Adaptive relevance matrices in Learning Vector Quantization

Petra Schneider[1], Michael Biehl[1], Barbara Hammer[2]

[1]Institute for Mathematics and Computing Science, University of Groningen
P.O. Box 407, 9700 AK Groningen, The Netherlands
{petra,biehl}@cs.rug.nl

[2]Institute of Computing Science, Clausthal University of Technology
Julius Albert Strasse 4, 38678 Clausthal - Zellerfeld, Germany
hammer@in.tu-clausthal.de

Abstract

We propose a new matrix learning scheme to extend relevance learning vector quantization (RLVQ), an efficient prototype-based classification algorithm, towards a general adaptive metric. By introducing a full matrix of relevance factors in the distance measure, correlations between different features and their importance for the classification scheme can be taken into account and automated, general metric adaptation takes place during training. In comparison to the weighted Euclidean metric used in RLVQ and its variations, a full matrix is more powerful to represent the internal structure of the data appropriately. Large margin generalization bounds can be transfered to this case leading to bounds which are independent of the input dimensionality. This also holds for local metrics attached to each prototype which corresponds to piecewise quadratic decision boundaries. The algorithm is tested in comparison to alternative LVQ schemes using an artificial data set, a benchmark multi-class problem from the UCI repository, and a problem from bioinformatics, the recognition of splice sites for *C. elegans*.

Keywords: learning vector quantization, generalized LVQ, metric adaptation, generalization bounds

1 Introduction

Learning vector quantization (LVQ) as introduced by Kohonen is a particularly intuitive and simple though powerful classification scheme [16] which is very appealing for several reasons: The method is easy to implement; the complexity of the resulting classifier can be controlled by the user; the classifier can naturally deal with multi-class problems; and, unlike many alternative classification schemes such as feedforward networks or the support vector machine (SVM),

the LVQ system is straightforward to interpret because of the intuitive assignment of data to the class of the closest prototype. For these reasons, LVQ has been used in a variety of academic and commercial applications such as image analysis, bioinformatics, telecommunication, robotics, etc. [4].
Original LVQ, however, suffers from several drawbacks such as potentially slow convergence and instable behavior because of which various alternatives have been proposed, see for instance [16]. Still, there are two major drawbacks of these methods, which have been tackled only recently.
On the one hand, LVQ relies on heuristics and a full mathematical investigation of the algorithm is lacking. This problem relates to unexpected behavior and instabilities of training. It has been shown that already slight variations of the basic LVQ learning scheme yield quite different results [2, 3]. Variants of LVQ which can be derived from an explicit cost function are particularly interesting. Several proposals for cost functions can be found in the literature, one example being generalized LVQ (GLVQ) which forms the basis for the method we will consider in this article [21]. Two alternatives which implement soft relaxations of the original learning rule are presented in [22, 23]. These two approaches, however, have the drawback that the original crisp limit case does not exist (for [22]) resp. the discrete limit case shows poor results also in simple settings, see [9] (for [23]). The cost function as proposed in [21] displays stable behavior and aims at a good generalization ability already during training as pointed out in [12]. On the other hand, LVQ and variants severely rely on the standard Euclidean metric and they are not appropriate for situations where the Euclidean metric does not represent the underlying semantic. This is the case, e.g., for high dimensional data where noise accumulates and likely disrupts the classification, for heterogeneous data where the importance and nature of the dimensions differs, and for data which involves correlations of the dimensions. In these cases, which are quite common in practice, simple LVQ may fail. Recently, a cost function based generalization of LVQ has been proposed which allows to incorporate general differentiable similarity measures [13]. The specific choice of the similarity measure as a simple weighted diagonal metric with adaptive relevance terms has turned out particularly suitable in many practical applications since it can account for irrelevant or inadequately scaled dimensions. At the same time, it allows for straightforward interpretation of the result because the relevance profile can directly be interpreted as the contribution of the dimensions to the classification [15]. For an adaptive diagonal metric, dimensionality independent large margin generalization bounds can be derived [12]. This fact is remarkable since it accompanies the good experimental classification results for high dimensional data by a theoretical counterpart. The same bounds also hold for kernelized versions, but not for arbitrary choices of the metric. Often, different features are correlated in classification tasks. In unsupervised clustering, correlations of data are accounted for, e.g., by the classical Mahalanobis distance [7] or fuzzy-covariance matrices as proposed, e.g., in the approaches [8, 10]. It has been shown recently that general metric learning based on large margin principles can greatly improve the results obtained by distance-based schemes such as k-nearest neighbor classifier [24, 25]. For supervised LVQ classification tasks,

however, an explicit metric which takes into account correlations has not yet been proposed. Based on the general framework as presented in [13], we develop an extension of LVQ to an adaptive matrix of relevances which parameterizes a general Euclidean metric and accounts for pairwise correlations of features. By means of an implicit scaling and rotation of the data, the algorithm yields a discriminative distance measure which is particularly suitable for the given classification task. It can be parameterized in terms of a single, global matrix or by individual matrices attached to the prototypes. Interestingly, one can derive generalization bounds which are similar to the case of a simple diagonal metric for this more complex case. Apart from this theoretical guarantee, we demonstrate the usefulness of the novel scheme in the context of several classification problems.

2 Review of LVQ

LVQ aims at approximating a classification scheme by prototypes. Assume training data $(\boldsymbol{\xi}_i, y_i) \in \mathbb{R}^N \times \{1, \ldots, C\}$ are given, N denoting the data dimensionality and C the number of different classes. An LVQ network consists of a number of prototypes which are characterized by their location in the weight space $\boldsymbol{w}_i \in \mathbb{R}^N$ and their class label $c(\boldsymbol{w}_i) \in \{1, \ldots, C\}$. Classification is implemented as a winner takes all scheme. For this purpose, a possibly parameterized similarity measure $d^{\boldsymbol{\lambda}}$ is fixed for $\mathbb{R}^N$, where $\boldsymbol{\lambda}$ specifies the metric parameters which can be adapted during training. Often, the standard Euclidean metric is chosen. A data point $\boldsymbol{\xi} \in \mathbb{R}^N$ is mapped to the class label $c(\boldsymbol{\xi}) = c(\boldsymbol{w}_i)$ of the prototype i for which $d^{\boldsymbol{\lambda}}(\boldsymbol{w}_i, \boldsymbol{\xi}) \leq d^{\boldsymbol{\lambda}}(\boldsymbol{w}_j, \boldsymbol{\xi})$ holds for every $j \neq i$, breaking ties arbitrarily. Hence, it is mapped to the class of the closest prototype, the so-called winner.

Learning aims at determining weight locations for the prototypes such that the given training data are mapped to their corresponding class labels. This is usually achieved by a modification of Hebbian learning, which moves prototypes closer to the data points of their respective class. A very flexible learning approach has been introduced in [13]: Training is derived as a minimization of the cost function

$$\sum_i \Phi(\mu_i) \text{ where } \quad \mu_i = \frac{d_J^{\boldsymbol{\lambda}} - d_K^{\boldsymbol{\lambda}}}{d_J^{\boldsymbol{\lambda}} + d_K^{\boldsymbol{\lambda}}} \tag{1}$$

based on the steepest descent method. Φ is a monotonic function, e.g. the identity or the logistic function, $d_J^{\boldsymbol{\lambda}} = d^{\boldsymbol{\lambda}}(\boldsymbol{w}_J, \boldsymbol{\xi}_i)$ is the distance of data point $\boldsymbol{\xi}_i$ from the closest prototype $\boldsymbol{w}_J$ with the same class label y_i, and $d_K^{\boldsymbol{\lambda}} = d^{\boldsymbol{\lambda}}(\boldsymbol{w}_K, \boldsymbol{\xi}_i)$ is the distance from the closest prototype $\boldsymbol{w}_K$ with a different class label than y_i. Note that the numerator is smaller than 0 iff the classification of the data point is correct. The smaller the numerator, the greater the security of classification, i.e. the difference of the distance from a correct and wrong prototype. The denominator scales the argument of Φ such that it satisfies $-1 < \mu(\boldsymbol{\xi}) < 1$. A further possibly nonlinear scaling by Φ might be beneficial for applications.

This formulation can be seen as a kernelized version of so-called generalized LVQ introduced in [21].
The learning rule can be derived from this cost function by taking derivatives. We assume that the similarity measure $d^{\lambda}(\boldsymbol{w}, \boldsymbol{\xi})$ is differentiable with respect to the parameters $\boldsymbol{w}$ and $\boldsymbol{\lambda}$. As shown in [13], for a given pattern $\boldsymbol{\xi}$ the derivatives yield

$$\Delta \boldsymbol{w}_J \;=\; -\epsilon \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^{+}(\boldsymbol{\xi}) \cdot \boldsymbol{\nabla}_{\boldsymbol{w}_J} d_J^{\lambda} \tag{2}$$

where $\epsilon > 0$ is the learning rate, the derivative of Φ is taken at position $\mu(\boldsymbol{\xi})$, and $\mu^{+}(\boldsymbol{\xi}) = 2 \cdot d_K^{\lambda} / (d_J^{\lambda} + d_K^{\lambda})^2$. Further,

$$\Delta \boldsymbol{w}_K \;=\; \epsilon \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \mu^{-}(\boldsymbol{\xi}) \cdot \boldsymbol{\nabla}_{\boldsymbol{w}_K} d_K^{\lambda} \tag{3}$$

where $\mu^{-}(\boldsymbol{\xi}) = 2 \cdot d_J^{\lambda} / (d_J^{\lambda} + d_K^{\lambda})^2$. The derivative with respect to the parameters $\boldsymbol{\lambda}$ yields the update

$$\Delta \boldsymbol{\lambda} \;=\; \epsilon \cdot \Phi'(\mu(\boldsymbol{\xi})) \cdot \left(\mu^{+}(\boldsymbol{\xi}) \cdot \boldsymbol{\nabla}_{\lambda} d_J^{\lambda} - \mu^{-}(\boldsymbol{\xi}) \cdot \boldsymbol{\nabla}_{\lambda} d_K^{\lambda} \right) . \tag{4}$$

The adaptation of $\boldsymbol{\lambda}$ is often followed by normalization during training, e.g. enforcing $\sum_i \lambda_i = 1$ to prevent degeneration of the metric. It has been shown in [13] that these update rules are valid whenever the metric is differentiable. This argument also holds for the borders of receptive fields, i.e. an underlying continuous input distribution, as can be shown using delta-functions [13].
It has been demonstrated in [13] that the squared weighted Euclidean metric $d^{\lambda}(\boldsymbol{w}, \boldsymbol{\xi}) = \sum_i \lambda_i (w_i - \xi_i)^2$ with $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$ is a simple and powerful choice which allows to use prototype based learning also in the presence of high dimensional data with features of different, yet a priori unknown, relevance. This measure has the advantage that the relevance factors λ_i can be interpreted directly and provide insight into the classification task: Dimensions with large λ_i are most important for the classification while very small or zero relevances indicate that the corresponding feature could be omitted. We refer to this method as generalized relevance learning vector quantization (GRLVQ) [15]. Very similar schemes have been motivated heuristically which lack an interpretation of the algorithm as stochastic gradient descent with respect to a cost function [5]. Alternative choices have been introduced in [13], including, for example, metrics which take local windows into account, e.g., for time series data.
Note that the relevance factors, i.e. the choice of the metric need not be global, but can be attached to single prototypes, locally. In this case, individual updates take place for the relevance factors λ^j for each prototype j, and the distance of a data point $\boldsymbol{\xi}_i$ from prototype $\boldsymbol{w}_j$, $d^{\lambda^j}(\boldsymbol{w}_j, \boldsymbol{\xi}_i)$ is computed based on λ^j. This allows a local relevance adaptation, taking into account that the relevance might change within the data space. This method has been investigated e.g. in [11]. We refer to this version as localized GRLVQ (LGRLVQ).

3 The GMLVQ Algorithm

Here, we introduce an important extension of the above concept, which employs a full matrix of relevances in the similarity measure. We consider a generalized distance of the form

$$d^{\Lambda}(\boldsymbol{w}, \boldsymbol{\xi}) \;=\; (\boldsymbol{\xi} - \boldsymbol{w})^T \, \Lambda \, (\boldsymbol{\xi} - \boldsymbol{w}) \tag{5}$$

where Λ is a full $N \times N$ matrix which can account for correlations between the features. Note that, this way, arbitrary Euclidean metrics can be realized by an appropriate choice of the parameters. In particular, correlations of dimensions and rotation of the axes can be accounted for. Such choices have already successfully been introduced in unsupervised clustering methods such as fuzzy clustering [8, 10], however, at the expense of increased computational costs, since these methods require a matrix inversion at each adaptation step. For the metric as introduced above, a variant which costs $\mathcal{O}(N^2)$ can be derived.
Note that the above similarity measure defines a general squared Euclidean distance in an appropriately transformed space only if Λ is positive (semi-) definite. We can achieve this by substituting

$$\Lambda \;=\; \Omega \, \Omega^T \tag{6}$$

which yields $\boldsymbol{u}^T \Lambda \boldsymbol{u} = \boldsymbol{u}^T \Omega \, \Omega^T \boldsymbol{u} = \left(\Omega^T u\right)^2 \geq 0$ for all $\boldsymbol{u}$, where Ω is an arbitrary real $N \times N$ matrix. Every positive semidefinite symmetric matrix can be written as the square of some matrix Ω, which is not uniquely defined, in general. As Λ is symmetric, it has only $N(N+1)/2$ independent entries. We can therefore assume without loss of generality that Ω itself is a symmetric $N \times N$ matrix with $\Omega^T = \Omega$, i.e. $\Lambda = \Omega\Omega$ in the following. This corresponds to the unique positive root of the matrix Λ. Assuming symmetry of Ω reduces the computational costs by a factor 2.
Now the squared distance reads

$$d^{\Lambda}(\boldsymbol{w}, \boldsymbol{\xi}) \;=\; \sum_{i,j,k} (\xi_i - w_i) \, \Omega_{ik} \Omega_{kj} \, (\xi_j - w_j). \tag{7}$$

To obtain the adaptation formulas we need to compute the derivatives with respect to $\boldsymbol{w}$ and Ω. The derivative of d^{Λ} with respect to $\boldsymbol{w}$ yields

$$\boldsymbol{\nabla}_w \, d^{\Lambda} \;=\; -2\Lambda \, (\boldsymbol{\xi} - \boldsymbol{w}) \;=\; -2\Omega\Omega(\boldsymbol{\xi} - \boldsymbol{w}). \tag{8}$$

Derivatives with respect to a single element Ω_{lm} give

$$\begin{aligned} \frac{\partial d^{\Lambda}}{\partial \Omega_{lm}} \;&=\; \sum_j (\xi_l - w_l) \Omega_{mj} (\xi_j - w_j) + \sum_i (\xi_i - w_i) \Omega_{il} (\xi_m - w_m) \\ &=\; (\xi_l - w_l) \, [\Omega(\boldsymbol{\xi} - \boldsymbol{w})]_m + (\xi_m - w_m) \; [\Omega(\boldsymbol{\xi} - \boldsymbol{w})]_l \end{aligned} \tag{9}$$

where subscripts l, m specify components of vectors. Thus, we get the update equations

$$\begin{aligned} \Delta \boldsymbol{w}_J \;&=\epsilon \cdot \phi'(\mu(\boldsymbol{\xi})) \cdot \mu^+(\boldsymbol{\xi}) \cdot \Omega\Omega \cdot (\boldsymbol{\xi} - \boldsymbol{w}_J) \\ \Delta \boldsymbol{w}_K \;&=\; -\,\epsilon \cdot \phi'(\mu(\boldsymbol{\xi})) \cdot \mu^-(\boldsymbol{\xi}) \cdot \Omega\Omega \cdot (\boldsymbol{\xi} - \boldsymbol{w}_K)\,. \end{aligned} \tag{10}$$

Note that these updates correspond to the standard Hebb terms of LVQ, pushing the closest correct prototype towards the considered data point and the closest wrong prototype away from the considered data point. For the update of the matrix elements Ω_{lm} we get

$$\begin{aligned}\Delta\Omega_{lm} \;=\; & -\epsilon\cdot\phi'(\mu(\boldsymbol{\xi}))\cdot \\ & \Big(\mu^{+}(\boldsymbol{\xi})\cdot\Big([\Omega(\boldsymbol{\xi}-\boldsymbol{w}_J)]_m(\xi_l-w_{J,l})+[\Omega(\boldsymbol{\xi}-\boldsymbol{w}_J)]_l(\xi_m-w_{J,m})\Big)- \\ & \mu^{-}(\boldsymbol{\xi})\cdot\Big([\Omega(\boldsymbol{\xi}-\boldsymbol{w}_K)]_m(\xi_l-w_{K,l})+[\Omega(\boldsymbol{\xi}-\boldsymbol{w}_K)]_l(\xi_m-w_{K,m})\Big)\Big)\,.\end{aligned} \tag{11}$$

This update also corresponds to a Hebbian term, since the driving force consists of the derivative of the distance from the closest correct prototype (scaled with -1) and the closest incorrect prototype. Thus, the parameters of the matrix are changed in such a way that the distance from the closest correct prototype becomes smaller, whereas the distance from the closest wrong prototype is increased. Similar to the case of diagonal relevances [5], matrix updates can be formulated on heuristic grounds, as well. Such variants will be discussed in forthcoming publications.

The learning rate for the metric can be chosen independently of the learning rate of the prototypes. We set it an order of magnitude smaller in order to achieve a slower time-scale of metric learning compared to the weight updates. Note that the above update (11) preserves the assumed symmetry of Ω as it is symmetric w.r.t. exchange of indices l and m. After each update Λ should be normalized to prevent the algorithm from degeneration. One possibility is to enforce

$$\sum_i \Lambda_{ii} = 1 \tag{12}$$

by dividing all elements of Λ by the *raw* value of $\sum_i \Lambda_{ii}$ after each step. In this way we fix the sum of diagonal elements which coincides with the sum of eigenvalues. This generalizes the normalization of relevances $\sum_i \lambda_i = 1$ for a simple diagonal metric. One can interpret the eigendirections of Λ as the temporary coordinate system with relevances Λ_{ii}.

Note that

$$\Lambda_{ii} = \sum_k \Omega_{ik}\Omega_{ki} \;= \sum_k \left(\Omega_{ik}\right)^2 . \tag{13}$$

So normalization can be done by dividing all elements of Ω by $(\sum_{ik}(\Omega_{ik})^2)^{1/2} = (\sum_i[\Omega\Omega]_{ii})^{1/2}$ after every update step.

We term this learning rule generalized matrix LVQ (GMLVQ). The complexity of one adaptation step is determined by the computation of the closest correct and incorrect prototypes ($\mathcal{O}(N^2\cdot P)$, P being the number of prototypes), and the adaptation ($\mathcal{O}(N^2)$). Usually, this procedure is repeated a number of time steps which is linear in the number of patterns to achieve convergence. Thus, this procedure is faster than unsupervised fuzzy-clustering variants which use

a similar form of the metric but which require a matrix inversion in each step. Apart from this improved efficiency, the metric is determined in a supervised way in this approach, such that the parameters are optimized with respect to the given classification task.
We can work with one full matrix which accounts for a transformation of the whole input space, or, alternatively, with local matrices attached to the individual prototypes. In the latter case, the squared distance of data point $\boldsymbol{\xi}$ from a prototype $\boldsymbol{w}_j$ is computed as $d^{\Lambda^j}(\boldsymbol{w}_j, \boldsymbol{\xi}) = (\boldsymbol{\xi} - \boldsymbol{w}_j)^T \Lambda^j (\boldsymbol{\xi} - \boldsymbol{w}_j)$. Each matrix is adapted individually in the following way: Given $\boldsymbol{\xi}$ with closest correct prototype $\boldsymbol{w}_J$ and closest incorrect prototype $\boldsymbol{w}_K$, we get the update equations

$$\begin{aligned} \Delta\Omega^J_{lm} \; = \; & -\epsilon \cdot \phi'(\mu(\boldsymbol{\xi})) \cdot \\ & \mu^+(\boldsymbol{\xi}) \cdot \Big([\Omega^J(\boldsymbol{\xi} - \boldsymbol{w}_J)]_m (\xi_l - w_{J,l}) + [\Omega^J(\boldsymbol{\xi} - \boldsymbol{w}_J)]_l (\xi_m - w_{J,m}) \Big) \end{aligned} \tag{14}$$

$$\begin{aligned} \Delta\Omega^K_{lm} \; = \; & +\epsilon \cdot \phi'(\mu(\boldsymbol{\xi})) \cdot \\ & \mu^-(\boldsymbol{\xi}) \cdot \Big([\Omega^K(\boldsymbol{\xi} - \boldsymbol{w}_K)]_m (\xi_l - w_{K,l}) + [\Omega^K(\boldsymbol{\xi} - \boldsymbol{w}_K)]_l (\xi_m - w_{K,m}) \Big) . \end{aligned} \tag{15}$$

Localized matrices have the potential to take into account correlations which can vary between different classes or regions in feature space. For instance, clusters with ellipsoidal shape and different orientation could be present in the data.
Note that LGMLVQ leads to nonlinear decision boundaries which are composed of quadratic pieces, unlike GMLVQ which is characterized by piecewise linear decision boundaries. This way, the receptive fields of the prototypes need no longer be convex or even connected for LGMLVQ, as we will see in the experiments. Depending on the data at hand, this effect can largely increase the capacity of the system.

4 Generalization ability

One of the benefits of prototype-based learning algorithms consists in the fact that they show very good generalization ability also for high dimensional data. This observation can be accompanied by theoretical guarantees. It has been proved in [6] that basic LVQ networks equipped with the Euclidean metric possess dimensionality independent large-margin generalization bounds, whereby the margin refers to the security of the classification, i.e. the distance of a given data point to the classification boundary. A similar result has been derived in [12] for LVQ networks as considered above which possess an adaptive diagonal metric. Remarkably, the margin is directly correlated to the numerator of the cost function as introduced above, i.e. these learning algorithms inherently aim at margin optimization during training. As pointed out in [13], these results transfer immediately to kernelized versions of the algorithm where the similarity measure can be interpreted as the composition of the standard scaled Euclidean metric and a fixed kernel map. In the case of an adaptive full matrix, however,

these results are not applicable, because the matrix is changed during training. Hence a large number of additional free parameters is introduced since the kernel is optimized according to the given classification task.
Here, we directly derive a large margin generalization bound for LGMLVQ networks with a full adaptive matrix attached to every prototype, whereby we use the ideas of [12]. Thus, despite the large flexibility of LGMLVQ networks, excellent generalization ability of trained networks can be expected due to an inherent regularization of the models. Since the function class implemented by GMLVQ networks is contained in the class of LGMLVQ networks, the bounds also hold for the simpler case.
We consider a LGMLVQ network given by P prototypes $\boldsymbol{w}_i$. We assume that all inputs $\boldsymbol{\xi}$ fulfill the condition $|\boldsymbol{\xi}| \leq B$ for some $B > 0$, and we assume that weights are also restricted by $|\boldsymbol{w}_i| \leq B$. As beforehand, we assume that Λ^i is a symmetric positive semidefinite matrix such that the trace is normalized to 1. We consider the case of a binary classification, i.e. only two classes, -1 and 1, are present. We refer to prototypes labeled with $S \in \{+, -\}$ by $\boldsymbol{w}_i^S$.
Classification takes place by a winner takes all rule, i.e.

$$\boldsymbol{\xi} \mapsto \operatorname{sgn}\left(\min_{\boldsymbol{w}_i^-}\{d^{\Lambda^i}(\boldsymbol{w}_i^-, \boldsymbol{\xi})\} - \min_{\boldsymbol{w}_j^+}\{d^{\Lambda^j}(\boldsymbol{w}_j^+, \boldsymbol{\xi})\}\right) \tag{16}$$

where $d^{\Lambda^i}(\boldsymbol{w}_i, \boldsymbol{\xi}) = (\boldsymbol{\xi} - \boldsymbol{w}_i)^T \Lambda^i (\boldsymbol{\xi} - \boldsymbol{w}_i)$ as beforehand and sgn selects the sign ± 1 of the real number. A trainable LGMLVQ network corresponds to a function in the class

$$\begin{aligned}\mathcal{F} := \{f : \mathbb{R}^N \to \{-1, 1\} \mid \exists |\boldsymbol{w}_i| \leq B, \exists \Lambda^i \text{ such that } \Lambda^i \text{ is} \\ \text{symmetric and positive semidefinite with trace 1,} \\ \text{such that } f \text{ is given by (16)}\}\end{aligned} \tag{17}$$

Assume some unknown underlying probability measure P is given on $\mathbb{R}^N \times \{-1, 1\}$ according to which training examples are drawn. The goal of learning is to find a function $f \in \mathcal{F}$ such that the generalization error

$$E_P(f) := P(y \neq f(\boldsymbol{\xi})) \tag{18}$$

is as small as possible. However, P is not known during training; instead, examples for the distribution $(\boldsymbol{\xi}_i, y_i)$, $i = 1, \ldots, m$ are available, which are independent and identically distributed according to P. Training aims at minimizing the empirical error on the given training data

$$\hat{E}_m(f) := \sum_{i=1}^{m} |\{y_i \neq f(\boldsymbol{\xi}_i)\}|/m\,. \tag{19}$$

Thus, the learning algorithm generalizes to unseen data if $\hat{E}_m(f)$ becomes representative for $E_P(f)$ for an increasing number of examples m with high probability, i.e. if we can automatically guarantee a small error on *any* possible input to the learned function, given the *trained* inputs are correct. This bound should

hold simultaneously for any function f of the class, in particular for the network trained according to the given sample set.
We will not derive bounds which are directly based on the empirical error $\hat{E}_m(f)$, rather, we incorporate the security of a classification in terms of the classification margin. For a function f as given by (16), we consider the related real-valued function

$$M_f : \boldsymbol{\xi} \mapsto \left(\min_{\boldsymbol{w}_i^-} \{d^{\Lambda^i}(\boldsymbol{w}_i^-, \boldsymbol{\xi})\} - \min_{\boldsymbol{w}_j^+} \{d^{\Lambda^j}(\boldsymbol{w}_j^+, \boldsymbol{\xi})\} \right) \tag{20}$$

which is obtained by dropping the function sgn. The sign of this real value determines the output class and the size of its absolute value indicates the security of the classification, i.e. the margin of the classifier with respect to input $\boldsymbol{\xi}$ around the decision boundary. The larger this margin, the more robust is the classification of $\boldsymbol{\xi}$ with respect to noise in the input or function parameters. We refer to the resulting class of real-valued functions implemented by LGMLVQ networks by $M_{\mathcal{F}}$.
Assume $\rho > 0$ estimates the minimum security of the classification. For the moment, we assume that $\rho \in (0, 1)$ is a fixed number which is chosen a priori independent of the training set and the final LGMLVQ function. Following the approach [1], we define the loss function

$$L : \mathbb{R} \to \mathbb{R}, t \mapsto \begin{cases} 1 & \text{if } t \leq 0 \\ 1 - t/\rho & \text{if } 0 < t \leq \rho \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

The term

$$\hat{E}_m^L(f) := \sum_{i=1}^{m} L(y_i \cdot M_f(\boldsymbol{\xi}_i))/m \tag{22}$$

accumulates the number of errors for a given data set, and, in addition, also punishes all correct classifications if their margin is smaller than ρ, i.e. it measures the classification accuracy and its robustness with respect to noise. The term $\hat{E}_m^L(f)$ is small iff the number of misclassifications is small and almost all correctly classified points have margin larger than ρ.
It is possible to correlate the generalization error of LGMLVQ networks and this modified empirical error by a dimensionality independent bound: According to [1](Theorem 7) the inequality

$$E_P(f) \leq \hat{E}_m^L(f) + \frac{2}{\rho} \cdot R_m(M_{\mathcal{F}}) + \sqrt{\frac{\ln(4/\delta)}{2m}} \tag{23}$$

holds simultaneously for all functions in $M_{\mathcal{F}}$ with probability at least $1 - \delta/2$, whereby $R_m(M_{\mathcal{F}})$ is the so-called Rademacher complexity of the function class $M_{\mathcal{F}}$.
The Rademacher complexity of a function class measures the complexity of the class by considering the correlation of outputs of a function of the class on a

given set of points and random variables. The empirical Rademacher complexity of the function class $M_{\mathcal{F}}$ given m samples $\boldsymbol{\xi}_i$, is defined as the expectation

$$\hat{R}_m(M_{\mathcal{F}}) = E_{\sigma_1,\ldots,\sigma_m}\left(\sup_{M_f \in M_{\mathcal{F}}} \left| \frac{2}{m} \sum_{i=1}^{m} \sigma_i \cdot M_f(\boldsymbol{\xi}_i) \right| \right) \tag{24}$$

where σ_i are independent $\{-1, 1\}$-valued random variables with zero mean. The Rademacher complexity is defined as the expectation over the samples

$$R_m(M_{\mathcal{F}}) = E_{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_m} \hat{R}_m(M_{\mathcal{F}}) \tag{25}$$

where $\boldsymbol{\xi}_i$ are independent and identically distributed according to the marginal distribution of P on the input space.
Using techniques of [1], we will show in the appendix, that the Rademacher complexity of functions given by a LGMLVQ architecture, $R_m(M_{\mathcal{F}})$, can be limited by the term

$$\mathcal{O}\left(\frac{P^2 B^3 + \sqrt{\ln(1/\delta)}}{\sqrt{m}}\right) \tag{26}$$

with probability at least $1 - \delta/2$ where P denotes the number of prototypes and B the size restriction of inputs and prototypes. Thus, the overall inequality

$$E_P(f) \leq \hat{E}_m^L(f) + \frac{1}{\sqrt{m}} \mathcal{O}\left(\frac{P^2 B^3}{\rho} + \frac{\sqrt{\ln(1/\delta)}}{\min\{1, \rho\}}\right) \tag{27}$$

results which holds simultaneously for all $M_f \in M_{\mathcal{F}}$ and training data with probability at least $1 - \delta$. Since this is valid for all $M_f \in M_{\mathcal{F}}$, matrix parameters as well as prototypes can be adaptive. This bound allows to estimate the deviation of the generalization error of a LGMLVQ network and its result on a given training set. Obviously, the bound is small for a small number of errors and a large margin ρ for almost all correctly classified points. Note that the cost function of LGMLVQ is correlated to the classification error, but it also contains the margin of a data point as denominator of the summands. Thus, LGMLVQ aims at an optimization of the margin during training and at a corresponding simplification of the classifier, such that good generalization can be expected. Note that this bound is independent of the dimensionality of the data. Thus, excellent generalization can be expected also for high dimensional settings.
So far, we assumed that the bound ρ for the margin (points with smaller margin contribute to the error) is fixed a priori. In applications, it is reasonable to choose ρ based on the outcome of a training algorithm such that almost all training examples have a margin larger than ρ. For this case, a generalization of the argumentation is possible which only assumes some prior about a reasonable range of the margin: Assume the empirical margin can be upper bounded by $C > 0$, a naive bound being e.g. the maximum distance of data in the given training set. We define $\rho_i = C/i$ for $i \geq 1$, and we choose prior probabilities $p_i \geq 0$ with $\sum p_i = 1$ which indicate the confidence in achieving an empirical

margin of size at least ρ_i for almost all training data. We define the cost function L_i as above associated to margin ρ_i and the corresponding empirical error as $\hat{E}_m^{L_i}(f)$. We are interested in the probability

$$P\left(\exists i\, E_P(f) \geq \hat{E}_m^{L_i}(f) + \epsilon(i)\right) \tag{28}$$

where the bound

$$\epsilon(i) = \frac{1}{\sqrt{m}}\mathcal{O}\left(\frac{P^2B^3}{\rho} + \frac{\sqrt{\ln\left(1/(p_i\delta)\right)}}{\min\{1,\rho\}}\right) \tag{29}$$

is chosen according to the inequality (27). We can argue

$$\begin{aligned} P\left(\exists i\, E_P(f) \geq \hat{E}_m^{L_i}(f) + \epsilon(i)\right) &\leq \sum_i P\left(E_P(f) \geq \hat{E}_m^{L_i}(f) + \epsilon(i)\right) \\ &\leq \sum_i p_i \cdot \delta = \delta \end{aligned} \tag{30}$$

because the bounds $\epsilon(i)$ are chosen according to equation (27). Thus, posterior bounds depending on the empirical margin and the prior confidence in achieving this margin can be derived.

5 Experiments

In the following experiments, we study the performance of matrix relevance adaptation in the context of several learning problems. We compare global and local matrix schemes with the corresponding schemes for relevance vectors as used in GRLVQ, for instance. To this end, we restrict our GMLVQ algorithm to the adaptation of diagonal matrices. Note that we implement gradient steps in Ω, while the original GRLVQ scheme corresponds to steepest descent w.r.t. diagonal elements of $\Lambda = \Omega\Omega$, directly. This slight modification was necessary in order to allow for a fair comparison of vector and matrix adaptation with the same learning rates.
As initial metric parameters the matrix Λ is set to be diagonal with $\Lambda_{ii} = 1/N$, $i{=}1, \ldots, N$. The same holds for local relevance matrices, respectively. To initialize the prototypes, we choose the mean values of random subsets of data points selected from each class.

Artificial data

In a first illustrative experiment, the algorithms are applied to two-dimensional artificial data in a binary classification problem. Each class corresponds to a cigar-shaped cluster with equal prior weights. Raw data is generated according to axis-aligned Gaussians with mean $\mu_1 = [1.5, 0.0]$ for class 1 and $\mu_2 = [-1.5, 0.0]$ for class 2 data, respectively. In both classes the standard deviations are $\sigma_{11} = 0.5$ and $\sigma_{22} = 3.0$. These clusters are rotated independently by

the angles $\varphi_1 = \pi/4$ and $\varphi_2 = -\pi/6$ so that the two clusters intersect. Training and test set consist of 600 data points per class, respectively. In order to reduce the influence of *lucky set* compositions, the experiments are performed on ten statistically independent data sets. One of these data sets is visualized in Fig. 1(a). It will be used for demonstration purposes in the following.
For training, we use one prototype per class and the following settings: We use the standard Euclidean metric (GLVQ), an adaptive diagonal metric (GRLVQ), individual adaptive diagonal metrics for each prototype (LGRLVQ), a global adaptive matrix (GMLVQ), and individual adaptive matrices for every prototype (LGMLVQ). Relevance- or matrix learning starts after an initial phase of 100 epochs of pure prototype adaptation. Training is done for 1000 epochs in total. In all experiments, the learning rates are chosen differently for prototypes and metric parameters and are annealed during training. The initial learning rate $\eta_p(0)$ for prototypes is chosen as 0.05, the initial learning rate for the metric parameters $\eta_m(0)$ is set to 0.005. Annealing is performed according to the following learning rate schedule:

$$\eta_{p,m}(t) = \frac{\eta_{p,m}(0)}{1 + \tau \cdot (t - t_{\text{start}})} \tag{31}$$

where t_{start} denotes the starting epoch for the adaptation, i.e. it is 1 for the prototype adaptation and 100 for the adaptation of matrix elements and relevance factors. The parameter τ is chosen as 0.0001. The mean classification accuracies on the training and test sets are summarized in the left panel of Tab. 1. The position of the resulting prototypes and decision boundaries for the example data set are shown in Fig. 1 (b)-(f).
The relevance matrix

$$\Lambda \approx \begin{pmatrix} 0.817 & 0.3867 \\ 0.3867 & 0.183 \end{pmatrix}$$

which results from GMLVQ training on the example data set has the eigenvalues one and zero. The same eigenvalue spectrum is obtained in all runs, i.e. for all randomly shuffled data sets. It implies that the algorithm determines only one

Table 1: Percentage of correctly classified patters for the artificial data and the image segmentation data using different LVQ algorithms.

Artificial data		
Algorithm	Training	Test
GLVQ	74.4	74.5
GRLVQ	74.5	74.5
GMLVQ	79.8	79.0
LGRLVQ	81.1	80.8
LGMLVQ	92.1	90.8

Image data		
Algorithm	Training	Test
GLVQ	84.8	83.2
GRLVQ	88.9	88.8
GMLVQ	91.1	90.2
LGRLVQ	91.4	90.0
LGMLVQ	99.1	94.4

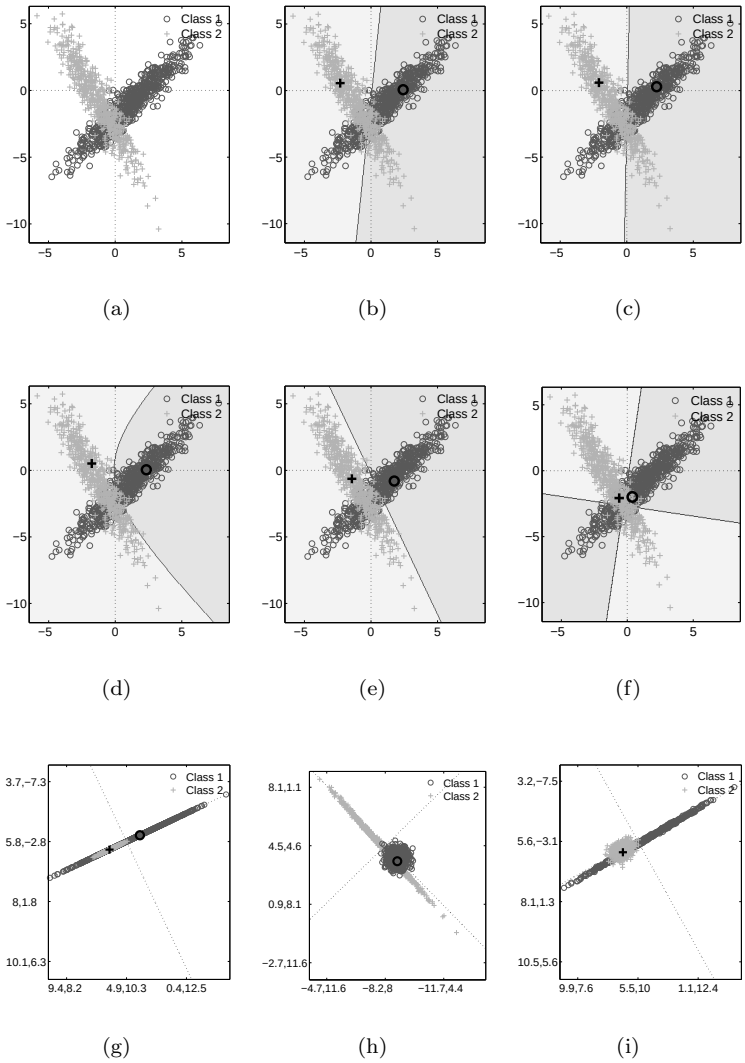


Figure 1: (a) Artificial dataset, (b)-(f) Prototypes and receptive fields, (b) GLVQ, (c) GRLVQ, (d) LGRLVQ, (e) GMLVQ, (f) LGMLVQ, (g) Training set transformed by global matrix Ω, (h) Training set transformed by local matrix Ω_1, (i) Training set transformed by local matrix Ω_2.
In (g), (h), (i) the dotted lines correspond to the eigendirections of Λ, Λ_1 and Λ_2, respectively.

new feature to discriminate the data. The respective direction in feature space

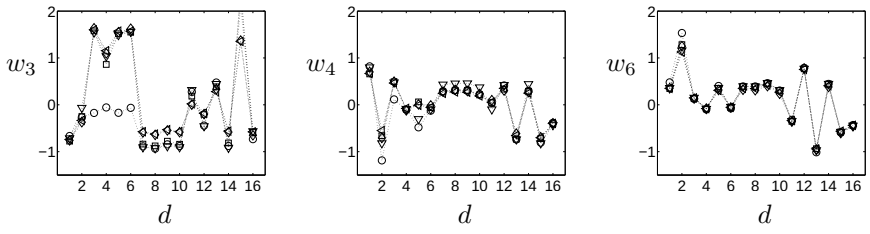
is defined by the first eigenvector of Λ. The corresponding matrix Ω projects the data onto this 1-dimensional subspace as depicted in Fig. 1(g). Furthermore, this figure displays that class 2 samples spread only slightly around their prototype in the new feature space. The opposite holds for class 1 samples, implying large distances of these data points to their prototype. Accordingly, the classification performance is much better for class 2 samples. This can also be seen in the receptive fields in Fig. 1(e). On this data set, almost 98% of the training error goes back to class 1 data.
For local matrix adaptation, the algorithm also tends towards a state with eigenvalues one and zero for both matrices Λ_1 and Λ_2. However, for parameter constellations being close to this extreme state, training samples from the overlapping region cause numerical instabilities. These data points result in small values of $|d_J - d_K|$ and, in consequence, lead to large parameter updates in equations (10) and (11). For this reason, we add the constant term $c = 0.0005$ to the diagonal elements of the matrices Ω_1 and Ω_2 after each update step before the normalization. This prevents the eigenvalues from reaching the extreme state and eliminates sudden changes of the performance. Note, that this phenomenon is due to the specific structure of the considered data set and does not constitute a general drawback of our method. The resulting local relevance matrices on the example data set are

$$\Lambda_1 \approx \begin{pmatrix} 0.4886 & -0.4716 \\ -0.4716 & 0.5114 \end{pmatrix} \qquad \Lambda_2 \approx \begin{pmatrix} 0.7584 & 0.4114 \\ 0.4115 & 0.2416 \end{pmatrix}.$$

Their eigenvalues read $\mathrm{eig}(\Lambda_1) \approx (0.972, 0.028)$ and $\mathrm{eig}(\Lambda_2) \approx (0.986, 0.014)$ at the end of training. Figures 1(h) and 1(i) denote the projections of the training set to the feature spaces which are determined for the two prototypes individually. One can clearly observe the benefit of individual matrix adaptation: It allows each prototype to shape its distance measure according to the local ellipsoidal form of the class. This way, the data points of both ellipsoidal clusters can be classified correctly except for the tiny region where the classes overlap. Note that, for local metric parameter adaptation, the receptive fields of the prototypes are no longer separated by straight lines (Fig. 1(d)) and need no longer be convex (Fig. 1(f)).

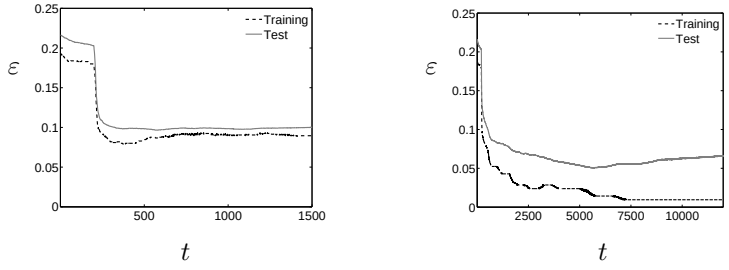
Image Segmentation Data

In a second experiment, we apply the algorithms to the image segmentation data set provided by the UCI repository [18]. The data set contains 19-dimensional feature vectors, which encode different attributes of 3×3 pixel regions extracted from outdoor images. Each such region is assigned to one of seven classes (brickface, sky, foliage, cement, window, path, grass). The features 3-5 are (nearly) constant and are eliminated for this experiment. As a further preprocessing step, the features are normalized to zero mean and unit variance. The training set consists of 210 data points (30 samples per class), the test set contains 300 data points per class.



(a) Class 3: foliage (b) Class 4: cement (c) Class 6: path

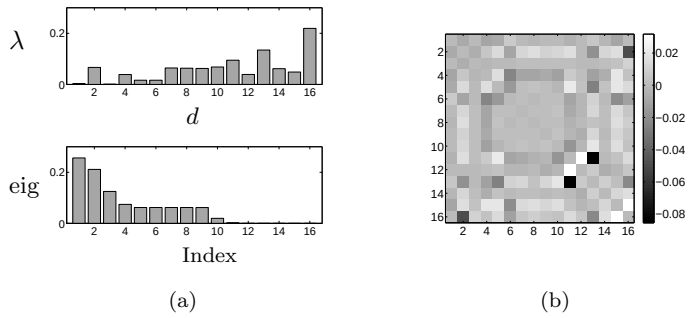
Figure 2: Prototypes of 3 classes, identified by the different algorithms. GLVQ: circle, GRLVQ: square, GMLVQ: diamond, LGRLVQ: triangle (down), LGMLVQ: triangle (left).



(a) (b)

Figure 3: Evolution of the mean training and test error in the course of (a) GMLVQ-Training. (b) LGMLVQ-Training.

Each class is approximated by one prototype, respectively. We use the same learning rate schedule as in the previous experiment, Eq. (31). The initial learning parameters are chosen as follows: $\eta_p(0) = 0.001, \eta_m(0) = 0.0001$ and $\tau = 0.0001$. In all experiments, the adaptation of the metric parameters starts after 200 epochs of pure prototype training, i.e. $t_{start} = 200$. We continue learning until the training error remains constant. In order to reduce the influence of random fluctuations, we average our results over five runs with varying initializations. The mean classification accuracies are summarized in the right panel of Tab. 1. The algorithms based on adaptive distance measures show a better performance than GLVQ. Remarkably, using different metrics influences the final location of the prototypes in feature space only slightly. Clear differences affect only a small number of features in certain classes. Significant variations are observed for the GLVQ prototypes, mainly, see Fig. 2.
Figure 3(a) displays the averaged classification errors on training- and test sets in the course of GMLVQ-Training. The training error always stabilizes after



(a) (b)

Figure 4: Visualization of the global relevance matrix Λ after 1000 epochs GMLVQ-Training (a) Diagonal elements and eigenvalues (b) Off-diagonal elements. The diagonal elements are set to zero for the plot.

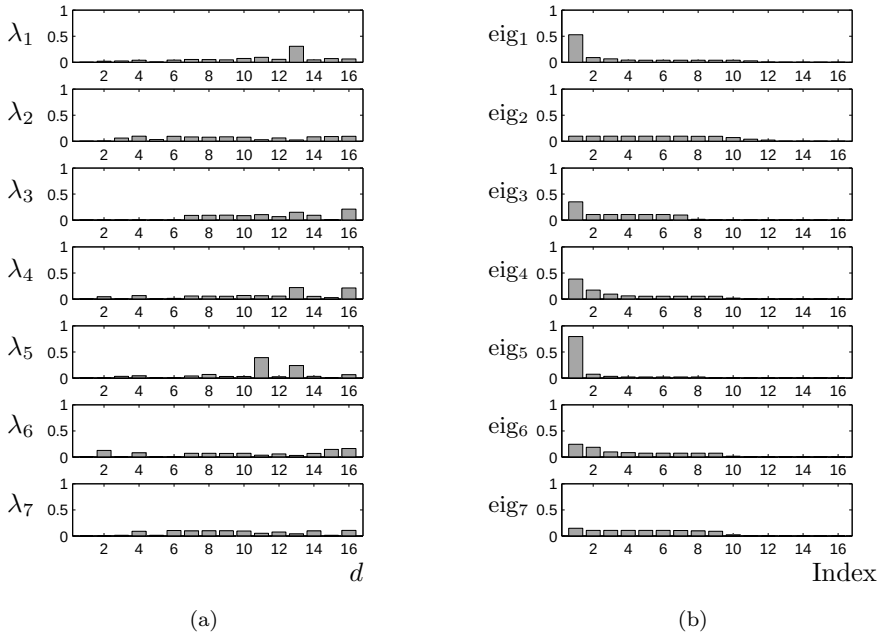
approx. 1000 epochs. The relevance matrices observed at that point in time turn out to be very robust with respect to different initializations. In single runs we observe that the algorithm finally can converge to different local minima of the cost function when training is continued. One of the matrices after 1000 epochs is visualized in Figure 4. The eigenvalue spectrum shows that the classifier uses a ten dimensional space to classify the data. The dimension weighted as most relevant in the original space is feature 16 (hue-mean, see [18]). GRLVQ training with identical initializations and learning parameters also weights the same dimension as most important. However, the relevance profile is much more pronounced ($\lambda_{16}^{\mathrm{GRLVQ}} \approx 0.9$). The additional consideration of correlations for the computation of distance values causes a less distinct relevance profile.

The training error during LGMLVQ-training remains constant after approx. 7500 sweeps through the training set (Fig. 3(b)). The test error shows slight overfitting effects. It reaches a minimum after approx. 6000 epochs and increases slightly in the further course of training. In the following we present the results obtained after 7500 epochs. At this point, training of individual matrices per prototype achieves a test accuracy of 94.4%, an improvement of approx. 4.9% compared to GMLVQ and LGRLVQ. Slightly better performance could be obtained by an appropriate *early stopping scheme* using an evaluation set. We are aware of only one SVM result in the literature which is applicable for comparing the performance. In [19], the authors achieve 93.95% accuracy on the test set.
Figure 5 shows the diagonal elements and eigenvalue spectra of all local matrices we obtain in one run which are also representative for the other experiments. Matrices with a clear preference for certain dimensions on the diagonal also display a distinct eigenvalue profile (e.g. Λ_1, Λ_5). Similarly, matrices with almost balanced relevance values on the diagonal exhibit only a weak decay from the first to the second eigenvalue (e.g. Λ_2, Λ_7). This observation for diago-

nal elements and eigenvalues coincides with a similar one for the off-diagonal elements. Figure 6 visualizes the off-diagonal elements of the local matrices Λ_1, Λ_2 and Λ_5. Corresponding to the balanced relevance- and eigenvalue profile of matrix Λ_2, the off-diagonal elements are only slightly different from zero. This may indicate diffuse data without a pronounced, hidden structure. There are obviously no other directions in feature space which could be used to significantly minimize distances within this class. On the contrary, the matrices Λ_1 and Λ_5 show a clearer structure. The off-diagonal elements cover a much wider range and there is a clearer emphasis on particular dimensions. This implies that class-specific correlations between certain features have significant influence. The most distinct weights for correlations with other dimensions are obtained for features, which also gain high relevance values on the diagonal. It is visible that especially relations between the dimensions encoding color information are emphasized. The dimensions weighted as most important are features 11: exred-mean (2R - (G + B)) and 13: exgreen-mean (2G - (R + B)) in both classes. Furthermore, the off-diagonal elements highlight correlations with e.g. feature 8: rawred-mean (average over the regions red values), feature 9: rawblue-mean (average over the regions green values), feature 10: rawgreen-mean (average over the regions green values). For a description of the features, see [18].

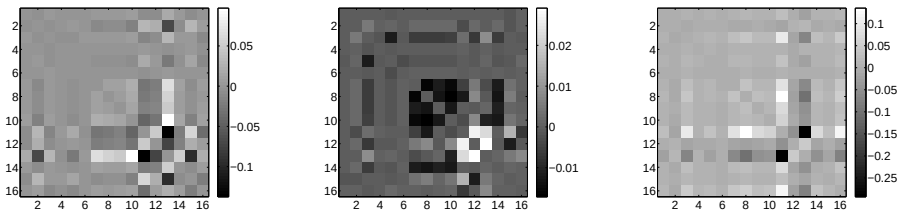
Splice Site Recognition

As a second benchmark test, we apply the algorithms to the publicly available *C. elegans* data set for the detection of splice sites. The data can be downloaded at `http://www2.fml.tuebingen.mpg.de/raetsch/projects/AnuSplice`. The feature vectors encode a sequence of 50 nucleotides with a potential splice site in the center, in between the characteristic dinucleotide AG. The classification task consists in separating sequences containing a splice site from sequences without a splice site, i.e. a two-class problem is defined accordingly. The 4 nucleotides are encoded as corners of a tetrahedron in a 3-dimensional vector space. This realizes equal pairwise distances between the nucleotides. The redundant dinucleotide AG in the center of all feature vectors is removed. Accordingly, the sequence information is represented by a vector in $\mathbb{R}^N$ with $N = 144$. The data consists of 5 data sets containing 1000 data points for training and 10000 data points for testing, respectively. The sets are not balanced and their composition varies slightly. They contain approximately two times more non-splice site-samples than examples of splice sites.
We would like to stress that our main interest in this experiment is not related to the biological aspects of the classification problem. We will put emphasis on the analysis of our method and the comparison of the new algorithm to the adaptation of relevance vectors.
We choose the simplest setting and approximate each class with one prototype respectively. The initial learning parameters are chosen as $\eta_p(0) = 0.001$ and $\eta_m(0) = 5 \cdot 10^{-6}$. Equation (31) is used for annealing the values during training



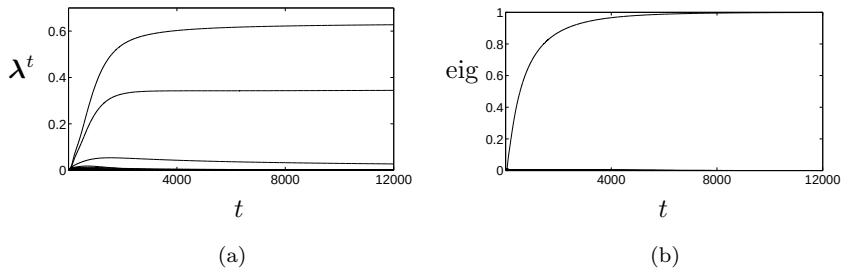
(a) (b)

Figure 5: (a) Diagonal elements of local relevance matrices Λ_{1-7}. (b) Eigenvalue spectra of local relevance matrices Λ_{1-7}.



(a) Class 1: brickface (b) Class 2: sky (c) Class 5: window

Figure 6: Visualization of the off-diagonal elements of the local matrices $\Lambda_{1,2,5}$. The diagonal elements are set to zero for the plot.



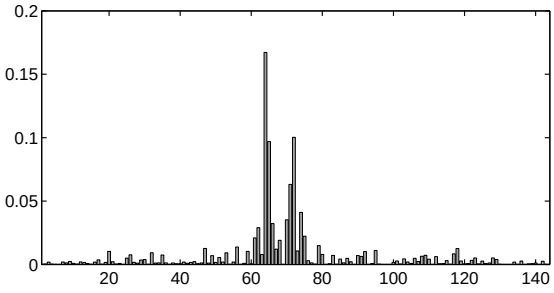
(a) (b)

Figure 7: (a) Evolution of the elements of relevance vector $\boldsymbol{\lambda}$ in the course of GRLVQ-Training. (b) Evolution of the eigenvalues of relevance matrix Λ in the course of GMLVQ training.

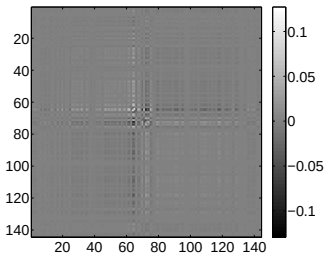
with $\tau = 0.0001$ and $t_{start} = 50$. Hence, the adaptation of metric parameters starts after 50 epochs of GLVQ training.
In all experiments, GMLVQ turns out to be more robust than GRLVQ. The learning curves of GRLVQ show strong fluctuations until they finally saturate at a constant level. The mean test set accuracy in this limit is 86.8%±1.7%. In earlier states of training the system shows better classification accuracy of above 90%. But GRLVQ performs a very strong feature selection in the further course of learning and the performance degrades in response to this oversimplification. Fig. 7(a) displays the evolution of the weight values on one of the five data sets which is also representative for the other experiments. When the error finally converges, only three factors remain significantly different from zero. Typical relevance factors are

$$\lambda_{\mathrm{GRLVQ}}^{61,64,72} \approx (0.03,\, 0.63,\, 0.34)\,.$$

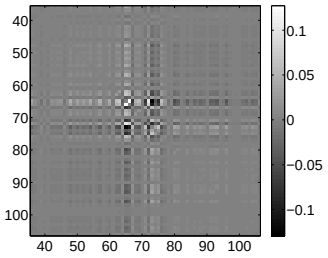
GMLVQ shows a larger stability during learning. The error curves display only small oscillations, but indicate slight overfitting effects. In the course of training, we observe an immediate focus on a single linear combination of the original features. Fig. 7(b) displays the eigenvalues of Λ as a function of training time. Except for the first eigenvalue, all other factors begin to decrease to zero immediately after starting metric adaptation. After approx. 12 000 epochs, the system finally reaches a state with only one eigenvalue remaining. At this point, the mean classification accuracy is 93.4%±0.28% on the test sets. Due to these extreme configurations of the relevance matrix, the same accuracy can be achieved in the 1-dimensional subspace defined by the first eigenvector. Accordingly, our method allows to reduce the number of features dramatically, without loosing classification performance.
Fig. 8 visualizes one of the final global matrices Λ. Features in the center display the highest relevances on the diagonal. This implies that the region around the potential splice site is of particular importance and mirrors biological knowledge. Additionally, the classifier considers correlations between different features to evaluate the similarity between the prototypes and new feature vectors. Similar



(a) Diagonal elements



(b) Off-diagonal elements

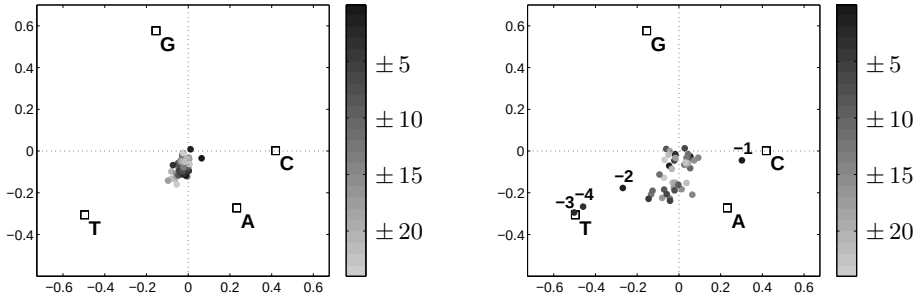


(c) Off-diagonal elements dimensions 38-108

Figure 8: Visualization of the resulting global matrix Λ after GLVMQ-Training. The diagonal elements are set to zero in (b) and (c).

to the previous experiment, the most significant off-diagonal Λ_{ij} relate to the features with the highest diagonal relevances. These correlations result in a cross-like structure in the visualization of the off-diagonal elements, see Fig. 8(b),(c).

The prototypes can be interpreted as a sequence of 48 nontrivial combinations of the four bases. They converge after approx. 5000 epochs, independent of the additional adaptation of a relevance vector or a relevance matrix in the distance measure. The representatives detected by GMLVQ approximate the data more appropriately compared to the GRLVQ-prototypes. GRLVQ slightly pushes the prototypes away from the data, several components leave the boundaries of the tetrahedron. This effect is even stronger when we train GLVQ with the fixed Euclidean metric. Fig. 9 visualizes the prototypes identified by GMLVQ on one data set. All subcomponents of the class 1 prototype are located close to the origin, the tetrahedron's center of mass. In this position they have almost equal distance to the four vertices which represent the bases. On the contrary, the



(a) Class 1: Non-Splice Site

(b) Class 2: Splice Site

Figure 9: Visualization of the resulting prototypes after GMLVQ training. The plots show the projections of the 3-dimensional elements encoding the separate components of the sequences, onto the x-y plane. The gray values display the subcomponent's position in the sequence, i.e. the distance to the potential splice site. In the right plot, we labeled the positions relative to the center which are lying closest to one of the bases.

splice site prototype exhibits a more specific structure. Especially the components with high relevance values are located close to one of the corners, i.e. one of the nucleotides, and allow for a better semantic interpretation.
In accordance with our findings for GMLVQ, localized matrix learning detects one dominating feature per class. As in our GMLVQ experiments, we train the LGMLVQ system for 12 000 epochs. Note, that both local matrices are updated in each learning step. The resulting matrices Λ_1 and Λ_2 do not show the extreme eigenvalue settings like the global relevance matrix Λ. But the error curves indicate overfitting and we do not continue training. The largest eigenvalues range from 0.71 to 0.79 (Λ_1) and from 0.95 to 0.96 (Λ_2) in the different experiments. The diagonal and the off-diagonal elements of the local matrices show the same characteristic patterns as the global matrix. However, the values of matrix Λ_2 are more distinct. The mean classification accuracy is slightly better compared to GMLVQ ($94.2\% \pm 0.31\,\%$). When we perform the classification based on the two features defined by the first eigendirections of Λ_1 and Λ_2, we loose almost no performance and still achieve $94.0\% \pm 0.38\%$ test accuracy. SVM results reported in the literature even lie above 96% [14, 20] test accuracy. Note, however, that our classifier is extremely sparse and simple and still achieves a performance which is only slightly worse.

6 Discussion

We have proposed a new metric learning scheme for LVQ classifiers which allows to adapt a full matrix according to the given classification task. This scheme extends the successful relevance learning vector quantization algorithm such that correlations of dimensions can be accounted for during training. The learning scheme can be derived directly as a stochastic gradient of the GLVQ cost function such that convergence and flexibility of the original GLVQ is preserved. Since the resulting classifier is represented by prototype locations and matrix parameters, the results can be interpreted by humans: Prototypes show typical class representatives and matrix parameters reveal the importance of input dimensions for the diagonal elements and the importance of correlations for the off-diagonal elements. Local as well as global parameters can be used, i.e. relevance terms which contribute to a good description of single classes or the global classification, respectively, can be identified. The efficiency of the model has been demonstrated in several application scenarios, demonstrating impressively the increased capacity of local matrix adaptation schemes. Interestingly, local matrix learning obtains a classification accuracy which is similar to the performance of the SVM in several cases, while it employs less complex classification schemes and maintains intuitive interpretability of the results.

The new class of algorithms drastically increases the number of free parameters of training, since full $N \times N$ matrices are updated. For a global metric, this corresponds to an adaptive linear transformation of the space according to the given classification task. For local metrics, every prototype uses its own transformation to emphasize characteristics of the respective classes. In this case, the receptive fields are no longer separated by planes but quadratic surfaces. Furthermore, they need not be convex, such that more complex settings can easily be accounted for. Clearly, straightforward modifications can be considered which employ class-wise relevance matrices or other intermediate schemes. Interestingly, only very mild overfitting is observed in our experiments, and matrix adaptation leads to excellent generalization despite the increased number of free parameters. This effect can be explained by an inherent regularization which is present in GLVQ adaptation schemes: The margin of the classifier with respect to training points, i.e. the difference of their distance to the closest correct versus the closest wrong prototype is optimized. We have rigorously shown that generalization bounds which do include the margin, but which are independent of the dimensionality of the input space and the dimensionality of the adaptive matrices, can be derived. Thus, the extended classification scheme provides increased capacity without diminishing the excellent generalization capability of LVQ classifiers.

Due to the large number of parameters, one drawback of the method consists in computational costs which scale quadratically with the data dimensionality. Therefore, the method becomes computationally infeasible for very high dimensional data. One possibility is to reduce the number of free parameters of a given matrix by enforcing e.g. a block or band structure or a controlled, limited rank. These possibilities are currently investigated in our research groups.

Appendix

Here we derive upper bounds for the Rademacher complexity of LGMLVQ networks. Assume a function class implemented by LGMLVQ networks is given as above, $M_{\mathcal{F}}$. In analogy to the Rademacher complexity, one can define the empirical Gaussian complexity, given m samples $\boldsymbol{\xi}_i$, as the expectation

$$\hat{G}_m(M_{\mathcal{F}}) = E_{g_1,\ldots,g_m}\left(\sup_{M_f \in M_{\mathcal{F}}} \left| \frac{2}{m} \sum_{i=1}^{m} g_i \cdot M_f(\boldsymbol{\xi}_i) \right| \right) \tag{32}$$

where g_i are independent Gaussian variables with zero mean and unit variance. The Gaussian complexity is defined as the expectation over the samples

$$G_m(M_{\mathcal{F}}) = E_{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_m} \hat{G}_m(M_{\mathcal{F}}) \tag{33}$$

where $\boldsymbol{\xi}_i$ are independent and identically distributed according to the marginal of P. These quantities are closely related to the Rademacher complexity. According to [1](Lemma 4) and [17], respectively, the inequality

$$\sqrt{\pi/2} \cdot R_m(M_{\mathcal{F}}) \leq G_m(M_{\mathcal{F}}) \tag{34}$$

holds.
Our aim is to upper bound the Rademacher complexity $R_m(M_{\mathcal{F}})$ with probability at least $1 - \delta/2$ whereby M_f has the form

$$M_f(\boldsymbol{\xi}) = \left(\min_{\boldsymbol{w}_i^-} \{ d^{\Lambda^i}(\boldsymbol{w}_i^-, \boldsymbol{\xi}) \} - \min_{\boldsymbol{w}_j^+} \{ d^{\Lambda^j}(\boldsymbol{w}_j^+, \boldsymbol{\xi}) \} \right) \tag{35}$$

As beforehand, we assume that $|\boldsymbol{\xi}| \leq B$, $|\boldsymbol{w}_i| \leq B$, Λ^i is symmetric and positive semidefinite with trace 1, and we assume that P prototypes are present. Because of equation (34), we can substitute the Rademacher complexity by the Gaussian complexity. The empirical Gaussian complexity and the Gaussian complexity differ by more than ϵ with probability at most $2\exp(-\epsilon^2 m/8)$ according to [1](Theorem 11), i.e. they differ by no more than $\sqrt{8/m \cdot \ln(4/\delta)}$ with probability at least $1 - \delta/2$. Thus, it is sufficient to upper bound the empirical Gaussian complexity of LGMLVQ networks.
Note that

$$\hat{R}_m\left(\sum_i \mathcal{F}_i\right) \leq \sum_i \hat{R}_m(\mathcal{F}_i) \tag{36}$$

holds for all function classes $\mathcal{F}_i$ due to the triangle inequality. Further, the empirical Gaussian complexity does obviously not change when multiplying a function class by -1. Thus, we can upper bound $\hat{G}_m(M_{\mathcal{F}})$ by twice the complexity of a function class of functions of the form

$$\boldsymbol{\xi} \to \min_{\boldsymbol{w}_i} \{ d^{\Lambda^i}(\boldsymbol{w}_i, \boldsymbol{\xi}) \} \tag{37}$$

where the minimum is taken over at most P terms.
The function which computes the minimum of P values,

$$(a_1, \ldots, a_P) \mapsto \min\{a_1, \ldots, a_P\} \tag{38}$$

is Lipschitz continuous with constant $\sqrt{8P}$, as can be seen as follows: Obviously, $|\min\{a, 0\} - \min\{a', 0\}| \leq |a - a'|$. Further, $\min\{a, b\} = \min\{a - b, 0\} + b$. Hence, by induction, $|\min\{a_1, \ldots, a_P\} - \min\{a'_1, \ldots, a'_P\}| \leq 2|a_P - a'_P| + |\min\{a_1, \ldots, a_{P-1}\} - \min\{a'_1, \ldots, a'_{P-1}\}| \leq \ldots \leq 2|a_1 - a'_1| + \ldots + 2|a_P - a'_P|$. Thus, $|\min\{a_1, \ldots, a_P\} - \min\{a'_1, \ldots, a'_P\}|^2 \leq 4 \sum_{ij} |a_i - a'_i| \cdot |a_j - a'_j| \leq 8P \sum_i |a_i - a'_i|^2$.
Because of [1](Theorem 14), we find

$$\hat{G}_m(\Phi \circ \mathcal{F}) \leq 2L \sum_i \hat{G}_m(\mathcal{F}_i) \tag{39}$$

for every Lipschitz continuous function Φ on a real vector space with Lipschitz constant L and a function class $\mathcal{F}$ contained in the direct sum of the classes $\mathcal{F}_i$. Thus, because of the Lipschitz continuity of the function min, it is sufficient to upper bound the empirical Gaussian complexity of function classes of the form,

$$\boldsymbol{\xi} \mapsto (\boldsymbol{\xi} - \boldsymbol{w})^t \Lambda (\boldsymbol{\xi} - \boldsymbol{w}) = \boldsymbol{\xi}^t \Lambda \boldsymbol{\xi} - 2\boldsymbol{\xi}^t \Lambda \boldsymbol{w} + \boldsymbol{w}^t \Lambda \boldsymbol{w} \,. \tag{40}$$

This decomposes into a linear function

$$\boldsymbol{\xi} \mapsto -2\boldsymbol{\xi}^t \Lambda \boldsymbol{w} + \boldsymbol{w}^t \Lambda \boldsymbol{w} \tag{41}$$

and a quadratic form

$$\boldsymbol{\xi} \mapsto \boldsymbol{\xi}^t \Lambda \boldsymbol{\xi} = \sum_{ij} \Lambda_{ij} \xi_i \xi_j \,. \tag{42}$$

According to [1](Lemma 22), the empirical Gaussian complexity of linear forms $\boldsymbol{\xi} \mapsto \boldsymbol{w}^t \boldsymbol{\xi}$ can be upper bounded by

$$\frac{2C_1 C_2}{\sqrt{m}} \tag{43}$$

where inputs are restricted to $|\boldsymbol{\xi}| \leq C_1$ and weights are restricted to $|\boldsymbol{w}| \leq C_2$. Note that inputs to the LGMLVQ network and prototypes have length at most B, further, the sum of eigenvalues of every matrix Λ^i of the LGMLVQ network is 1. Thus, functions of the form (41) correspond to linear functions with inputs restricted to $B + 1$ and weights restricted to $2B + B^2$. Functions of the form (42) can be interpreted as linear functions with enlarged inputs which size is restricted by B^2, and weights restricted by 1, since the Frobenius-norm of the matrix is given by the sum of squared eigenvalues in this case.
Collecting all inequalities, we can finally upper bound the Rademacher complexity of LGMLVQ networks by

$$\begin{aligned} &\sqrt{\pi/2} \cdot \left(\sqrt{8/m \cdot \ln(4/\delta)} + 36 \cdot P^2 \cdot \frac{2}{\sqrt{m}} \cdot \left((B+1)(2B + B^2) + B^2 \right) \right) \\ &= \frac{1}{\sqrt{m}} \cdot \mathcal{O}\left(\sqrt{\ln(1/\delta)} + P^2 B^3 \right) \,. \end{aligned} \tag{44}$$

Acknowledgement

The authors would like to thank the Max Planck Institute for the Physics of Complex Systems, Dresden, for hospitality during a seminar at which this work was finalized.

References

[1] P.L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research* 3:463–4812, 2002.

[2] M. Biehl, A. Ghosh, and B. Hammer. Learning vector quantization: The dynamics of winner-takes-all algorithms. *Neurocomputing* 69(7-9):660-670, 2006.

[3] M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research* 8:323-360, 2007.

[4] *Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ)*. Neural Networks Research Centre, Helsinki University of Technology, 2002.

[5] T. Bojer, B. Hammer, D. Schunk and K. Tluk von Toschanowitz. Relevance determination in learning vector quantization. In M.Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN'01)*, D-facto publications, pages 271-276, 2001

[6] K. Crammer, R. Gilad-Bachrach, A. Navot, and A. Tishby. Margin analysis of the LVQ algorithm. In *Advances of Neural Information Processing Systems*, 2002.

[7] R. Duda, P. Hart, and D. Storck. *Pattern Classification*, Wiley, 2001.

[8] I. Gath and A.B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11:773–781, 1989.

[9] A. Ghosh, M. Biehl, and B. Hammer. Performance analysis of LVQ algorithms: a statistical physics approach. *Neural Networks* 19:817-829, 2006.

[10] E.E. Gustafson and W.C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. *IEEE CDC*, pages 761–766, San Diego, California, 1979.

[11] B. Hammer, F.-M. Schleif, and T. Villmann. On the Generalization Ability of Prototype-Based Classifiers with Local Relevance Determination. Technical Report, Clausthal University of Technology, Ifi-05-14, 2005.

[12] B. Hammer, M. Strickert, and T. Villmann. On the generalization ability of GRLVQ networks. *Neural Processing Letters* 21(2):109–120, 2005.

[13] B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters* 21(1): 21–44, 2005.

[14] B. Hammer, M. Strickert, and T. Villmann. Prototype based recognition of splice sites. In U. Seifert, L. Jain, and P. Schweitzer, editors, *Bioinformatics using computational intelligence paradigms*, Springer-Verlag, 2004.

[15] B. Hammer and T. Villmann. Generalized relevance learning vector quantization, *Neural Networks* 15:1059-1068, 2002.

[16] T. Kohonen, *Self-organizing maps*, Springer, Berlin, 1995.

[17] M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Springer, 1991.

[18] D.J.Newman, S.Hettich, C.L.Blake, C.J.Merz. *UCI Repository of machine learning databases* [http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[19] Y. Prudent and A. Ennaji, *A K Nearest Classifier design*, Electronic Letters on Computer Vision and Image Analysis 5(2):58–71, 2005.

[20] G. Rätsch and S. Sonnenburg. Accurate Splice Site Prediction for Caenorhabditis Elegans, pages 277-298. MIT Press series on Computational Molecular Biology. MIT Press, 2004.

[21] 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.

[22] S. Seo, M. Bode, and K. Obermayer. Soft nearest prototype classification. *IEEE Transactions on Neural Networks* 14(2):390-398, 2003.

[23] S. Seo and K. Obermayer. Soft Learning Vector Quantization. *Neural Computation* 15(7):1589-1604, 2003.

[24] S.Shalev-Schwartz, Y. Singer, and A.Y. Ng. Online and batch learning of pseudo-metrics. *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

[25] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems* 19, MIT Press, Cambridge, MA, 2006.