots (a) which show holes for these regions. In contrast the learned prototypes for $'2'$ and $'8'$ do not suffer from this error but show a more realistic, prototypical representation of the digit. For the SVM we would expect that the items at the decision borders are pronounced most, such that the most atypical items are represented. Indeed the plots (c) appear to be quite blurred and are hard to interpret. For the digit $'0'$ one observes that the median and prototype reconstruction are almost identical, which is caused by a strong homogeneity in the data. The rare abnormal $'0'$s in the data are either wrong oriented, with an open loop, very tight or almost without the hole, appearing as a bold blot. These examples are selected by the SVM as the model parameters and hence the reconstruction in (c) is hard to interpret as well[§]

For the faces data, exemplary results are shown in Figure 7. The shown person moved the head in the different recordings, so more or less only the background and the body shape are stable. This is reflected by all reconstructions. The median plot (a) shows the raw shape of the person but the face is blurred. The prototype reconstruction (b) is less blurred and reflects also the movement of the head, showing also traces of the turned heads in the image. The prototype is actually so accurate that also the sun-glasses and the lips, nose and ears can be identified. The reconstruction of the SVM (c) is again hard to interpret. The raw shape is preserved but the shown picture is clearly not representative but more a mixture of the abnormal cases in the data, as expected.

Considering the model complexity we find that with very few prototypes for all datasets the AKGLVQ performs quite well. Using the Nyström approximation the memory consumption of the kernel matrix can be substantial reduced such that AKGLVQ becomes an interesting, efficient and prototype based complement to SVM. This is especially interesting if an interpretable prototype of a class is

[‡]The picture is not identical but similar due to random effects in the initialization of NeXOM

[§]The SVM model of the USPS data contains all points, because all α-weights are significant different from 0, but the extreme symbols have the largest α weights.

needed like for image retrieval systems to label the underlying data by a typical image. Also in other cases of interpretable data like clinical recordings, the prototypical model parameters are much easier accessible for the domain expert. It also helps to get a better understanding of the information encoded in the model. If the model fails to classify specific items in the data or assigns them constantly to one wrong class, the prototype can help to interpret the reason for this. In that way the system can be improved by incorporating additional knowledge in an user interactive manner.

Overall we found that AKGLVQ is now capable to learn also very complex data sets with multiple 1000 of items and due to the Nyström approximation and the integrated sparsity constraint the memory complexity of the model is quite low. AKGLVQ allows, like all prototype methods, explicit control over the model complexity by specifying the number of prototypes. For the considered data the prediction accuracy of AKGLVQ was similar to that of KGLVQ but with a significant reduced model complexity and a substantial speed up in the calculations. In comparison to SVM the AKGLVQ models are very compact but were less efficient in prediction for the large data set while comparable effective for the experiments with the more simple data. The obtained prototypes of the KGLVQ- and AKGLVQ-model are much easier to interpret, whereas for SVM the model is less informative.

5. CONCLUSIONS

In this paper we proposed an extended variant of kernelized learning vector quantizer with a significantly reduced model complexity through the integration of the Nyström method and sparse learning. The obtained models use much less memory due to a compact, approximated kernel representation and a sparse coefficient matrix Ψ. Further we compared the efficiency of our new approach with KGLVQ and SVM considering prediction accuracy, model complexity and interpretability. We found that the generalization capability of AKGLVQ is similar to those of KGLVQ and less to SVM. AKGLVQ is much quicker than KGLVQ and needs markable less memory. AKGLVQ and KGLVQ provides interpretable models in contrast to SVM. If not only prediction accuracy but also compactness and interpretability matters AKGLVQ provide an interesting alternative to the considered standard kernel learning methods and is now applicable for medium-sized sets of data, which was not possible before. One very important subject of future works will be to further decrease runtime and memory requirements while parallel increase the prediction efficiency for extremely large data sets.

Acknowledgment: This work was supported by the German Res. Fund. (DFG), HA2719/4-1 (Relevance Learning for Temporal Neural Maps) and by the Cluster of Excellence 277 Cognitive Interaction Technology funded in the framework of the German Excellence Initiative. The authors would like to thank Kerstin Bunte, University of Groningen, The Netherlands, for providing the USPS visualization.

1. Wesam Barbakh and Colin Fyfe. Online clustering algorithms. *Int. J. Neural Syst.*, 18(3):185–194, 2008.
2. P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2003.
3. C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, NY, 2006.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, available at: http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.
5. Kerstin Bunte, Barbara Hammer, Thomas Villmann, Michael Biehl, and Axel Wismüller. Neighbor embedding XOM for dimension reduction and visualization. *Neurocomputing*, 74(9):1340–1350, 2011.
6. Emilio Corchado and Colin Fyfe. Relevance and kernel self-organising maps. In Okyay Kaynak, Ethem Alpaydin, Erkki Oja, and Lei Xu, editors, *ICANN*, volume 2714 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2003.
7. K. Crammer, R. Gilad-Bachrach, A.Navot, and A.Tishby. Margin analysis of the LVQ algorithm. In *Proc. NIPS 2002*, pages 462–469, 2002.
8. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2010.
9. B. Frenay and M. Verleysen. Parameter-free kernel in extreme learning for non-linear support vector regression. *NeuroComputing*, page in press, 2010.
10. Roberto Gil-Pita and Xin Yao. Evolving edited k-nearest neighbor classifiers. *Int. J. Neural Syst.*, 18(6):459–467, 2008.
11. B. Hammer, M. Strickert, and Th. Villmann. Relevance LVQ versus SVM. In L. Rutkowski, J. Siek-

mann, R. Tadeusiewicz, and L.A. Zadeh, editors, *Artificial Intelligence and Soft Computing (ICAISC 2004)*, Lecture Notes in Artificial Intelligence 3070, pages 592–597. Springer Verlag, Berlin-Heidelberg, 2004.

12. B. Hammer and Th. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.
13. B. Hammer and Th. Villmann. Mathematical aspects of neural networks. In M. Verleysen, editor, *Proc. Of European Symposium on Artificial Neural Networks (ESANN'2003)*, pages 59–72, Brussels, Belgium, 2003. d-side.
14. Barbara Hammer, Marc Strickert, and Thomas Villmann. On the generalization ability of GRLVQ networks. *Neural Processing Letters*, 21(2):109–120, 2005.
15. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, New York, 2001.
16. P. Hoyer. Non-negative Matrix Factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.
17. A. K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31:651–666, 2010.
18. Alexander Kaiser, Wolfram Schenck, and Ralf Möller. Coupled singular value decomposition of a cross-covariance matrix. *Int. J. Neural Syst.*, 20(4):293–318, 2010.
19. S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.
20. Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences.* Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).
21. K. Labusch, E. Barth, and T. Martinetz. Learning data representations with sparse coding neural gas. In *Proc. of ESANN'08*, pages 233–238, 2008.
22. Mu. Li, J.T. Kwok, and B.-L. Lu. Making large-scale Nyström approximation possible. In *Proceedings of the International Conference on Machine Learning (ICML)'2010*, 2009.
23. Ezequiel López-Rubio, Rafael Marcos Luque Baena, and Enrique Domínguez. Foreground detection in video sequences with probabilistic self-organizing maps. *Int. J. Neural Syst.*, 21(3):225–246, 2011.
24. Joshua E. Menke and Tony R. Martinez. Improving supervised learning by adapting the problem to the learner. *Int. J. Neural Syst.*, 19(1):1–9, 2009.
25. Britta Mersch, Tobias Glasmachers, Peter Meinicke, and Christian Igel. Evolutionary optimization of sequence kernels for detection of bacterial gene starts. *Int. J. Neural Syst.*, 17(5):369–381, 2007.
26. B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Letters to Nature*, 381:607–609, 1996.
27. John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208. MIT Press, Cambridge, MA, USA, 1999.
28. A. K. Qin and P. N. Suganthan. A novel kernel prototype-based learning algorithm. In *Proc. of ICPR'04*, pages 2621–624, 2004.
29. A. S. Sato and K. Yamada. Generalized learning vector quantization. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 423–429. MIT Press, 1995.
30. F.-M. Schleif, B. Hammer, and Th. Villmann. Margin based active learning for LVQ networks. *Neurocomputing*, 70(7-9):1215–1224, 2007.
31. F.-M. Schleif, T. Villmann, B. Hammer, P. Schneider, and M. Biehl. Generalized derivative based kernelized learning vector quantization. In *Proceedings of IDEAL 2010*, pages 21–28, 2010.
32. P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21:3532–3561, 2009.
33. B. Schoelkopf and A. Smola. *Learning with Kernels.* MIT Press, 2002.
34. S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2003.
35. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis and Discovery.* Cambridge University Press, 2004.
36. Liangdong Shi, Yinghuan Shi, Yang Gao, Lin Shang, and Yubin Yang. XCSC: a novel approach to clustering with extended classifier system. *Int. J. Neural Syst.*, 21(1):79–93, 2011.
37. Ivor Wai-Hung Tsang, András Kocsor, and James Tin-Yau Kwok. Large-scale maximum margin discriminant analysis using core vector machines. *IEEE Transactions on Neural Networks*, 19(4):610–624, 2008.
38. V Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
39. C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. 2001.
40. Yang Yang and Bao-Liang Lu. Protein subcellular multi-localization prediction using a min-max modular support vector machine. *Int. J. Neural Syst.*, 20(1):13–28, 2010.
41. K. Zhang, I.W. Tsang, and J.T. Kwok. Improved Nyström low-rank approximation and error analysis o. In *Proceedings of the International Conference on Machine Learning (ICML)'2010*, 2009.