1. Metric Learning for Prototype-based classification

Michael Biehl[1], Barbara Hammer[2], Petra Schneider[1], and Thomas Villmann[3]

[1] University of Groningen, The Netherlands
[2] Clausthal University of Technology, Germany
[3] University of Leipzig, Germany

In this chapter, one of the most popular and intuitive prototype-based classification algorithms, learning vector quantization (LVQ), is revisited, and recent extensions towards automatic metric adaptation are introduced. Metric adaptation schemes extend LVQ in two aspects: on the one hand a greater flexibility is achieved since the metric which is essential for the classification is adapted according to the given classification task at hand. On the other hand a better interpretability of the results is gained since the metric parameters reveal the relevance of single dimensions as well as correlations which are important for the classification. Thereby, the flexibility of the metric can be scaled from a simple diagonal term to full matrices attached locally to the single prototypes. These choices result in a more complex form of the classification boundaries of the models, whereby the excellent inherent generalization ability of the classifier is maintained, as can be shown by means of statistical learning theory.

1.1 Introduction

An ubiquitous problem in machine learning is the inference of a classification model of given patterns into known classes; e.g. the classification of the outcome of several medical tests into different types of diseases, the classification of handwritten symbols into one of the letters 'a', ..., 'z', or the classification of photographs into objects depicted on these images. There exist numerous algorithms to automatically learn a classification given several training examples, ranging from logic-based approaches which model the class decisions by rules, statistical approaches which implement e.g. Bayesian inference, up to many classification algorithms connected to neural networks such as the simple perceptron or support vector machines [1.13].

Learning vector quantization (LVQ) has been introduced by Kohonen [1.19] as a prototype-based classification algorithm based on metric comparisons of data. It enjoys a great popularity due to several reasons: LVQ constitutes a classifier which is very intuitive since it represents classes by prototypical examples in the original data space, and classification takes place as an inference of the class of the respective closest prototype given a data point. Thus, the decision rules are interpretable and they can be inspected by experts in the field who can, for example, directly inspect and verify

prototypical examples used by LVQ for the classification. Learning rules are typically based on intuitive Hebbian learning, and the core algorithms can be implemented in just a few lines of code, i.e. implementation and realization of the algorithms is very simple. Unlike alternatives such as the perceptron or support vector machines which are restricted to only two classes in its basic form, LVQ can naturally deal with any number of given classes without making the classification rule or learning algorithm more complicated. Similarly, missing values do hardly constitute a problem for LVQ classifiers. Missing values are simply ignored, and only the known parts are compared to the LVQ prototype vectors; the prototypes, in turn, simply infer their values from all data points where the considered attribute is known.

Despite these advantages, LVQ faces several drawbacks in its original form which were overcome only recently. One major problem of original LVQ algorithms consists in the fact that the underlying learning rules are usually based on heuristics and a mathematical investigation of their learning behavior and generalization dynamics was lacking. This problem has recently been addressed by several approaches: first, alternative learning rules were proposed which derive LVQ type learning from an explicit cost function or statistical model, see e.g. [1.18, 1.29, 1.28]. This way, the objective which is optimized by the learning rules is stated explicitly and questions such as the learning dynamics and generalization behavior can be investigated based on the underlying cost function. Further, extensions of LVQ towards more adaptive parameters can be developed in a principled way by means of changing the cost function. Among several classical heuristic LVQ rules, we will introduce one important LVQ learning rule which has been derived from a cost function in this chapter.

As an alternative, several approaches try to mathematically investigate the learning dynamics and generalization behavior of LVQ rules (including heuristics) in typical model situations. The mathematical treatment becomes possible if certain assumptions are made (such as the fact that training patterns are independent and identically distributed, and the limit of infinite dimensionality is taken) such that powerful methods which stem from statistical physics can be applied. These methods yield very interesting results which demonstrate the ability (or inability) of several LVQ heuristics to learn the classification rules underlying specific simple though prototypical settings [1.5]. We will not consider these methods in this chapter, rather, we refer to the recent overview article [1.4].

As further possibility, the worst case generalization ability of LVQ-classifiers can be investigated using statistical learning theory, resulting in explicit bounds on the generalization behavior of the models which are independent of the underlying (unknown) data distribution [1.16]. Since these bounds hold for every possible input distribution, they are rather loose in general. However, they reveal the important parameters which influence the generalization behavior as well as the scaling of the error with respect to these

parameters. Interestingly, it can be shown that the bounds do not depend on the input dimensionality for LVQ networks, such that excellent generalization can be expected for LVQ networks also for high-dimensional data. In this chapter, we will shortly introduce this framework and we will explain a few relevant facts which can be proved within this framework for LVQ networks.

One major drawback of original LVQ algorithms with respect to practical applications consists in the limitation of LVQ to the standard Euclidean metric. In consequence, prototypes represent isotropic classes, and class boundaries are formed by hyperplanes which are perpendicular to the lines connecting the prototypes. This setting is inappropriate for practical applications: often, input dimensions are not scaled equally such that the relevance of the dimensions for the classification does not coincide. Despite this fact, the Euclidean metric scales every dimension equally, such that single dimensions can easily dominate the classification simply due to their inappropriate scaling. A related problem is particularly pronounced for high-dimensional data. Often, every dimension is disrupted by a small amount of noise, which accumulates in high dimensionality such that the overall classification can become meaningless. The situation might be even more complex since, in general, correlations of the data dimensions influence the classifications and should be taken into account for the decisions. Because of these aspects, generalizations of LVQ to more complex metric schemes have been proposed, one particularly relevant approach being the generalization to adaptive metrics which can set relevance terms of the input dimensions or even the data correlations according to the given classification task [1.18, 1.6]. For LVQ schemes which stem from a cost function, learning rules can be directly derived thereof taking the gradients, for example. We will introduce this intuitive and elegant scheme for general matrix learning in this chapter.

1.2 Learning vector quantization

Learning vector quantization (LVQ) as introduced by Kohonen [1.19] aims at learning a clustering given example data. Assume labelled data points $\{\boldsymbol{x}_i, y_i\} \in \mathbb{R}^n \times \{1, \ldots, C\}$ are given, the training set. An LVQ network consists of k prototypes $\boldsymbol{w}_i \in \mathbb{R}^n$ in the same space as the inputs, which are labelled by $c(\boldsymbol{w}_i) \in \{1, \ldots, C\}$. The prototypes define a classifier by means of a winner takes all rule: for a point $\boldsymbol{x} \in \mathbb{R}^n$ the output class is determined by

$$c(\boldsymbol{x}) := c(\boldsymbol{w}_i) \text{ such that } \boldsymbol{w}_i = \operatorname{argmin}_{\boldsymbol{w}_j} d(\boldsymbol{x}, \boldsymbol{w}_j) \tag{1.1}$$

where $d(\boldsymbol{x}, \boldsymbol{w}_j)$ usually denotes the squared Euclidean distance

$$d(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{i=1}^{n} ([\boldsymbol{x}]_i - [\boldsymbol{w}_j]_i)^2 \,. \tag{1.2}$$

The subscript $[\cdot]_i$ refers to the ith component of the vectors. The receptive field of prototype $\boldsymbol{w}_i$ is defined as the set of points which pick this prototype as their winner. Obviously, LVQ defines a classification which is constant on the receptive fields of the prototypes.

The goal of learning is to adapt prototypes automatically such that the class label of data points in the receptive fields coincide with the label of the respective prototype. Hence the classification error

$$|\{\boldsymbol{x}_i \mid c(\boldsymbol{x}_i) \neq y_i\}| \tag{1.3}$$

should be minimized. Different strategies to achieve this goal have been proposed. LVQ1 implements Hebbian learning as a direct heuristic. Prototypes are initialized randomly. Afterwards, training points $\boldsymbol{x}_i$ are presented repeatedly in random order, and the location of the prototype $\boldsymbol{w}_j$ which becomes winner for $\boldsymbol{x}_i$ is adapted according to the following rule:

$$\boldsymbol{w}_j = \begin{cases} \boldsymbol{w}_j + \eta \cdot (\boldsymbol{x}_i - \boldsymbol{w}_j) & \text{if } c(\boldsymbol{w}_j) = y_i \\ \boldsymbol{w}_j - \eta \cdot (\boldsymbol{x}_i - \boldsymbol{w}_j) & \text{if } c(\boldsymbol{w}_j) \neq y_i \end{cases} \tag{1.4}$$

Interestingly, this simple learning rule leads to a quite efficient algorithm which displays remarkable results. Kohonen proposed a few alternatives which account for a faster convergence or better adaptation of the decision boundaries. One popular alternative, for example, is given by LVQ2.1, which adapts the two closest prototypes in every step as follows: if the closest two prototypes $\boldsymbol{w}_{j_1}$ and $\boldsymbol{w}_{j_2}$ belong to different classes and they fall into a window near the decision boundary, simultaneous adaptation of both vectors $\boldsymbol{w}_{j_1}$ and $\boldsymbol{w}_{j_2}$ according to the LVQ1 learning rule takes place. Otherwise, no adaptation takes place. Further variants include, for example optimized LVQ which has a data-adapted learning rate for fast convergence, or LVQIII which is often used to further tune the classification borders after initial training with one of the other methods.

All LVQ schemes as introduced above have the drawback that their learning rule is heuristically motivated, thus, their dynamical behavior and convergence properties are not clear. One might attempt to derive LVQ1 or LVQ2.1 as gradient descent methods of cost functions by formally integrating the models. By symbolic integration, interpreting the learning rule (1.4) as a stochastic gradient descent, we obtain the following cost term

$$E_{\mathrm{LVQ1}} = \frac{1}{2} \sum_{i,j} \delta_{N(\boldsymbol{x}_i),\boldsymbol{w}_j} \cdot (-1)^{1+\delta_{y_i,c(\boldsymbol{w}_j)}} \cdot d(\boldsymbol{x}_i, \boldsymbol{w}_j) \tag{1.5}$$

for LVQ1, where $\delta_{i,j}$ denotes the Kronecker delta-symbol and $N(\boldsymbol{x}_i)$ refers to the prototype closest to $\boldsymbol{x}_i$. For LVQ2.1, formal integration yields the formula

$$E_{\mathrm{LVQ2.1}} = \frac{1}{2} \sum_{i,j,k} \delta_{N^+(\boldsymbol{x}_i),\boldsymbol{w}_j} \delta_{N^-(\boldsymbol{x}_i),\boldsymbol{w}_k} \cdot 1_W(\boldsymbol{w}_j, \boldsymbol{w}_k) \cdot (d(\boldsymbol{x}_i, \boldsymbol{w}_j) - d(\boldsymbol{x}_i, \boldsymbol{w}_k))$$

(1.6)

where $N^+(\boldsymbol{x}_i)$ refers to the closest prototype to $\boldsymbol{x}_i$ with class label y_i and $N^-(\boldsymbol{x}_i)$ refers to the closest prototype to $\boldsymbol{x}_i$ with a class label different from the label y_i. $1_W(\boldsymbol{w}_j, \boldsymbol{w}_k)$ implements the window rule of LVQ2.1.

Note that, for both cases, the gradient of the cost function is not well defined if points $\boldsymbol{x}_i$ lie at the boundaries of receptive fields. Thus, the cost function cannot directly be extended towards the case of an underlying smooth distribution of data points in a real vector space according to some density function. One can easily see that, for the cost function (1.5), such an extension cannot exist in principle, since the function would be discontinuous at borders of receptive fields. Further, the discrete cost function (1.5) is only locally a valid cost function, since the overall value of $E_{\mathrm{LVQ2.1}}$ is smaller if there exist many misclassified points compared to a correct classification. Compared to this fact, the cost function (1.6) allows an extension towards a continuous input distribution by means of an interpretation of the indicator functions by means of delta-functions. Optimizing this cost term directly, however, shows instabilities and divergence due to the unboundedness of this cost function, such that, in practice, a subtle tuning by means of the window rule becomes necessary [1.22].

Several researchers have proposed alternative LVQ schemes which are directly derived from an appropriate cost function, such that convergence and learning dynamics are explicitly stated in terms of the objective, see e.g. [1.28, 1.29, 1.23, 1.18, 1.17]. We will focus on the approach [1.23] and generalizations thereof, since it provides a very convenient extension of LVQ2.1. The cost function underlying generalized LVQ (GLVQ) as introduced in [1.23] has the form

$$E_{\mathrm{GLVQ}} = \frac{1}{2} \sum_{i,j,k} \delta_{N^+(\boldsymbol{x}_i),\boldsymbol{w}_j} \delta_{N^-(\boldsymbol{x}_i),\boldsymbol{w}_k} \cdot \Phi \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{w}_j) - d(\boldsymbol{x}_i, \boldsymbol{w}_k)}{d(\boldsymbol{x}_i, \boldsymbol{w}_j) + d(\boldsymbol{x}_i, \boldsymbol{w}_k)} \right) \quad (1.7)$$

where $\Phi : \mathbb{R} \to \mathbb{R}$ is a strictly monotonic increasing function such as the identity or the logistic function. This cost function involves the same term as LVQ2.1, the comparison $d(\boldsymbol{x}_i, \boldsymbol{w}_j) - d(\boldsymbol{x}_i, \boldsymbol{w}_k)$ which is negative iff the classification of $\boldsymbol{x}_i$ is correct, and which is smaller the larger the difference of the distance to the closest correct prototype versus the closest wrong prototype. Unlike LVQ2.1, this cost term is scaled by means of the denominator such that summands in $(-1, 1)$ arise. These are possibly subject to a further nonlinear transformation Φ – however, it is also possible to choose Φ as the identity.

It has been shown in [1.18] that the definition (1.7) constitutes a valid cost function also in the case of an underlying continuous smooth input distribution. In this case, the borders of receptive fields are realized by means of delta functions and the computation leads to update rules which are valid for every possible data point. The general learning rule can be derived from

(1.7) by means of a stochastic gradient descent. Prototypes are initialized randomly and the repeated presentation of data points $\boldsymbol{x}_i$ give rise to the update of the closest prototype $\boldsymbol{w}_j$ with the same label as $\boldsymbol{x}_i$ and the prototype $\boldsymbol{w}_k$ with a different label than $\boldsymbol{x}_i$ as follows:

$$\boldsymbol{w}_j = \boldsymbol{w}_j + \eta \cdot \Phi' \cdot \mu^+(\boldsymbol{x}_i) \cdot \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} \tag{1.8}$$

$$\boldsymbol{w}_k = \boldsymbol{w}_k - \eta \cdot \Phi' \cdot \mu^-(\boldsymbol{x}_i) \cdot \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{w}_k)}{\partial \boldsymbol{w}_k} \tag{1.9}$$

where Φ' is evaluated at $(d(\boldsymbol{x}_i, \boldsymbol{w}_j) - d(\boldsymbol{x}_i, \boldsymbol{w}_k))/(d(\boldsymbol{x}_i, \boldsymbol{w}_j) + d(\boldsymbol{x}_i, \boldsymbol{w}_k))$, the factor $\mu^+(\boldsymbol{x}_i)$ equals $2d(\boldsymbol{x}_i, \boldsymbol{w}_k)/(d(\boldsymbol{x}_i, \boldsymbol{w}_j) + d(\boldsymbol{x}_i, \boldsymbol{w}_k))^2$, the factor $\mu^-(\boldsymbol{x}_i)$ equals $2d(\boldsymbol{x}_i, \boldsymbol{w}_j)/(d(\boldsymbol{x}_i, \boldsymbol{w}_j) + d(\boldsymbol{x}_i, \boldsymbol{w}_k))^2$, and the derivative of the squared Euclidean metric is

$$\frac{\partial d(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} = -2(\boldsymbol{x} - \boldsymbol{w})\,. \tag{1.10}$$

Note the similarity of this cost function to the LVQ2.1 update rule: instead of using a window technique, the LVQ2.1 updates are scaled using the factors $\mu^+(\boldsymbol{x}_i)$ and $\mu^-(\boldsymbol{x}_i)$, respectively, which stem from the denominator of the summands of the cost function. It has been demonstrated e.g. in [1.23, 1.22] that a robust convergence and classification accuracy can be obtained this way provided a suitable function Φ is chosen.

A simple example of a GLVQ network is displayed in Fig. 1.1: three multimodal classes in two dimensions are trained using two prototypes per class, random initialization of the prototypes, constant learning rate $\eta = 0.01$, and 500 epochs. Fig. 1.1 shows the final prototype locations and the receptive fields, resulting in a classification accuracy of 98.89%. This excellent classification accuracy can be obtained because the setting can be described by means of prototypes and their receptive fields based on Euclidean distances. Fig.1.1 (right) displays the behavior of GLVQ if the two dimensional data set is embedded in ten dimensions the following way: $(x_1, x_2) \mapsto (x_1, x_2, x_1 + \eta_1, x_1 + \eta_2, x_1 + \eta_3, x_1 + \eta_4, \eta_5, \eta_6, \eta_7, \eta_8)$ where η_i refers to noise with increasing variance. In this case, data are disrupted by noise which does not contribute to the classification accuracy. It can be expected that the classification accuracy decreases since the standard Euclidean metric cannot account for this fact. Indeed, the classification accuracy drops to at most 83.33% due to the included noise, and the result of several runs shows small differences due to the fact that several local optima of the optimized cost function exist for this setting. In Fig. 1.1, the projection of the resulting classifier to the first two dimensions is depicted and receptive fields in these two dimensions are computed.

This observation suggests that it would be fruitful to change the underlying metric in this classification task. The general formulation of LVQ learning schemes in terms of cost functions has the benefit that the integration and adaptation of additional model parameters besides the prototype locations

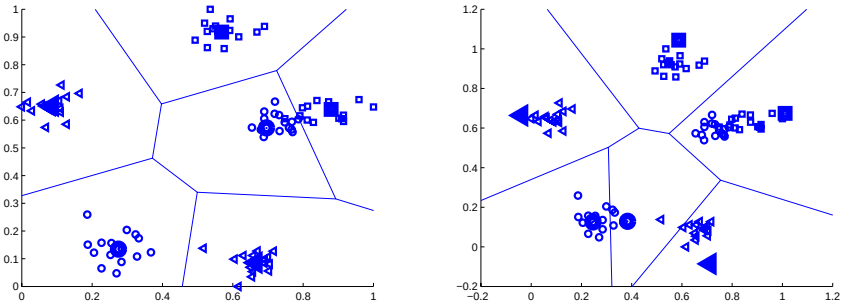

Fig. 1.1. Left: A simple two-dimensional data set which can easily be learned with LVQ. The result of a GLVQ run is depicted. Right: If the same data set is embedded into ten dimensions by adding noise, the classification accuracy of GLVQ decreases from 98.89% to only 83.33%, as depicted on the right image by projecting onto the first two relevant dimensions.

becomes easily possible, since a well-defined interface is offered by means of the cost function. This possibility has been used to integrate metric adaptation into the learning schemes. As seen in the previous example, a wrong choice of the metric, e.g. induced by useless noisy dimension can severely disrupt the classification capacity of GLVQ networks, thus metric adaptation becomes necessary.

1.3 Metric learning

One drawback of LVQ classification schemes given by (1.1) consists in the restriction of the metric to the standard squared Euclidean distance (1.2). This restricts the classes to decision boundaries which stem from isotropic classes, i.e. the decision boundaries are given by hyperplanes which are perpendicular to the lines connecting the prototypes. This setting is problematic if dimensions are not scaled properly. Further, numerical problems might occur for LVQ schemes if huge dimensionality has to be dealt with and noise accumulates. It has been discussed e.g. in [1.20] that Euclidean distances become more and more meaningless if the dimensionality of the data increases. This problem is partially prevented by LVQ schemes which consider only differences of distances. Nevertheless, when irrelevant and noisy data is included, it can become essential to identify the relevant dimensions for a given classification task. Because of this reason, much research exists on methods to adapt the metric according to the given classification at hand, such as pruning methods (see e.g. [1.14, 1.32, 1.15]) or a direct adaptation of the metric used for classification (see e.g. [1.36]).

A few heuristics to adapt the metric for LVQ schemes have been proposed e.g. in [1.8, 1.21]. Here, we present a more principled approach which has been

introduced in [1.18] and later been extended in [1.6] to adapt the metric based on a formulation of LVQ learning in terms of a cost function. The Euclidean distance (1.2) is substituted by a more general form such as

$$d_\lambda(\boldsymbol{x}, \boldsymbol{w}_j) = \sum_{i=1}^{n} \lambda_i ([\boldsymbol{x}]_i - [\boldsymbol{w}_j]_i)^2 \tag{1.11}$$

where $\lambda_i \geq 0$ with $\sum_i \lambda_i = 1$ or

$$d_\Lambda(\boldsymbol{x}, \boldsymbol{w}_j) = (\boldsymbol{x} - \boldsymbol{w}_j)^t \cdot \Lambda \cdot (\boldsymbol{x} - \boldsymbol{w}_j) \tag{1.12}$$

where Λ is a $n \times n$ symmetric and positive semidefinite matrix with $\sum_i [\Lambda]_{ii} = 1$. Note that definiteness can be enforced by setting $\Lambda = \Omega\Omega^t$, for example, the regularization then corresponds to $\sum [\Lambda]_{ii} = \sum_{ij} [\Omega]_{ij}^2 = 1$. $\Omega \in \mathbb{R}^{n \times n}$ has the same rank as Λ. These choices provide valid metrics, whereby the diagonal metric (1.11) introduces relevance terms which can weight the separate dimensions independently according to their contribution for the general learning task. In particular, useless dimensions can be dropped by setting the corresponding relevance to a small value. The full matrix (1.12) refers to a general metric which, besides appropriate scaling, can take correlations of the data dimensions into account. It provides a general linear transformation Ω of the data. This way, the shapes of receptive fields do no longer stem from a metric with isotropic isobars, rather, isobars have the form of achses aligned resp. arbitrary ellipsoids.

The transformations provided by these more general metrics can either be assigned globally to the whole data space, or they can be applied locally to the distance calculation of every specified prototype. In the latter case, the metric becomes parameterized as

$$d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j) = (\boldsymbol{x} - \boldsymbol{w}_j)^t \cdot \Lambda_j \cdot (\boldsymbol{x} - \boldsymbol{w}_j) \tag{1.13}$$

where $\Lambda_j = \Omega_j \Omega_j^t$ constitutes a positive semidefinite matrix which is now attached to the prototype $\boldsymbol{w}_j$. Note that, this way, local matrix adaptation becomes possible, and cluster boundaries obtain a more general form described by quadratic equations. In the general setting, convexity or even connectedness of receptive fields need no longer hold. We will consider this most general setting in the following.

The question occurs how metric parameters can be determined to give an optimum classification accuracy. Since we have considered LVQ schemes which are derived from an explicit cost function, the derivation of explicit learning rules is rather straightforward: A more general metric such as e.g. (1.13) can directly be included into the update rule for prototypes, eqn. (1.9), where the more general metric leads to the derivative

$$\frac{\partial d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j)}{\partial \boldsymbol{w}_j} = -2\Lambda_j \cdot (\boldsymbol{x} - \boldsymbol{w}_j) \tag{1.14}$$

Further, the updates for the prototypes are accompanied by updates for the metric parameters Ω_j and Ω_k assigned to the closest correct and wrong prototype, respectively, as follows:

$$\Delta\Omega_j \sim -\Phi' \cdot \mu^+(\boldsymbol{x}_i) \cdot \frac{\partial d_{\Lambda_j}(\boldsymbol{x}_i, \boldsymbol{w}_j)}{\partial \Omega_j} \tag{1.15}$$

$$\Delta\Omega_k \sim \Phi' \cdot \mu^-(\boldsymbol{x}_i) \cdot \frac{\partial d_{\Lambda_k}(\boldsymbol{x}_i, \boldsymbol{w}_k)}{\partial \Omega_k} \tag{1.16}$$

where the quantities Φ', μ^+ and μ^- are as beforehand, using the more general metric, and the derivative of the distance with respect to the matrix yields

$$\frac{\partial d_{\Lambda_j}(\boldsymbol{x}, \boldsymbol{w}_j)}{\partial (\Omega_j)_{lm}} = 2 \cdot \left[\Omega_j^t(\boldsymbol{x} - \boldsymbol{w}_j)\right]_m ([\boldsymbol{x}]_l - [\boldsymbol{w}_j]_l) \tag{1.17}$$

See e.g. [1.6] for the exact derivation of this update rule and [1.17] for a proof that this rule is valid also for a smooth continuous underlying input distribution. This gives the update rule for the most general case, adaptation of individual full matrices attached to the prototypes. The corresponding learning algorithm is referred to as local generalized matrix LVQ (local GMLVQ). The update for one global matrix follows thereof by summation, referred to as GMLVQ. Similarly, the update of a simple diagonal matrix can be obtained thereof reducing the general matrix Λ_j to diagonal form, referred to as (local) generalized relevance LVQ ((local) GRLVQ). In all cases we require $\sum_i [\Lambda_j]_{ii} = 1$ for all j to prevent degeneration towards the meaningless global optimum given by a vanishing matrix. This constraint is simply realized by an explicit normalization after every update step.

Note that this generalized learning rule for the matrix parameters can be interpreted in the standard Hebbian style if a diagonal matrix is present: dimensions are decreased according to the squared distance of a data point to the closest correct prototype in every dimension, and increased according to the squared distance of the closest wrong prototype in every dimension. Taking normalization into account, this is a reinforcement of the dimensions where data point and correct prototype are close together, resp. a weakening of the dimensions where the wrong prototype and the data point are close together. For full matrix update, the correlations of the dimensions of the transformed data and the considered training point are also taken into account.

We would like to demonstrate the increased classification power of these more general LVQ forms in a simple example. First, we look again at the data displayed in Fig. 1.1, embedded in ten dimensions, as before. Instead of the standard Euclidean metric, we use a global diagonal metric as introduced in Eqn. (1.11) with adaptive relevance parameters for the separate dimensions. This way, the noise can be suppressed by the LVQ classifier by setting the relevance of the corresponding dimensions to small values. Training using the same parameters as before and the learning rate 0.001 for the relevance terms

leads to a classification accuracy of 98.89%, which is the same accuracy as for the original two dimensional data set. The relevance profile of the classifier is depicted in Fig. 1.2. As can be seen clearly, the relevance profile effectively reduces the problem to the original two dimensional problem, such that the same accuracy as for the simple two-dimensional setting can be achieved.

Next, we consider the even more complicated data set which is obtained by transforming the ten dimensional data by a random matrix in $(0,1)^{10\times 10}$. This way, the relevant dimensions for classification are no longer aligned with coordinate axes. In consequence, GRLVQ using the same parameters as beforehand yields to a decreased accuracy of about 75%, the best performance being around 77.78%, depending on the initialization of the prototypes. Unlike the previous setting, several local optima of the cost function exist, and a different optimum is found depending on the run. GLVQ without metric adaptation achieves only about 67.78% accuracy. Interestingly, as depicted in Fig. 1.2 (right), one solution found by GRLVQ emphasizes only few dimensions of the data, hence a very simple solution of the problem is favored by the algorithm which can increase the classification accuracy in comparison to simple GLVQ.

We apply full matrix adaptation for the same data set, using the same training parameters. The achieved classification accuracy of 94.44% approximates the classification accuracy of the original data set, since GMLVQ can rotate the data arbitrarily in Euclidean space, thus effectively reducing the data set to the original one. By the construction of the method, the matrix Ω found by GMLVQ constitutes one transformation of the data set such that standard LVQ becomes easier in the projected space. Interestingly, the found transformation clearly indicates that a two dimensional subspace is used for classification since the eigenvalue profile is almost zero for all but the largest

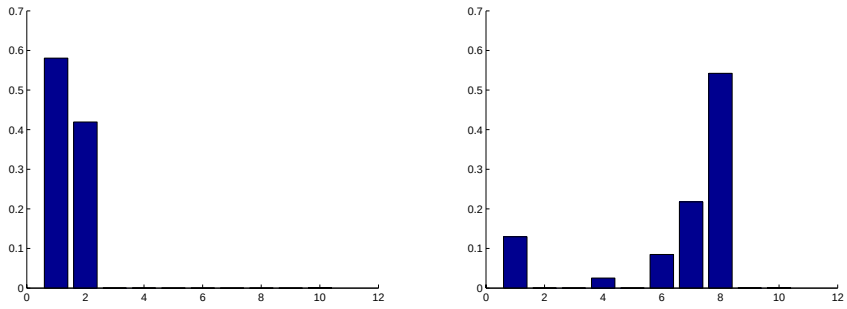

Fig. 1.2. Left: Relevance profile learned by LVQ with diagonal metric adaptation, obviously, the problem is almost reduced to the original two dimensional problem because of $\lambda_i \approx 0$ for all $i \geq 3$. Right: Relevance profile for the ten dimensional problem transformed by a random matrix in $\mathbb{R}^{10\times 10}$. Interestingly, one major component (8) is identified and data are projected to only few dimension.

two eigenvectors, see Fig. 1.3 (right). The projection of the data and prototypes by means of this matrix Ω is depicted in Fig. 1.3 exemplarily for dimensions 3 and 10 of the data. Obviously, the original data are found up to a rotation and scaling. Note that the transformation Ω is not unique and different equivalent projections can be found. Correspondingly, projections onto alternative dimensions either look qualitatively the same like Fig.1.3, or they display a strong correlation of the considered dimensions, indicating that the original data are recovered and stored in several of the projected points. It would be possible to obtain a unique representation by referring to the unique positive and symmetric root of Λ instead of Ω, which gives qualitatively the same image as depicted in Fig. 1.1.

The presented examples benefit from a global adaptation of the metric to take scaling and correlations of the axes into account, resulting in an adaptive global linear transformation of the data such that the classification becomes easier in the projected space. Geometrically, the transformations correspond to axis aligned or general ellipsoidal isobars of the clusters, instead of only spherical shapes. In practical applications, it is often the case that a specific scaling behavior is valid only locally around given prototypes, and relevant data correlations differ in different regions of the input space. In such settings, ellipsoidal cluster shapes with different main axes centered around the prototypes are needed. This situation can be taken into account by LVQ schemes if the metric parameters are attached to the different prototypes, and every prototype is allowed to determine the local metric individually. As a consequence, the receptive fields are no longer determined by global transformations, but local transformations, i.e. the separating borders are determined by a quadratic equation rather than a linear one, and the receptive fields need no longer be convex nor connected, see e.g. [1.6] for a very intuitive example which demonstrates this effect.

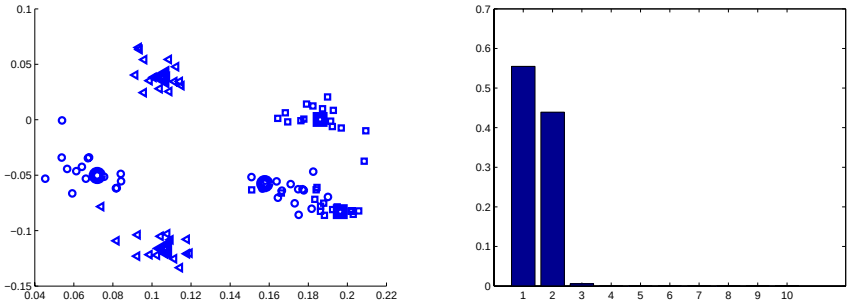

Fig. 1.3. Left: Data and prototypes found by GMLVQ when projected back using the trained matrix Ω, displaying dimensions 3 and 10. Obviously, the original data is recovered up to scaling and rotation this way. Right: Profile of the eigenvalues found by GMLVQ for the transformed data set.

LVQ with matrix adaptation has successfully been applied to a number of practical applications. Besides benchmark scenarios reported e.g. in [1.25], LVQ with local or global matrix adaptation has been used for the analysis of satellite remote sensing data [1.35], medical image processing [1.1], classification of gene expressions measured by micro- and macroarray data, respectively [1.3, 1.31], content based image retrieval [1.9], analysis of mass spectra in the frame of clinical proteomics [1.24], online object segmentation in digital images [1.12], or the supervision of technical systems such as piston engines [1.7]. In particular for applications in biology and medicine, the explicit interpretability of the found matrices in terms of relevance factors and relevant data correlations is thereby of particular importance. It opens the way towards efficient feature construction in image analysis, and identification of potential biomarkers when analyzing mass spectra, for example.

1.4 Generalization ability

When dealing with classification tasks, the goal is usually not to achieve a small classification error on the training set, rather the underlying regularity for the classification of arbitrary data points should be learned. Therefore the generalization ability of classification schemes is usually estimated by means of the classification error of data not used for training, or by means of statistical estimators with less variance such as crossvalidation in practical applications [1.13]. Nevertheless, it is relevant to guarantee that a classification method can generalize to unseen examples in principle – this does not necessarily hold for arbitrary algorithms (such as e.g. algorithms which only learn by heart using a table look-up).

Further, it is interesting to know which parameters influence the generalization error of the classifier. For LVQ, there exists a couple of free parameters such as the number of prototypes, the data dimensionality, and the complexity of the adaptive metric. It would be worthwhile to know whether these parameters influence the generalization ability on the same scale, requiring the same amount of training examples to fix the corresponding free parameters in a given training setting. Note that it does in general not hold that the generalization ability and the number of necessary examples for valid generalization scales linearly (or even polynomially) with the number of free parameters of a classifier: two counterexamples are the support vector machine, which generalizes well independent of the input dimensionality of the classifier (which determines the number of free parameters), rather, the so-called margin is the relevant quantity for characterizing the generalization ability in this case [1.34]. Conversely, there exist simple function classes with only one parameter, such as the class $\{x \mapsto \cos(tx) | t \in \mathbb{R}\}$ which do not allow valid generalization in the classical sense of PAC learnability [1.33, 1.30].

LVQ networks show an astonishing robustness towards overfitting. Even if the number of prototypes is chosen higher than necessary, this does hardly

decrease the generalization ability since, usually, the prototypes just share the prototypical positions in the data space due to the underlying Hebbian learning rule. This observation has first been formalized in [1.11], where the generalization ability of simple LVQ networks has been characterized using computational learning theory. These results were later extended towards more general schemes including adaptive metrics e.g. in [1.16, 1.6]. Here we introduce the setting formalized within statistical learning theory and we give a short overview of the most important bounds for LVQ networks. The overall argumentation follows the general theory of large margin classifiers as laid out in [1.2] and adapted to LVQ networks in [1.6].

Assume that a LVQ networks is given with k prototypes $\boldsymbol{w}_i$ and inputs in $\mathbb{R}^n$. For simplicity, we restrict to the case of binary classifications, i.e. class labels $\{-1, 1\}$ are considered. We refer to prototypes labeled by -1 by $\boldsymbol{w}_i^-$ and to prototypes labeled by 1 by the notation $\boldsymbol{w}_i^+$. Classification of a LVQ network takes place by a winner takes all rule, which can be written as

$$\boldsymbol{x} \mapsto f(\boldsymbol{x}) = \mathrm{sgn}\left(\min_{\boldsymbol{w}_i^-}\{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^-)\} - \min_{\boldsymbol{w}_i^+}\{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^+)\}\right) \tag{1.18}$$

where d_{Λ_i} refers to the (possibly matrix weighted and local) distance measure, and sgn selects the sign ± 1 of the overall value.

In the following argumentation, we will not be interested in the question of how exactly a certain LVQ network is trained. Rather, the role of the learning algorithm is characterized by the fact that the training error is small for a given training set. We are interested in the question whether this fact implies that the error for arbitrary, possibly unseen data points is also small. Assume that the underlying regularity which should be learned is described by an (unknown) probability distribution P on $\mathbb{R}^n \times \{-1, 1\}$. Then the goal of learning is to achieve a small generalization error

$$E_P(f) := P(y \neq f(x)) \tag{1.19}$$

where f is the function implemented by the LVQ network as denoted in (1.18). Since P is not known, the empirical error

$$\hat{E}_m(f) := \sum_{i=1}^{m} |\{y_i \neq f(\boldsymbol{x}_i)\}| \tag{1.20}$$

is usually minimized during training. Thereby, it is assumed that the training set $\{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, m\}$ is representative for the unknown regularity, i.e. the examples are drawn independently and identically distributed according to P. The capability of LVQ of generalizing to new data means that a small empirical error (1.20) also includes a small generalization error (1.19).

Obviously, the empirical error would be representative for the real error if the data were not used to infer the function f, e.g. $\boldsymbol{x}_i$ denote data from a test

set. In our situation, however, it is assumed that the data $(\boldsymbol{x}_i, y_i)$ are used for training and the function f is chosen such that the empirical error on the training data becomes small. Therefore, f depends on the given data, and it will change e.g. if the training set is enlarged. The trick to obtain bounds also in this setting relies on the derivation of uniform bounds of the deviation of the empirical error and the real error which hold simultaneously for every possible function f implemented by a LVQ network.

For this purpose, we specify the class of functions which can be realized by a LVQ network with p prototypes:

$$\mathcal{F} = \{f : \mathbb{R}^n \to \{-1, 1\} \mid f \text{ is defined by Eq. 1.18}\} \tag{1.21}$$

Further, we assume that inputs are restricted to a ball $|\boldsymbol{x}| \leq B$ for some $B > 0$, and, correspondingly, prototypes also fulfill $|\boldsymbol{w}_i| \leq B$. Further, we assume that Λ_i is symmetric and positive semidefinite for every i with $\sum_j [\Lambda_i]_{jj} = 1$. We refer to this restricted class also by the symbol $\mathcal{F}$. We are interested in bounds of the form $E_P(f) \leq \hat{E}_m(f) + \epsilon(m)$ which hold simultaneously with high probability for every $f \in \mathcal{F}$ and sample drawn i.i.d. according to P. Thereby, $\epsilon(m)$ will include the relevant parameters of the learning method such as the number of prototypes for LVQ.

Similar to corresponding bounds for support vector machines, we do not directly derive bounds of this form which depend on $\mathcal{F}$, rather, we look at a slight modification which also takes the security of classifications provided by $f \in \mathcal{F}$ into account. We define the following real-valued function which is obtained from f as given in 1.18 by dropping sgn:

$$M_f : \boldsymbol{x} \mapsto \left(\min_{\boldsymbol{w}_i^-} \{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^-)\} - \min_{\boldsymbol{w}_i^+} \{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^+)\} \right) \tag{1.22}$$

The sign of this function determines the output class. Simultaneously, the absolute value of this function indicates the security or margin of the classification for the input $\boldsymbol{x}$. The larger this margin, the more robust is the classification of $\boldsymbol{x}$ with respect to changes or noise in the input and classification parameters, since $|M_f(\boldsymbol{x})|$ gives the difference of the distance of $\boldsymbol{x}$ from the closest correct versus the closest wrong prototype. We refer to the function class defined in analogy to 1.21 by $M_{\mathcal{F}}$.

This margin can be integrated into the empirical error. The empirical error 1.20 only counts points which are misclassified by f. The extension towards M_f gives us the opportunity to judge correct but insecure classifications differently. Thus, we can take into account a margin of the classification which should be obeyed by the LVQ function. Formally, we fix some $\rho > 0$, the margin accepted by the classification, and define the associated loss function

$$L : \mathbb{R} \to \mathbb{R}, t \mapsto \begin{cases} 1 & \text{if } t \leq 0 \\ 1 - t/\rho & \text{if } 0 < t \leq \rho \\ 0 & \text{otherwise} \end{cases} \tag{1.23}$$

We can extend the empirical error 1.20 correspondingly as

$$\hat{E}_m^\rho(f) := \sum_{i=1}^{m} L(y_i \cdot M_f(\boldsymbol{x}_i))/m \tag{1.24}$$

This term accumulates the misclassified points and also punishes all correct classifications if their margin is smaller than ρ.

As shown in [1.2], it is in general possible to bound the deviation of the real error and this extended empirical error with probability at least $1 - \delta/2$ for $\delta \in (0, 1)$ uniformly for every $f \in \mathcal{F}$ as follows:

$$E_P(f) \leq \hat{E}_m^\rho(f) + \frac{2}{\rho} \cdot R_m(M_\mathcal{F}) + \sqrt{\frac{\ln(4/\delta)}{2m}} \tag{1.25}$$

if the m samples are drawn i.i.d. with respect to P. The key quantity of this bound is the so-called Rademacher complexity $R_m(M_\mathcal{F})$ of the function class $M_\mathcal{F}$ defined by LVQ classifiers, a measure of the richness of this function class. The richer the function class, the more degrees of freedom have to be specified by the training examples and, in consequence, the larger the bounds on a possible deviation of the empirical and real error. The Rademacher complexity is defined as

$$R_m(M_\mathcal{F}) = E_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m} E_{\sigma_1,\ldots,\sigma_m} \left(\sup_{M_f \in M_\mathcal{F}} \left| \frac{2}{m} \sum_{i=1}^{m} \sigma_i \cdot M_f(\boldsymbol{x}_i) \right| \right) \tag{1.26}$$

where σ_i are independent uniform $\{-1, 1\}$-valued random variables, and expectation of $\boldsymbol{x}_i$ is taken with respect to the marginal distribution induced by P on $\mathbb{R}^n$. This measure, in essence, counts the fraction of cases in which, on average, a random classification of m data points (realized by σ_i) can be implemented by a function in $M_\mathcal{F}$ such that the signs coincide and the absolute values are as large as possible.

It is possible to find explicit bounds on the Rademacher complexity of LVQ function classes as specified in 1.22 using techniques established in [1.2] as detailed in [1.26]. We obtain the overall bound

$$R_m(M_\mathcal{F}) \leq \mathcal{O}\left(\frac{k^2 B^3 + \sqrt{\ln(1/\delta)}}{\sqrt{m}} \right) \tag{1.27}$$

with probability at least $1 - \delta/2$ where k refers to the number of prototypes and B the maximum size of the inputs and prototypes. Thus, the overall bound

$$E_P(f) \leq \hat{E}_m^\rho(f) + \frac{1}{\sqrt{m}} \mathcal{O}\left(\frac{k^2 B^3}{\rho} + \frac{\sqrt{\ln(1/\delta)}}{\min\{1, \rho\}} \right) \tag{1.28}$$

results which holds with probability at least $1 - \delta$ simultaneously for every function $f \in \mathcal{F}$. Since this is valid for every such function f, prototypes

as well as local matrix parameters can be adapted according to the given training setting without affecting the bound. I.e. the bound also holds in the most general setting of complete local adaptive matrices of LVQ classifiers.

In principle, the bound states that the difference of the real error and the extended empirical error, taking the margin ρ into account, decreases with $\sqrt{m}$, m being the number of training examples. Important parameters of the LVQ model which influence this bound are (besides the achieved training error and margin) the number and the size of the prototypes. Interestingly, the number of free parameters is not directly included in this bound, which is $\mathcal{O}(k \cdot n^2)$ where n denotes the input dimensionality of the classifier. The dimensionality of the data contributes only indirectly through the margin ρ. Thus, similar to support vector machines, large margin generalization bounds can be obtained for local GMLVQ which are independent of the input dimensionality.

Naturally, the question occurs how to choose the margin parameter ρ in this bound. Eqn. 1.28 holds for every $\rho > 0$, but it will give different results for different values, since a balance between a low empirical error and a low structural term stemming from the function class has to be obtained. In practice, ρ will be chosen a posteriori according to the margin which can be achieved on (large parts of) the training set. Eqn. 1.28 does not hold for this scenario, since, in this case, ρ is no longer independent of the data. However, it is possible to generalize the above inequality such that bounds for arbitrary (posterior) ρ can be derived thereof, whereby the strict dependence on ρ is substituted by a prior believe value in the flavor of PAC-Bayesian bounds. We refer to [1.26] for further details.

We would like to conclude with a short look at the concrete training algorithms for LVQ networks. The bound 1.28 holds for every LVQ network regardless of the underlying training algorithm. However, it can be expected that a training algorithm which does not only reduce the number of misclassifications but which also has a look at the margin of the classification would be beneficial. Note that the margin $|\min_{\boldsymbol{w}_i^-}\{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^-)\} - \min_{\boldsymbol{w}_i^+}\{d_{\Lambda_i}(\boldsymbol{x}, \boldsymbol{w}_i^+)\}|$ directly corresponds to the nominator of the summands optimized by GLVQ and extensions as well as by LVQ2.1. Thus, it can be expected that training algorithms based on this cost function display an inherent tendency towards a good generalization of the classifier. Interestingly, in applications, one can often observe a tendency of matrix adaptation towards simple solutions, i.e. few dimensions are emphasized by the adaptive metric and the remaining ones are effectively dropped, as we have already seen in one example as depicted in Fig. 1.2.

1.5 Conclusions

We have discussed prototype-based classification algorithms in terms of the popular LVQ family, and we have put these algorithms into one mathematical

framework by referring to underlying cost functions. This point of view allows to easily generalize LVQ schemes towards very efficient and more powerful classifiers which extend the original LVQ scheme by adaptive metrics. This extension, though conceptually simple, drastically increases the applicability of LVQ classifiers which would be otherwise restricted to comparably simple low dimensional and Euclidean settings. While increasing the capacity of LVQ classifiers, these extensions maintain or even improve the interpretability of prototype-based classification rules by introducing additional terms which can be directly inspected such as a relevance weighting of the data dimensions. This scheme has proven beneficial in a variety of application areas ranging from the supervision of technical systems up to the analysis of biomedical data, as referenced in this chapter.

Interestingly, besides the increased capacity of LVQ classifiers, their excellent generalization ability is maintained by these models. This observation can be substantiated by an explicit mathematical framework stemming from computational learning theory, which allows the derivation of explicit worst case generalization bounds of LVQ classifiers, which demonstrate that these models can be interpreted as large margin classifiers similar to support vector machines. These mathematical foundations as well as the dramatically improved capability of LVQ networks makes them state-of-the-art classifiers with the interesting benefit of a direct interpretability of the models and linear runtime depending on the number of training examples.

GLVQ without matrix adaptation constitutes a $\mathcal{O}(n)$ algorithm, n referring to the data dimensionality. This complexity is not increased if relevance matrices are adapted during training, resulting in a highly efficient and flexible model. When dealing with full matrices, however, the complexity increases to $\mathcal{O}(n^2)$ which becomes infeasible for large input dimensionality. Therefore, it might be advisable to priorly reduce the ranks of the included matrices. This scheme has been proposed e.g. in [1.10]. In addition, further regularization might be interesting such as a controlled transition of the matrix from the standard Euclidean form to an adaptive matrix with possibly reduced rank. One very interesting possibility to achieve this goal using a simple regularization term has recently been introduced in [1.27].

References

1.1 Alegre, ,E., Biehl, M., Petkov, N., and Sanchez, L. (2008). Automatic classification of the acrosome status of boar spermatozoa using digital image processing and LVQ. Computers in Biology and Medicine **38**: 461–468.

1.2 Bartlett, P.L. and Mendelson, S. (2002). Rademacher and Gaussian complexities: risk bounds and structural risks. Journal of Machine Learning Research **3**:463-481.

1.3 Biehl, M., Breitling, R., and Li, Y. (2007). Analysis of Tiling Microarray Data by Learning Vector Quantization and Relevance Learning In: Proc. 8th Intl. Conf. on Intelligent Data Engineering and Automated Learning, IDEAL 2007, Springer LNCS, pp.880–889.

1.4 Biehl, M., Caticha, N., and Riegler, P. (2009): Statistical mechanics of online learning. In: Biehl, M., Hammer, B., Verleysen, M., and Villmann, T. Similarity-based clustering, biomedical applications, and beyond. Springer, to appear.
1.5 Biehl, M., Gosh, A., and Hammer, B. (2007): Dynamics and generalization ability of LVQ algorithms. Journal of Machine Learning Research **8**:323–360.
1.6 Biehl, M., Hammer, B., and Schneider, P. (2006): Matrix Learning in Learning Vector Quantization, Technical Report Clausthal University of Technology, Department of Computer Science, IfI-06-14.
1.7 Bojer, T., Hammer, B., and Koers, C. (2003). Monitoring technical systems with prototype based clustering. In: M.Verleysen (ed.), European Symposium on Artificial Neural Networks'2003, D-side publications, 433–439.
1.8 Bojer, T., Hammer, B., Schunk, D., and Tluk von Toschanowitz, K. (2001): Relevance determination in learning vector quantization. In Proc. Of European Symposium on Artificial Neural Networks (ESANN'01), pages 271–276. D facto publications.
1.9 Bunte, K., Petkov, N., Bosman, H.H.W.J., Biehl, M., and Jonkman, M. (2009), Efficient color features for content based image retrieval in dermatolgoy, submitted.
1.10 Bunte, K., Schneider, P., Hammer, B., Schleif, F.-M., Villmann, T., and Biehl, M. (2008). Discriminative visualization by limited rank matrix learning. Machine Learning Reports MLR-03-2008, ISSN:1865-3960 http://www.uni-leipzig.de/~compint/mlr/mlr_03_2008.pdf.
1.11 Crammer, K., Gilad-Bachrach, R., Navot, A. and Tishby, A. (2002). Margin analysis of the LVQ algorithm. In: Advances of Neural Information Processing Systems.
1.12 Denecke, A., Wersing, H., Steil, J.J., and Körner, E. (2009). Robust object segmentation by adaptive metrics in Generalized LVQ, submitted.
1.13 Duda, R.O., Hart, P.E., and Storck, D.G. (2001), Pattern Classification, Wiley.
1.14 Grandvalet, Y. (2000): Anisotropic noise injection for input variable relevance determination. IEEE Transactions on Neural Networks **11**(6):1201–1212.
1.15 Guyon, I. and Elisseeff, A. (2003): An Introduction to Variable and Feature Selection **3**:1157–1182.
1.16 Hammer, B., Strickert, M., and Villmann, T. (2005): On the generalization ability of GRLVQ networks. Neural Processing Letters **21**(2):109–120.
1.17 Hammer, B., Strickert, M., and Villmann, T. (2005): Supervised neural gas with general similarity measure. Neural Processing Letters **21**(1):21–44.
1.18 Hammer, B. and Villmann, T. (2002): Generalized relevance learning vector quantization. Neural Networks **15**: 1059–1068.
1.19 Kohonen, T. (2001): Self-Organizing Maps. Springer.
1.20 Lee, J.A. and Verleysen, M. (2007): Nonlinear dimensionality reduction, Springer.
1.21 Pregenzer, M., Pfurtscheller, G., and Flotzinger, D. (1996): Automated feature selection with distinction sensitive learning vector quantization. Neurocomputing **11**:19– 29.
1.22 Sato, A.S. and Yamada, K. (1998): An analysis of convergence in generalized LVQ. In Niklasson, L., Boden, M., and Ziemke, T. (eds.) ICANN'98, pp.172–176, Springer.
1.23 Sato, A.S. and Yamada, K. (1995): Generalized learning vector quantization. In Tesauro, G., and Touretzky, D., and Leen, T., Advances in Neural Information Processing Systems, 7, pp.423–429, MIT Press.

1.24 Schleif, F.-M., Hammer, B., Kostrzewa, M., and Villmann, T. (2007). Exploration of Mass-Spectrometric Data in Clinical Proteomics Using Learning Vector Quantization Methods. Briefings in Bioinformatics **9**(2): 129–143.
1.25 Schneider, P., Biehl, M., and Hammer, B. (2008). Matrix adaptation in discriminative vector quantization. Technical Report Clausthal University of Technology, Department of Computer Science, IfI-08-08.
1.26 Schneider, P., Biehl, M., and Hammer, B. (2009). Adaptive relevance matrices in learning vector quantization. Submitted.
1.27 Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T., and Biehl, M. (2008). Regularization in matrix relevance learning. Machine Learning Reports MLR-02-2008, ISSN:1865-3960 http://www.uni-leipzig.de/~compint/mlr/mlr_02_2008.pdf.
1.28 Seo, S., Bode, M., and Obermayer, K. (2003): Soft nearest prototype classification. IEEE Transations on Neural Networks, **14**(2): 390–398.
1.29 Seo, S. and Obermayer, K. (2003): Soft learning vector quantization. Neural Computation **15**(7): 1589–1604.
1.30 Sontag, E.D. (1992). Feedforward nets for interpolation and classification.Jornal of Computer and System Sciences **45**.
1.31 Strickert, M., Seiffert, U., Sreenivasulu, N., Weschke, W., Villmann, T., and Hammer, B. (2006). Generalized Relevance LVQ (GRLVQ) with Correlation Measures for Gene Expression Data. Neurocomputing, **69**: 651–659.
1.32 Tibshirani, R. (1996): Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B, **58**:267–288, 1996.
1.33 Valiant, L. (1984). A Theory of the Learnable. Communications of the ACM, **27**(11):1134–1142.
1.34 Vapnik, V. (1998). Statistical Learning Theory. Wiley, New York.
1.35 Villmann, T., Merenyi, E., and Hammer, B. (2003). Neural maps in remote sensing image analysis, Neural Networks **16**(3-4),389–403.
1.36 Villmann, T., Schleif, F.-M., and Hammer, B. (2006): Comparison of Relevance Learning Vector Quantization with other Metric Adaptive Classification Methods, Neural Networks **19**:610–622.
1.37 Weinberger, K., Blitzer, J., and Saul, L. (2006): Distance metric learning for large margin nearest neighbor classification. In Weiss, Y., Scholkopf, B., and Platt, J., editors, Advances in Neural Information Processing Systems 18, pp. 1473-1480. MIT Press.