Matrix Learning in Learning Vector Quantization

Biehl, Michael; Hammer, Barbara; Schneider, Petra

TU Clausthal

Clausthal University of Technology

Matrix Learning in Learning Vector Quantization

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

IfI Technical Report Series **IfI-06-14**

Department of Informatics
Clausthal University of Technology

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Matrix Learning in Learning Vector Quantization

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

1- Rijksuniversiteit Groningen - Mathematics and Computing Science
P.O. Box 800, 9700 AV Groningen - The Netherlands

2- Clausthal University of Technology - Institute of Computer Science
Julius Albert Strasse 4, 38678 Clausthal-Zellerfeld - Germany

Abstract

We propose a new matrix learning scheme to extend Generalized Relevance Learning Vector Quantization (GRLVQ), an efficient prototype-based classification algorithm. 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 for GRLVQ, a full matrix is more powerful to represent the internal structure of the data appropriately. Interestingly, large margin generalization bounds can be transfered to the case of a full matrix such that bounds which are independent of the input dimensionality and the number of parameters arise. This also holds for local metrics attached to each prototype. The algorithm is tested and compared to GLVQ without metric adaptation [16] and GRLVQ with diagonal relevance factors using an artificial dataset and the image segmentation data from the UCI repository [15].

1 Introduction

Learning vector quantization (LVQ) as introduced by Kohonen constitutes a particularly intuitive and simple though powerful classification scheme [14] 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 multiclass problems; and, unlike many alternative neural classification schemes such as feedforward networks and support vector machines, the resulting classifier is human understandable because of the intuitive classification of data points to the class of their closest prototypes. For these reasons, LVQ has been used in a variety of academic and commercial applications such as image analysis, telecommunication, robotics, etc. [4].

Original LVQ, however, suffers from several drawbacks such as slow convergence and instable behavior because of which a variety of alternatives have been proposed, as explained e.g. in [14]. Still, there are two major drawbacks of these methods, which have only recently been tackled. On the one hand, LVQ relies on heuristics and a full

mathematical investigation of the algorithm is lacking. This problem often leads to unexpected behavior and instabilities of training. Recently, it has rigorously been shown that already slight variations of the basic LVQ learning scheme yield quite different results [2, 3]. For this reason, variants of LVQ which can be derived from an explicit cost function are particularly interesting, since they obey a well-defined dynamics. Several proposals for cost functions can be found in the literature, the first one being generalized LVQ which forms the basis for the method we will consider in this article [16]. Two alternatives which implement soft relaxations of the original learning rule are [17, 18]. These two approaches, however, have the drawback that the original crisp limit case does not exist (for [17]) resp. it shows poor results also in simple settings [8] (for [18]). The cost function as proposed in [16] has the benefit that it shows stable behavior and it aims at a good generalization ability already during training as pointed out in [11].

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 fit 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 of the dimensions differs, and for data which involves correlations of the dimensions. In these cases, which are quite common in practice, simple LVQ fails. Recently, a generalization of LVQ has been proposed based on the formulation as cost optimization in [16] which allows the incorporating of every differentiable similarity measure [12]. The specific choice of the similarity measure as a simple weighted diagonal metric with adaptive relevance terms turned out as particularly suitable in many practical applications since it can easily account for irrelevant or inadequately scaled dimensions. At the same time, it allows easy interpretation of the result because the relevance profile can directly be interpreted as the contribution of the dimensions to the classification [13]. For an adaptive diagonal metric, dimensionality independent large margin generalization bounds can be derived [11]. 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 an arbitrary choice of the metric.

Often, the dimensions are correlated in classification tasks. In unsupervised clustering, correlations of data are accounted for e.g. by the classical Mahalanobis distance [6] or fuzzy-covariance matrices as derived e.g. in the fuzzy-classifiers [7, 9]. For supervised classification tasks, however, an explicit metric which takes correlations into account has not yet been proposed. Based on the general framework as presented in [12], we develop an extension of LVQ to an adaptive full matrix which describes a general euclidean metric and which can account for correlations of any two data dimensions in this article. This algorithm allows for an appropriate scaling and also an appropriate rotation of the data to learn a coordinate system which is optimum for the given classification task. Thereby, the matrix can be chosen as one global matrix, or as individual matrices attached to the prototypes, the latter accounting for local ellipsoidal shapes of the classes. 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 benefit of this extended method for several concrete classification tasks.

2 Generalized metric LVQ

LVQ aims at approximating a clustering by prototypes. Assume training data $(\vec{\xi}_i, y_i) \in \mathbb{R}^N \times \{1, \ldots, C\}$ are given, N denoting the data dimensionality and C the number of different classes. A LVQ network consists of a number of prototypes which are characterized by their location in the weight space $\vec{w}_i \in \mathbb{R}^N$ and their class label $c(\vec{w}_i) \in \{1, \ldots, C\}$. Classification takes place by a winner takes all scheme. For this purpose, a (possibly parameterized) similarity measure d^λ is fixed for $\mathbb{R}^N$. Often, the standard euclidean metric is chosen. A data point $\vec{\xi} \in \mathbb{R}^N$ is mapped to the class label $c(\vec{\xi}) = c(\vec{w}_i)$ of the prototype i for which $d^\lambda(\vec{w}_i, \vec{\xi}) \leq d^\lambda(\vec{w}_j, \vec{\xi})$ holds for every $j \neq i$ (breaking ties arbitrarily), i.e. it is mapped to the class of the closest prototype, the 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 [12]: Training is derived as a minimization of the cost function

$$\sum_i \Phi\left(\frac{d_J^\lambda - d_K^\lambda}{d_J^\lambda + d_K^\lambda}\right)$$

where Φ is a monotonic function, e.g. the identity or the logistic function, $d_J^\lambda = d^\lambda(\vec{w}_J, \vec{\xi}_i)$ is the distance of data point $\vec{\xi}_i$ from the closest prototype $\vec{w}_J$ with the same class label y_i, and $d_K^\lambda = d^\lambda(\vec{w}_K, \vec{\xi}_i)$ is the distance from the closest prototype $\vec{w}_K$ with a different class label than y_i. Note that the nominator is smaller than 0 iff the classification of the data point is correct. The smaller the nominator, the greater the security of classification, i.e. the difference of the distance from a correct and wrong prototype. The denominator scales this term such that it lies in between $(-1, 1)$. 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 as introduced in [16].

The learning rule can be derived from this cost function taking the derivative. We assume that the similarity measure $d^\lambda(\vec{w}, \vec{\xi})$ must be differentiable with respect to the parameters $\vec{w}$ and λ. As shown in [12], for a given pattern $\vec{\xi}$ the derivatives yield

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

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

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

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

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

The adaptation of λ 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 [12] that these update rules are valid if the metric is differentiable. Thereby, this argument also holds for the borders of receptive fields, i.e. an underlying continuous input distribution, as can be shown using delta-functions [12].

It has been demonstrated in [12], the squared weighted euclidean metric $d^\lambda(\vec{w}, \vec{\xi}) = \sum_i \lambda_i (w_i - \xi_i)^2$ where $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$ constitutes a simple and powerful choice which allows to use prototype based learning also in the presence of high dimensional data with a different (but priorly not known) relevance of the input dimensions. This choice has the benefit that the relevance terms λ_i which are automatically adapted during training allow an interpretation of the classification: the dimensions with large parameters λ_i contribute most to the classification. We refer to this method as generalized relevance learning vector quantization (GRLVQ). Alternative choices have been introduced in [12], including, for example, metrics which take local windows into account e.g. for time series processing.

Note that the relevance factors, i.e. the choice of the metric need not be global, but it can be attached to a single prototype. In this case, individual updates take place for the relevance factors λ^j for each prototype j, and the distance of a data point $\vec{\xi}_i$ from prototype $\vec{w}_j$, $d^{\lambda_j}(\vec{w}_j, \vec{\xi}_i)$ 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 [10]. We refer to this version as localized GRLVQ (LGRLVQ).

3 Generalized matrix LVQ

Here, we introduce another specific relevant choice of the similarity measure, a full matrix, which can account for arbitrary correlations of the dimensions. We are interested in a similarity measure of the form

$$d^\Lambda(\vec{w}, \vec{\xi}) = (\vec{\xi} - \vec{w})^T \, \Lambda \, (\vec{\xi} - \vec{w})$$

where Λ is a full matrix. Note that, this way, arbitrary euclidean metrics can be achieved 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 [7, 9], however, accompanied by the drawback of increased computational costs, since these methods require a matrix inversion at each adaptation step. For the metric as introduced above, a variant which has costs $\mathcal{O}(N^2)$ can be derived.

Note that the above similarity measure only leads to a squared euclidean distance in an appropriately transformed space if Λ is positive (semi-) definite. We can achieve this by substituting

$$\Lambda = \Omega\,\Omega^T$$

which yields $\vec{u}^T\Lambda\vec{u} = \vec{u}^T\Omega\,\Omega^T\vec{u} = \left(\Omega^T u\right)^2 \geq 0$ for all $\vec{u}$, where Ω is an arbitrary matrix. 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 reduces the computational costs by 2. Hence

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

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

$$\vec{\nabla}_w\, d^{\Lambda} = -2\Lambda\,(\vec{\xi} - \vec{w}) = -2\Omega\Omega(\vec{\xi} - \vec{w}).$$

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

$$\begin{aligned}\frac{\partial d_{\Lambda}}{\partial \Omega_{lm}} &= \sum_j (\xi_l - w_l)\Omega_{mj}(\xi_j - w_j) + \sum_i (\xi_i - w_i)\Omega_{il}(\xi_m - w_m) \\ &= (\xi_l - w_l)\left[\Omega(\vec{\xi} - \vec{w})\right]_m + (\xi_m - w_m)\left[\Omega(\vec{\xi} - \vec{w})\right]_l\end{aligned}$$

where subscript l denotes component l of the vector. Thus, we get the update formulas

$$\begin{aligned}\Delta\vec{w}_J &= \epsilon \cdot \phi'(\mu(\vec{\xi})) \cdot \mu^+(\vec{\xi}) \cdot \Omega\Omega \cdot (\vec{\xi} - \vec{w}_J) \\ \Delta\vec{w}_K &= -\epsilon \cdot \phi'(\mu(\vec{\xi})) \cdot \mu^-(\vec{\xi}) \cdot \Omega\Omega \cdot (\vec{\xi} - \vec{w}_K)\end{aligned}$$

For the update of the matrix elements Ω_{lm} we get

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

Thereby, the learning rate for the metric can be chosen independently of the learning rate of the prototypes. Usually, it is an order of magnitude smaller to account for a slower time-scale of metric learning compared to the weight updates. We assume Ω to be symmetric. Note that this is automatically fulfilled because the above updates are symmetric w.r.t. l and m.

After each update Λ should be normalized to prevent the algorithm from degeneration. One possibility is to enforce

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

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, here. 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

$$\Lambda_{ii} = \sum_k \Omega_{ik}\Omega_{ki} = \sum_k (\Omega_{ik})^2 .$$

So normalization takes place 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.

Note that 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 $\vec{\xi}$ from a prototype $\vec{w}_j$ is computed as $d^{\Lambda^j}(\vec{\xi} - \vec{w}_j) = (\vec{\xi} - \vec{w}_j)^T \Lambda^j (\vec{\xi} - \vec{w}_j)$. Each matrix is adapted individually in the following way: given $\vec{\xi}$ with closest correct prototype $\vec{w}_J$ and closest incorrect prototype $\vec{w}_K$, we get the update formula

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

Localized matrices have the benefit that general ellipsoidal clusters can be learned by the method whereby ellipsoidal clusters need not be aligned to the axes (this restriction holds for GRLVQ). And the main axes of the ellipsoid can be chosen independently for every prototype. Thus, general mixtures of Gaussians can be approximated in a very elegant way. We refer to this general version as localized GMLVQ (LGMLVQ).

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 shown in [5] 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 minimum difference of the distances computed for classification. A similar result has been derived in [11] for LVQ-networks as considered above which possess an adaptive diagonal metric. Remarkably, the margin is thereby directly correlated to the nominator of the cost function as introduced above, i.e. these learning algorithms inherently aim at margin optimization during training. As pointed out in [12], these results transfer immediately to kernelized versions of this 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, i.e. the kernel is optimized according to the given classification task in this setting.

Here, we directly derive a large margin generalization bound for (localized) GMLVQ networks with a full adaptive matrix attached to every prototype, whereby we use the ideas of [11]. We consider a LGMLVQ network given by P prototypes $\vec{w}_i$ with inputs $|\vec{\xi}| \leq B$ for some $B > 0$ and the case of a binary classification. That means, prototypes $c(\vec{w}_i)$ are labeled by 1 or -1. Classification takes place by a winner takes all rule, i.e.

$$\vec{\xi} \mapsto c(\vec{w}_i) \text{ where } (\vec{\xi} - \vec{w}_i)^T \Lambda^i (\vec{\xi} - \vec{w}_i) \leq (\vec{\xi} - \vec{w}_j)^T \Lambda^j (\vec{\xi} - \vec{w}_j) \, \forall j \neq i \tag{1}$$

with positive semidefinite matrix Λ^i with normalization $\sum_l \Lambda^i_{ll} = 1$. The network corresponds to a function in the class

$$\mathcal{F} := \{f : \mathbb{R}^N \to \{-1, 1\} \mid f \text{ is given by formula (1) for some } \Lambda^i, \vec{w}_i\}$$

We can assume that prototypes are located within the data points, i.e. $|\vec{w}_i| \leq B$.

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

$$E_P(f) := P(y \neq f(y))$$

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

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

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.

The training algorithm of LGMLVQ optimizes a cost function which is correlated to the number of misclassifications of training. Assume a pattern $(\vec{\xi}, y)$ is classified by a GMLVQ network which implements the function f. We define the margin

$$M_f(\vec{\xi}, y) = -d_J^{\Lambda^J} + d_K^{\Lambda^K}$$

whereby $d_J^{\Lambda^J}$ refers to the distance from the closest prototype with class y and $d_K^{\Lambda^K}$ refers to the distance from the closest prototype with class different from y. Note that LGMLVQ tries to maximize this margin during training since it occurs as nominator within the cost function. Following the approach [1], we define the loss function

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

where $\rho > 0$ is some fixed value. The term

$$\hat{E}_m^L(f) := \sum_{i=1}^{m} L(M_f(\vec{\xi}_i, y_i))/m$$

accumulates the number of errors for a given data set, and, in addition, also punishes all correct classifications with a margin smaller than ρ.

It is possible to correlate the generalization error and this modified empirical error by a dimensionality independent bound. According to [1](Theorem 7) for all $f \in \mathcal{F}$ with probability at least $1 - \delta/2$, the inequality

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

holds, whereby K is a universal constant and $G_m(\mathcal{F})$ is the Gaussian complexity which is the expectation of the quantity

$$E_{g_1,\ldots,g_m}\left(\sup_{f\in\mathcal{F}}\left|\frac{2}{m}\sum_{i=1}^{m} g_i \cdot f(\vec{\xi}_i)\right|\right)$$

with respect to the patterns $\vec{\xi}_i$ where g_i constitute independent Gaussian variables with zero mean and unit variance. It measures the amount of surprise in the considered function class.

A winner-takes-all classification according to equation (1) can be formulated as Boolean formula over terms of the form $d_i^{\Lambda^i} - d_j^{\Lambda^j}$ where i and j enumerate the mutually different prototypes. There exist $P(P-1)/2$ such pairs. According to [1](Theorem 16), we find

$$G_m(\mathcal{F}) \leq P(P-1) \cdot G_m(\mathcal{F}_{ij})$$

whereby $\mathcal{F}_{ij}$ denotes a LGMLVQ network with only two prototypes. For a network with two prototypes, we get for input $\vec{\xi}$

$$\begin{aligned} & d_i^\Lambda - d_j^{\Lambda^j} \leq 0 \\ \iff \quad & \vec{\xi}^T \Lambda^i \vec{\xi} - \vec{\xi}^T \Lambda^j \vec{\xi} - \\ & 2(\vec{w}_i^T \Lambda^i - \vec{w}_j^T \Lambda^j)\vec{\xi} + (\vec{w}_i)^T \Lambda^i \vec{w}_i + (\vec{w}_j)^T \Lambda^j \vec{w}_j \leq 0 \end{aligned}$$

This is the sum of a linear classifier and a quadratic term. Adding a constant input dimension 1 to $\vec{\xi}$, the input to the linear classifier is restricted by $B+1$ and the length of the weight vector is restricted by $4B + 2B^2$ because vectors $|\vec{w}_i|$ are limited by B and the sum of the eigenvalues of Λ^i is at most 1. The empirical Gaussian complexity of this linear part is limited by

$$\frac{4B(B+1)(B+2)\sqrt{m}}{m}$$

according to [1](Lemma 22). According to the same theorem, the Gaussian complexity of the quadratic term is limited by

$$\frac{4 \cdot B \cdot \sqrt{m}}{m}.$$

Thus, an overall estimation is given by the sum of these terms which is of order $B^3/\sqrt{m}$.

Since 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 $1-\delta/2$, we finally get

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

with probability $1-\delta$. This bound is independent of the dimensionality of the data. Rather, it involves the margin ρ which is directly optimized during GMLVQ training.

This bound holds for priorly fixed margin ρ. For posterior margin ρ, a generalization of the argumentation as follows can be applied: 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 ρ_i. Define the cost function L_i as above associated to margin ρ_i and the corresponding empirical error $\hat{E}_m^{L_i}(f)$. We are interested in the probability

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

where the bound

$$\epsilon(i) = \left(\frac{K'}{\sqrt{m}} \cdot \left(\frac{P^2 B^3}{\rho_i} + \frac{P^2 \sqrt{\ln(1/(p_i \delta))}}{\rho_i} + \sqrt{\ln(1/(p_i \delta))} \right) \right)$$

depends on the empirical margin, K' being a constant. We can argue

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

because the bounds $\epsilon(i)$ are chosen according to equation (2). Thus, posterior bounds depending on the empirical margin and the prior confidence in achieving this margin can be derived.

5 Experiments

We test the performance of GMLVQ in comparison to GRLVQ and GLVQ without metric adaptation using an artificial data set and the image segmentation data set provided in the UCI repository [15].

5.1 Artificial Data

In a first experiment, the algorithm is applied to a two-dimensional artificial dataset consisting of two classes with one cigar shaped cluster each. The two clusters intersect, as depicted in Fig.1(a). The data consists of two Gaussians with the same probability which give two classes. The Gaussians are generated with mean values $\mu_1 = [1.5, 0.0]$ and $\mu_2 = [-1.5, 0.0]$, respectively, and variance $\sigma_{1,2} = [0.5, 3.0]$, these axes-aligned cigars are rotated about the origin by the angles $\varphi_1 = \pi/4$ and $\varphi_2 = -\pi/6$, respectively. Training and test set consist of 300 and 600 datapoints per class, respectively. 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), an adaptive matrix (GMLVQ), and individual adaptive matrices for every prototype (LGMLVQ). Relevance or matrix learning is done after an initial phase consisting of 500 epochs prototype adaptation. Training is done for 2000 epochs for GLVQ and GRLVQ, and for 6000 epochs for GMLVQ to account for the more subtle matrix adaptation. In all experiments, learning rates are chosen differently for prototypes and weight vectors resp. matrix elements, and the learning rates are annealed during training. The initial learning rate $\epsilon_p(0)$ for prototypes is chosen as 0.01, the initial learning rate for the diagonal relevance terms $\epsilon_d(0)$

Table 1: (a): Percentage of correctly classified patters for the artificial training and test set using the different LVQ-algorithms. (b): Percentage of correctly classified patterns for image data on training and test set.

(a)

Algorithm	Training	Test
GLVQ	75.33	71.83
GRLVQ	74.33	72.33
GMLVQ	79.67	77.83
LGRLVQ	81.0	78.0
LGMLVQ	91.67	90.75

(b)

Algorithm	Training	Test
GLVQ	82.38	79.05
GRLVQ	85.71	84.52
GMLVQ	91.9	87.86
LGRLVQ	90.0	89.05
LGMLVQ	96.19	94.29

is chosen as 0.005 and the initial learning rate for nondiagonal matrix elements $\epsilon_m(0)$ is chosen as 0.0001. For annealing, we use the following learning rate schedules:

$$\epsilon(t) = \frac{\epsilon(0)}{1 + \tau \cdot (t - t_{\text{start}})}$$

where t_{start} denotes the starting epoch for the adaptation, i.e. it is 1 for the weight adaptation and 500 for the adaptation of matrix elements and relevance factors. The parameter τ is chosen as 0.0001.

The classification accuracies on the training and test set are summarized in Tab.1(a). The position of the resulting prototypes and decision boundaries are shown in Fig.1(b)-(f). GMLVQ determines one single direction in feature space, which is used for classification. The resulting matrix Ω projects the data onto the respective subspace as depicted in Fig.1(g), Fig.1(h) denotes the projections of the classes using the LGMLVQ matrices

$$\Lambda_{\text{Class1}} = \begin{pmatrix} 0.5156 & -0.4997 \\ -0.4997 & 0.4844 \end{pmatrix} \qquad \Lambda_{\text{Class2}} = \begin{pmatrix} 0.7195 & 0.4492 \\ 0.4492 & 0.2805 \end{pmatrix}$$

One can clearly observe the benefit of individual matrix adaptation: this allows each prototype to shape its cluster according to the local ellipsoidal form of the class. This way, the data points of both cigar shaped 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 (see Fig. 1(d)) and, further, need no longer be convex in this case (see Fig. 1(f)).

5.2 Image Data

In a second experiment, the algorithm was applied to the image segmentation dataset provided in the UCI repository [15]. The dataset contains 19 dimensional feature vectors, which encode different attributes of 3x3 Pixel regions extracted out of seven outdoor images (brickface, sky, foliage, cement, window, path, grass). The features 3-5 are

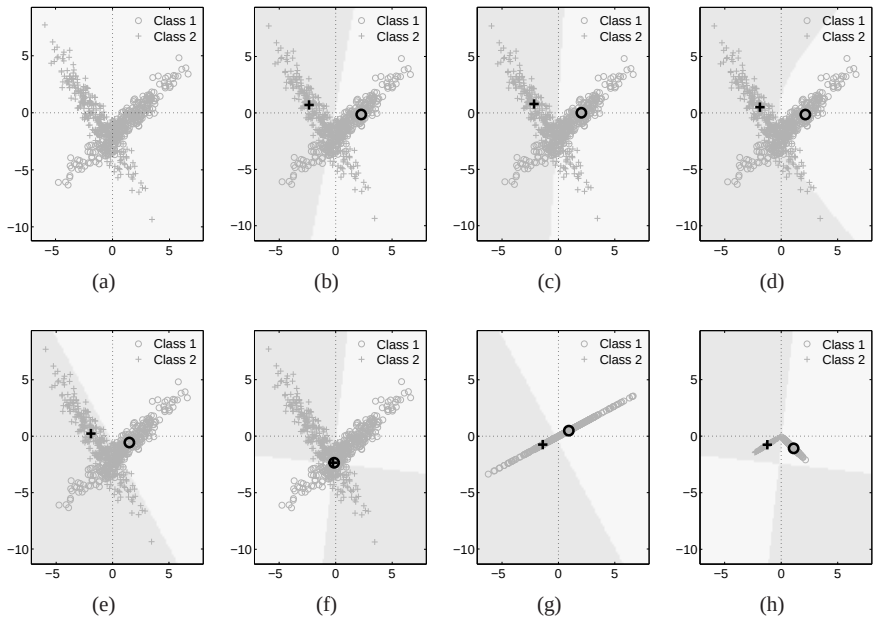

Figure 1: (a): Artificial dataset used for training. (b)-(f) Position of prototypes and decision boundaries after training with different LVQ-algorithms. (b):GLVQ, (c):GRLVQ, (d):LGRLVQ, (e):GMLVQ, (f):LGMLVQ, (g):data transformed by global matrix Ω, (h): data transformed by individual matrices for both prototypes.

(nearly) constant and were eliminated for this experiment. The training set consists of 210 datapoints (30 per class), the test sets contains 300 datapoints per class.

As beforehand, we used one prototype per class. In all experiments, the adaptation of metric parameters starts after an initial training phase for the prototypes consisting of a few 100 epochs. As beforehand, we compare global and local relevance resp. matrix adaptation and simple GLVQ. The initial learning rates have been optimized on the training set and they are chosen as follows:

- GLVQ: $\epsilon_p(0) = 0.8$
- GRLVQ: $\epsilon_p(0) = 0.8$, $\epsilon_d(0) = 5 \cdot 10^{-6}$
- LGRLVQ: $\epsilon_d(0) = 5 \cdot 10^{-5}$
- GMLVQ: $\epsilon_p(0) = 0.8$, $\epsilon_d(0) = 1 \cdot 10^{-4}$, $\epsilon_m(0) = 5 \cdot 10^{-5}$
- LGMLVQ: $\epsilon_p(0) = 0.8$, $\epsilon_d(0) = 1 \cdot 10^{-3}$, $\epsilon_m(0) = 5 \cdot 10^{-5}$

These learning rates are annealed as beforehand using $\tau = 0.001$.

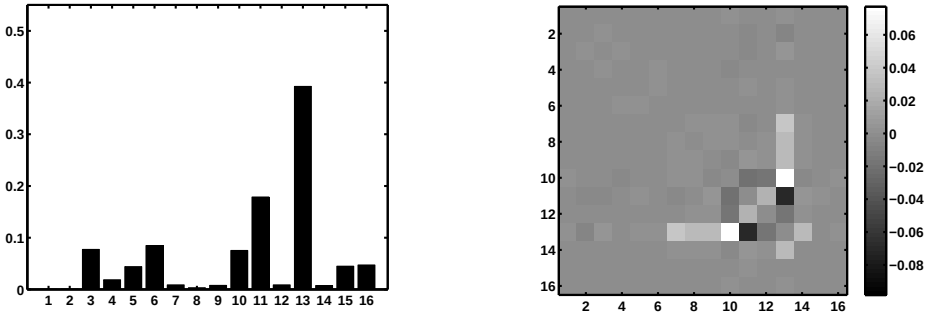

Figure 2: Visualization of the final relevance matrix for brickface-class. Left: diagonal elements. Right: nondiagonal elements

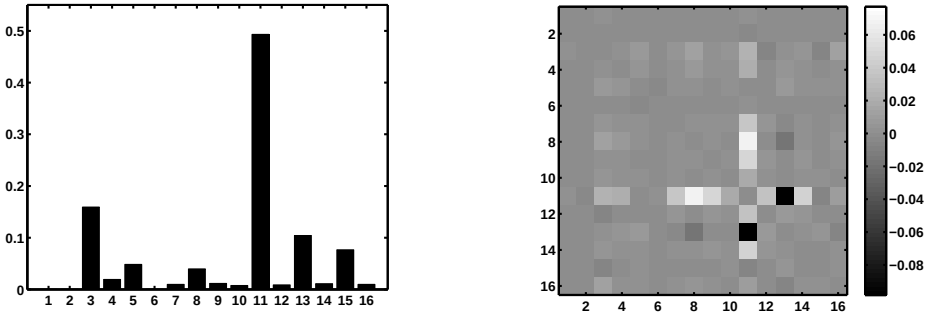

Figure 3: Visualization of the final relevance matrix for window-class. Left: diagonal elements. Right: nondiagonal elements

Figures 2 and 3 visualize the resulting matrices for the brickface- and the window class after training of individual matrices for all prototypes. It is visible that especially relations between the dimensions encoding color information are emphasized: The emphasized dimensions include 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), feature 11: exred-mean (2R - (G + B)), feature 12: exblue-mean (2B - (R + G)), feature 13: exgreen-mean (2G - (R + B)).

The classification accuracy can be observed in Table 1(b). Obviously, relevance and matrix adaptation allows to improve the classification accuracy. Thereby, local matrices yield an improvement of more than 10% compared to simple GLVQ. Remarkably, although the number of free parameters of the model is dramatically increased (being of order PN^2 for P prototypes and input dimensionality N) no overfitting takes place.

This demonstrates the good generalization ability of the model as substantiated by the formal generalization bounds.

6 Discussion

We have extended GRLVQ, a particularly efficient and powerful prototype based classifier, by a full matrix adaptation scheme. This allows the adaptation of the class borders such that local ellipsoidal shapes are taken into account. The possibility to improve the classification accuracy by this extension has been demonstrated in two examples. Remarkably, the generalization ability of the method is quite high as substantiated by theoretical findings.

The complexity of full local matrix adaptations scales with N^2 per epoch, N being the input dimensionality. This is better than comparable matrix adaptation methods as used e.g. in unsupervised fuzzy clustering [7, 9], however, the computational load becomes quite large for large input dimensionality. Therefore, specific schemes to shape the form of the matrix based on prior information are of particular interest. Nondiagonal matrix elements indicate a correlation of input features relevant for the classification. In many cases, one can restrict useful correlations due to prior knowledge. As an example, spatial or temporal data likely sho a high correlation of neighbored elements, whereas the other elements are probably independent. In such cases, one can restrict to a fixed adaptive bandwidth, decreasing the quadratic complexity to a linear term with respect to input dimensionality. Similarly, spatial correlations in images or functional data can lead to a massive restriction of the free parameters of the matrices to promising regions. This possibility will be the subject of future experiments.

References

[1] P.L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: risk bounds and structural risks. *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, to appear in *Journal of Machine Learning Research*, 2006.

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

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

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

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

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

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

[10] 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.

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

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

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

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

[15] 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.

[16] A.S. Sato and K. Yamada. Generalized learning vector quantization. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems*, volume 7, pages 423–429, MIT Press, 1995.

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

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