Data Mining and Machine Learning

Astroinformatics
Proceedings IAU Symposium No. 325, 2016
M. Brescia, S.G. Djorgovski, E. Feigelson,
G. Longo & S. Cavuoti, eds.

doi:10.1017/S1743921316012928

Prototype-based Models for the Supervised Learning of Classification Schemes

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

[1]Johann Bernoulli Institute for Mathematics and Computer Science
University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands
email: `m.biehl@rug.nl`
[2]CITEC Center of Excellence, Bielefeld University, Univ.-Str. 21-23, 33594 Bielefeld, Germany
email: `bhammer@techfak.uni-bielefeld.de`
[3]Computational Intelligence Group, Univ. of Applied Sciences, Technikumplatz 17, 09648 Mittweida, Germany
email: `villmann@hs-mittweida.de`

Abstract. An introduction is given to the use of prototype-based models in supervised machine learning. The main concept of the framework is to represent previously observed data in terms of so-called prototypes, which reflect typical properties of the data. Together with a suitable, discriminative distance or dissimilarity measure, prototypes can be used for the classification of complex, possibly high-dimensional data. We illustrate the framework in terms of the popular Learning Vector Quantization (LVQ). Most frequently, standard Euclidean distance is employed as a distance measure. We discuss how LVQ can be equipped with more general dissimilarites. Moreover, we introduce relevance learning as a tool for the data-driven optimization of parameterized distances.

Keywords. miscellaneous, methods: data analysis, techniques: miscellaneous

1. Introduction

Prototype-based models constitute a very successful family of methodological approaches in machine learning (see e.g. Kohonen 1990, Hastie *et al.* 2009, Biehl *et al.* 2016, Biehl *et al.* 2009). They are appealing for a number of reasons: The extraction of information from previously observed data in terms of typical representatives, so-called prototypes, is particularly transparent and intuitive, in contrast to many, more *black-box* like systems. The same is true for the working phase, in which novel data are compared with the prototypes by use of a suitable (dis-)similarity or distance measure.

Prototype systems are frequently employed for the *unsupervised* analysis of complex data sets, aiming at the detection of underlying structures, such as clusters or hierarchical relations, see for instance (Biehl *et al.* 2009). Competitive Vector Quantization or the well-known K-means algorithm are prominent examples for the use of prototypes in the context of unsupervised learning (Duda *et al.* 2001, Hastie *et al.* 2009).

Potential goals of *supervised* machine learning are the assignment of data to categories in *classification* problems, or their characterization by a continuous target value in *regression* tasks. In both cases, the learning or training process relies on the availability of labeled example data. The aim is to extract relevant information and represent it in terms of a hypothesis for the unknown target function. The obtained hypothesis can then be applied to novel data in a working phase.

In the following we focus on the framework of Learning Vector Quantization for classification. Besides basic concepts and training prescriptions we present extensions of the

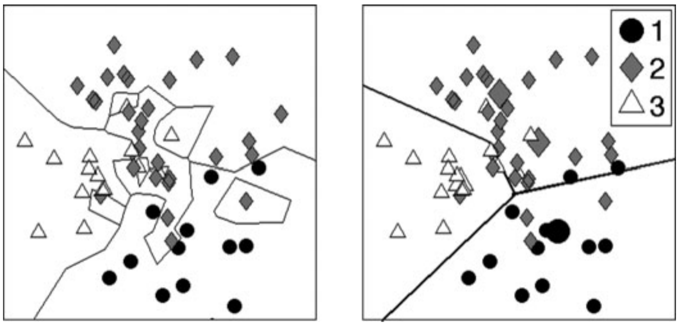

Figure 1. Illustration of the Nearest Neighbor Classifier (left panel) and an NPC scheme (right panel) for a data set comprising 3 classes. Both are based on Euclidean distance and yield piece-wise linear decision boundaries. Prototypes are represented by larger symbols (right).

framework to unconventional distances and, moreover, to the use of adaptive measures in so-called relevance learning schemes.

2. Prototypes in Supervised Learning

Among the many frameworks developed for supervised machine learning, prototype-based systems are particularly intuitive, flexible, and easy to implement. Although we restrict the discussion to classification problems, many of the concepts carry over to regression or, to a certain extent, also to unsupervised learning (Biehl *et al.* 2009).

Various prototype-based classifiers have been considered in the literature, some of them are derived from well-known unsupervised schemes like the Self-Organizing-Map or the Neural Gas (Kohonen 1990, Martinetz *et al.* 1993) which can be equipped with a posterior labeling of prototypes. Here, our focus is on the so-called Learning Vector Quantization (LVQ), a framework which was originally suggested by Kohonen (1990). As a starting point for the discussion, we briefly revisit the well-known k-Nearest-Neighbor (kNN) approach to classification (e.g. Duda *et al.* 2001, Cover 1967).

2.1. *Nearest Prototype vs. Nearest Neighbor Classifiers*

Nearest Neighbor classifiers (Duda *et al.* 2001, Cover 1967) constitute one of the simplest and most popular classification schemes. In this classical approach, a given set of labeled feature vectors is stored as a reference set:

$$I\!D = \{\mathbf{x}^\mu, y^\mu = y(\mathbf{x}^\mu)\}_{\mu=1}^{P} .$$

Here, the labels $y^\mu \in \{1, 2, \dots C\}$ indicate each example's membership to one of C classes.

An arbitrary observation or feature vector $\mathbf{x}$, can be classified according to its (dis-)similarities to the reference data. Most frequently, its (squared) Euclidean distance from all $\mathbf{x}^\mu \in I\!D$ is computed: $d(\mathbf{x}, \mathbf{x}^\mu) = (\mathbf{x} - \mathbf{x}^\mu)^2$, and $\mathbf{x}$ is then assigned to the class of its Nearest Neighbor exemplar in $I\!D$. Note that the square is irrelevant when comparing distances for the identification of the nearest neighbors. In the more general kNN classifier, the assignment is determined by means of a voting scheme that considers the k closest reference vectors (Cover 1967).

The NN or kNN classifier is obviously very easy to implement as it does not even require a *training phase*. Nevertheless one can show that the kNN approach bears the potential to yield Bayes optimal performance if the number k of neighbors is chosen carefully (Duda *et al.* 2001, Cover 1967). Consequently, the kNN method serves, to date, as an important baseline algorithm and is frequently used as a benchmark to compare with.

Figure 1 (left panel) depicts the NN classifier schematically and shows how the system implements piecewise linear decision boundaries. Several key difficulty become apparent already in the simple illustration: Class boundaries can become overly complex, e.g. in the presence of single data points which are potentially mislabeled. The fact that every exemplar contributes with equal weight can lead to undesired over-fitting effects where the classifier is overly specific to the stored examples, but does not generalize well with respect to novel data.

In naive implementations of kNN one would have to compute and sort the distances of $\mathbf{x}$ from all available examples. While efficient sorting strategies can reduce the computational complexity, the problem persists for very large data sets, in principle.

It should be possible to mitigate both difficulties by reducing the number of exemplars in an intelligent fashion, keeping essential properties of the data set. In fact, the selection of a suitable subset of reference vectors was already suggested by Hart (1968).

2.2. *Learning Vector Quantization*

This particularly intuitive approach to classification was introduced by Kohonen (1990). The basic idea is to represent the example data in terms of a (typically small) set of prototype vectors which capture the characteristics of the classes.

As pointed out in (Kohonen 1990), LVQ can be motivated as an approximation of a Bayes classifier assuming that the underlying density of data is a mixture of Gaussians (Duda *et al.* 2001). In Kohonen's LVQ, the actual density estimation is replaced by the much simpler and more robust method of supervised Vector Quantization: Each of the C classes is to be represented by at least one of the M labeled prototypes:

$$\left\{ \mathbf{w}^j , c^j \right\}_{j=1}^{M} \text{ where } \mathbf{w}^j \in I\!R^N \text{ and } c^j \in \{1, 2, \ldots C\}. \tag{2.1}$$

In analogy to the NN scheme, a Nearest Prototype classifier (NPC) assigns a feature vector $\mathbf{x}$ to the class

$$y(\mathbf{x}) = c^* \text{ where } c^* \text{ is the label of } \mathbf{w}^*(\mathbf{x}) = \text{argmin}_{\mathbf{w}^j} \left\{ \mathrm{d}(\mathbf{x}, \mathbf{w}^j) \right\}_{j=1}^{M} . \tag{2.2}$$

The term *winner* is used for the closest prototype $\mathbf{w}^*(\mathbf{x})$. Alternative voting rules, probabilistic or *soft* assignments can be used instead of Eq. (2.2), but we restrict the discussion to the simple and intuitive NPC scheme in the following. The right panel of Fig. 1 illustrates the concept: Class borders resulting from a few, carefully placed prototypes are much smoother than the corresponding NN decision boundaries (left). Obviously, an NPC classifier will be more robust with respect to details of the data set, outliers or mis-classified training samples. Consequently, one would expect better generalization behavior, as an LVQ classifier should be less data set specific.

The performance of LVQ systems has proven competitive in a variety of practical contexts. Moreover, the flexibility and intuitive accessibility of LVQ constitute the most striking advantages of this prototype-based approach to supervised learning. Prototypes are obtained and can be interpreted within the space of observed data, which makes possible direct discussion with domain experts. This is in contrast to many other, less transparent machine learning frameworks.

A variety of schemes have been suggested for the *training*, i.e. the computation of prototypes from a given set of data. The term LVQ1 refers to the original prescription suggested by Kohonen (1990). It includes essentially all aspects of the many modifications that were suggested later (e.g. Kohonen 1990, Nova & Estévez 2014). LVQ1 training can be summarized in terms of the following steps:

(1) Random sequential presentation of data: A single feature vector $\mathbf{x}^\mu$ with class label y^μ is randomly selected from the given data set with uniform probability.

(2) Identification of the *winning prototype*: The currently closest prototype, the so-called winner $\mathbf{w}^*(\mathbf{x}^\mu)$, abbreviated as $\mathbf{w}^*$ in the following, is determined according to (squared) Euclidean distance; it carries the class label $y^* = y(\mathbf{w}^*(\mathbf{x}^\mu))$.

(3) *Winner-Takes-All* (WTA) update:

$$\mathbf{w}^* \leftarrow \mathbf{w}^* + \eta\, \Psi(y^*, y^\mu)\ (\mathbf{x}^\mu - \mathbf{w}^*) \quad \text{where} \quad \Psi(y, \hat{y}) = \begin{cases} +1 & \text{if } y = \hat{y} \\ -1 & \text{else.} \end{cases} \tag{2.3}$$

In step (3), the winning prototype is moved closer to the presented feature vector if $\mathbf{w}^*$ and the example carry the same class label (as indicated by $\Psi = 1$). If the prototype represents a class different from y^μ, it is moved away from $\mathbf{x}^\mu$ ($\Psi = -1$). The magnitude of the update is controlled by the so-called *learning rate* η. One possible initialization strategy is to place prototypes close to the class-conditional mean vectors in the data set. After repeated presentations of the entire training set, the prototypes should represent their respective class by assuming *class–typical* positions in feature space.

Numerous modifications of the basic LVQ scheme have been considered in the literature, see for instance (Kohonen 1990, Nova & Estévez 2014) and references therein, mostly aiming at faster convergence or better generalization behavior. Most importantly, approaches that are based on suitable cost-functions have been suggested, which allow for training in terms of gradient-based or other well-known optimization schemes. Note that LVQ1 and other heuristic schemes cannot be interpreted as descent algorithms in a straightforward fashion.

A popular cost function algorithm is Robust Soft LVQ (RSLVQ), which can be motivated in the context of statistical models of the observed data (Seo *et al.* 2003). The so-called Generalized LVQ (GLVQ) (Sato & Yamada 1996) is guided by a cost function which can be related to the concept of *large margin* classifiers:

$$E = \sum_{\mu=1}^{P} \Phi(e^\mu) \quad \text{with} \quad e^\mu = \frac{d(\mathbf{w}_\mu^J, \mathbf{x}^\mu) - d(\mathbf{w}_\mu^K, \mathbf{x}^\mu)}{d(\mathbf{w}_\mu^J, \mathbf{x}^\mu) + d(\mathbf{w}_\mu^K, \mathbf{x}^\mu)}. \tag{2.4}$$

It is defined as a sum over all example data. Given a particular $\mathbf{x}^\mu$, the vector $\mathbf{w}_\mu^J$ denotes the closest of all prototypes which carry the same label as the example. Likewise, $\mathbf{w}^K$ denotes the closest prototype with a label different from y^μ. Popular choices for the increasing function $\Phi(e)$ in Eq. (2.4) are the identity $\Phi(e) = e$ or the sigmoidal $\Phi(e) = 1/[1 + exp(-\gamma\, e)]$, where $\gamma > 0$ controls its *steepness* (Sato & Yamada 1996). The arguments of Φ obeys $-1 \leqslant e^\mu \leqslant 1$ and negative values $e^\mu < 0$ indicate that the corresponding training example is correctly classified in the NPC scheme. Note that for large γ the costs approximate the number of misclassified training data, while for small γ the minimization of E corresponds to maximizing the margin-like quantities e^μ.

A popular and conceptually simple strategy to optimize E is *stochastic gradient descent* in which single examples are presented in randomized order (Robbins & Monro 1951, Bottou 1998). In contrast to LVQ1, two prototypes are updated in each step:

$$\mathbf{w}^J \leftarrow \mathbf{w}^J - \eta\, \frac{\partial \Phi(e)}{\partial \mathbf{w}^J} \quad \text{and} \quad \mathbf{w}^K \leftarrow \mathbf{w}^K - \eta\, \frac{\partial \Phi(e^\mu)}{\partial \mathbf{w}^K}. \tag{2.5}$$

Note that if Euclidean distance is used the gradient $\partial d(\mathbf{w}, \mathbf{x}) / \partial \mathbf{w} \propto (\mathbf{w} - \mathbf{x})$ yields updates which move the correct (incorrect) prototype towards (away) from the example, respectively. Hence, the basic concept of the intuitive LVQ1 is maintained in GLVQ.

In practice, very often a decreasing learning rate η is used to ensure convergence of the prototype positions (Robbins & Monro 1951). Alternatively, schemes for automated learning rate adaptation or more sophisticated optimization methods can be applied (e.g. Sra *et al.* 2011).

3. Unconventional distance measures and Relevance Learning

So far, the discussion focussed on Euclidean distance as a standard measure for the comparison of data points and prototypes. This choice appears natural and it is arguably the most popular one. One has to be aware, however, that other choices may be more suitable for real world data. Depending on the application at hand, unconventional measures might outperform Euclidean distance by far. Hence, the selection of a specific distance constitutes a key step in the design of prototype-based models. In turn, the possibility to choose a distance based on prior information and insight into the problem, contributes to the flexibility of the approach.

3.1. *Unconventional distances*

A large variety of dissimilarity measures could be employed in order to compare given N-dimensional vectors. We mention just a few prominent alternatives to the standard Euclidean metrics. In, for instance, (Hammer & Villmann 2005, Biehl *et al.* 2014, Biehl *et al.* 2016), more detailed discussions and further references can be found.

Statistical properties of a given data set can be taken into account explicitly by employing the so-called *Mahalanobis distance* (Mahalonobis 1936). This classical measure is a popular tool in the analysis of data sets. Duda *et al.* (2011) present a detailed discussion and application examples. The Mahalanobis distance can be used for comparing two vectors $\mathbf{x}, \mathbf{y}$ which are assumed to be generated according to the same distribution:

$$d_M(\mathbf{x}, \mathbf{y}) \;=\; \left[(\mathbf{x}-\mathbf{y})^\top C^{-1} (\mathbf{x}-\mathbf{y})\right]^{1/2}, \tag{3.1}$$

where C is the covariance matrix of the data. A diagonal approximation of C would rescale all features according to their empirical standard deviation, while replacing C by the identity matrix obviously recovers Euclidean metrics.

The family of *Minkowski distances* satisfies metric properties for values of $p \geqslant 1$ in

$$d_p(\mathbf{x}, \mathbf{y}) = \left[\textstyle\sum_{j=1}^N |x_j - y_j|^p\right]^{1/p} \quad \text{for} \;\; \mathbf{x}, \mathbf{y} \in I\!R^N. \tag{3.2}$$

Obviously, it includes the Euclidean measure as the special case with $p = 2$. Larger (smaller) values of p put emphasis on the components x_j and y_j with larger (smaller) deviations $|x_j - y_j|$. Setting $p \neq 2$ has been shown to improve performance in several applications, see (Biehl *et al.* 2007, Golubitsky & Watt 2010) for specific examples.

Note that the squared Euclidean distance can be rewritten as $d(\mathbf{x}, \mathbf{w})^2 = (\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{w} + \mathbf{w}\cdot\mathbf{w})$. So-called *kernelized distances* (Schölkopf 2001) replace the inner products $\mathbf{x}\cdot\mathbf{w}$ by a function $\kappa(\mathbf{x}, \mathbf{w})$:

$$d_\kappa(\mathbf{x}, \mathbf{w})^2 = \left[\kappa(\mathbf{x}, \mathbf{x}) - 2\kappa(\mathbf{x}, \mathbf{w}) + \kappa(\mathbf{w}, \mathbf{w})\right]. \tag{3.3}$$

A proper kernel function κ can be interpreted as an inner product of transformed feature vectors (Schölkopf 2001). Frequently, the kernel is associated with a non-linear transformation from $I\!R^N$ to a potentially higher-dimensional feature space.

The so-called *kernel trick* has become popular in the context of the Support Vector Machine (SVM) (Taylor & Cristianini 2004, Schölkopf & Smola 2002). It exploits the fact that the transformation itself does not have to be known explicitly. Instead it is possible to work with the kernel function only, which represents the non-linear mapping implicitly. For mathematical conditions on κ which justify this approach, see e.g. (Taylor & Cristianini 2004, Schölkopf & Smola 2002).

In SVM training one takes advantage of the fact that data which is difficult to separate in the original feature space can become linearly separable in the high-dimensional transformed space. Similarly, kernel distances can be used in the context of LVQ to translate complex classification tasks into simpler problems in higher dimensions. Villmann *et al.* (2012) provide an example in the context of face recognition.

As a last example, *statistical divergences* can be interpreted as measures of dissimilarity between densities or histograms representing the data. For example, image data is frequently characterized by color or other histograms. Similarly, text can be represented by frequency counts in a *bag of words* approach.

In the corresponding classification problems, the task could be to discriminate, for instance, uni-modal from multi-modal histograms or to detect the presence of a characteristic skewness. The Euclidean metric is often insensitive to these or similar properties. Hence, the efficient comparison of histogram data can benefit from using specific divergence-based measures, see (Cichocki *et al.* 2009) for a systematic description.

For an example application of divergences in the context of classification see (Mwebaze *et al.* 2011). There, it is also demonstrated that even non-symmetric divergences can be employed properly in the context of LVQ, as long as the measure is used consistently.

3.2. *LVQ training based on generalized distances*

As discussed above, training prescriptions based on Euclidean metrics generically yield prototype displacements along the vector $(\mathbf{x}^\mu - \mathbf{w})$ which can be related to the gradient of the distance measure. Replacing the Euclidean distance by a general, differentiable measure $\delta(\mathbf{x}^\mu, \mathbf{w})$, allows for the analogous derivation of LVQ training schemes. This is particularly straightforward in cost function based schemes like GLVQ, cf. Eq. (2.4), but it is also possible for the heuristic LVQ1, which we present here as an example. As a generalization of Eq. (2.3) we obtain the WTA update

$$\mathbf{w}^* \leftarrow \mathbf{w}^* \;-\; \eta\; \Psi(y^*, y^\mu)\; \tfrac{1}{2}\, \tfrac{\partial \delta(\mathbf{w}^*, \mathbf{x}^\mu)}{\partial \mathbf{w}^*}. \tag{3.4}$$

Obviously, the winner $\mathbf{w}^*$ has to be determined according to the measure δ, consistently.

Along these lines, LVQ1- or GLVQ-type update rules can be derived for quite general dissimilarities, provided δ is differentiable with respect to the prototype positions. Note that the formalism does not require metric properties like symmetry or the triangular inequality. As a minimal condition, non-negativity $\delta(\mathbf{w}, \mathbf{x}) \geqslant 0$ should be satisfied for $\mathbf{w} \neq \mathbf{x}$ and $\delta(\mathbf{x}, \mathbf{x}) = 0$.

The framework can be extended to non-differentiable measures, if differentiable approximations are available as in the case of the *Manhattan distance*, $p = 1$ in Eq. (3.2), see (Lange & Villmann 2013) for an example. Cost function based approaches could also employ non-differentiable measures if one resorts to alternative optimization strategies which do not require the computation of gradients (Sra *et al.* 2011).

3.3. *Adaptive Distances in Relevance Learning*

In the previous subsections, a few alternative distance measures have been discussed. In a given practical problem, a specific measure could be selected based on prior insights or according to an empirical comparison of performances. The framework of Relevance Learning allows for a significant conceptual extension of dissimilarity-based classification. It was introduced and extended in (e.g. Hammer & Villmann 2002, Schneider *et al.* 2009, Schneider *et al.* 2010, Bunte *et al.* 2012) and has proven useful in a variety of applications, including biomedical problems and image processing tasks, see for instance (Arlt *et al.* 2011, Denecke *et al.* 2009).

In this very elegant approach, only the parametric form of the distance measure is fixed in advance. Its parameters are considered adaptive quantities which can be adjusted or optimized in the data-driven training phase. The basic idea is very versatile and can be employed in a variety of learning tasks. We present here only one particularly clear-cut and successful example in the context of supervised learning: the so-called Matrix Relevance LVQ for classification (Schneider *et al.* 2009).

As suggested also in several other schemes (e.g. Weinberger & Saul 2009, Backhaus *et al.* (2012), Boareto *et al.* 2015), Matrix Relevance LVQ employs a generalized quadratic distance of the form

$$\delta_\Lambda(\mathbf{x},\mathbf{y}) \;=\; (\mathbf{x}-\mathbf{y})^\top \;\Lambda\; (\mathbf{x}-\mathbf{y})\,. \tag{3.5}$$

Here, the elements of the so-called relevance matrix $\Lambda \in I\!\!R^{N\times N}$ are considered degrees of freedom which are optimized in the training process. Note that, while the measure formally resembles the Mahalanobis distance, Eq. (3.1), it cannot be computed directly from the data and its statistical properties.

Diagonal entries of Λ represent the importance of single feature dimensions in the distance and can also account for potentially different magnitudes of the features. The contributions of pairs of features are weighted by the off-diagonal elements, which enables the system to reflect the interplay of different dimensions in feature space.

In order to fulfill the minimal requirement of non-negativity, $\delta_\Lambda(\mathbf{x},\mathbf{y}) \geqslant 0$, a re-pameterization is introduced in terms of an auxiliary matrix $\Omega \in I\!\!R^{N\times N}$ with

$$\Lambda = \Omega^\top\Omega, \quad \text{i.e.} \quad \delta_\Lambda(\mathbf{x},\mathbf{y}) \;=\; [\Omega\;(\mathbf{x}-\mathbf{y})]^2\,. \tag{3.6}$$

Hence, δ_Λ can be interpreted as the conventional Euclidean distance after a linear transformation of feature space. Note that (3.5) defines only a *pseudo-metric* in $I\!\!R^N$: Λ can become singular implying that $\delta_\Lambda(\mathbf{x},\mathbf{y}) = 0$ for particular $\mathbf{x} \neq \mathbf{y}$ is possible.

Several extensions and modifications of the basic idea have been considered in the literature: The restriction to diagonal matrices Λ corresponds to the original formulation of Relevance LVQ in (Hammer & Villmann 2002), which assigns a single adaptive weight to each feature. Rectangular ($N \times M$, with $M < N$) matrices Ω can be used to parameterize a low-rank relevance matrix. The corresponding low-dimensional intrinsic representation of data facilitates, for instance, the class-discriminative visualization of complex data (Bunte *et al.* 2012). The flexibility of the LVQ classifier is enhanced significantly when local distances are used, i.e. when separate relevance matrices are employed per class or even per prototype (Schneider *et al.* 2009).

Here we restrict ourselves to the simplest case of a single quadratic matrix Ω with $\Lambda = \Omega^\top\Omega$ defining a global distance. Gradient based updates for the simultaneous adapation of prototypes and relevance matrices can be derived from cost functions, observing that

$$\tfrac{\partial\delta_\Lambda(\mathbf{w},\mathbf{x})}{\partial\mathbf{w}} = \Omega^\top\Omega\,(\mathbf{x}-\mathbf{w}^*) \quad \text{and} \quad \tfrac{\partial\delta_\Lambda(\mathbf{w},\mathbf{x})}{\partial\Omega} = \Omega\;(\mathbf{x}-\mathbf{w}^*)\,(\mathbf{x}-\mathbf{w}^*)^\top\,. \tag{3.7}$$

Schneider *et al.* (2009) present the full form of the update equations based on the GLVQ cost funtion, yielding the so-called Generalized Matrix Relevance LVQ (GMLVQ). The extension of the heuristic LVQ1 prescription by means of relevance matrices is briefly discussed in (Biehl *et al.* 2016) and its convergence behavior is analysed in (Biehl *et al.* 2015).

In both, GMLVQ and Matrix LVQ1, the relevance matrix is updated in order to decrease or increase $\delta_\Lambda(\mathbf{w}^*,\mathbf{x}^\mu)$, depending on the class labels of the winning prototypes and the example vector. The matrix Ω can be initialized as the N-dimensional identity

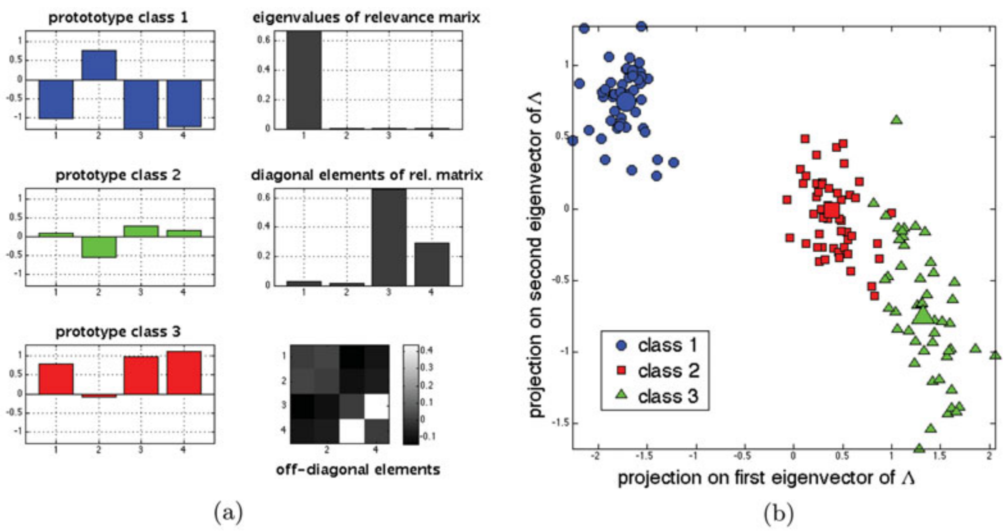

Figure 2. Visualization of the Generalized Matrix Relevance LVQ system as obtained from the z-score transformed Iris flower data set, see Sec. 3.3 for details. Panel (a), left column, displays the class prototypes as bar plots with respect to the four feature space components. The right column in panel (a) shows the eigenvalue spectrum of Λ in the uppermost bar plot, the diagonal elements of Λ, and the off-diagonal elements in a gray-scale representation. Panel (b) displays the projection of the entire data set onto the leading eigenvectors of the relevance matrix.

or in terms of independent random elements. In order to avoid numerical difficulties, a normalization of the form $\sum_i \Lambda_{ii} = \sum_{i,j} \Omega^2_{i,j} = 1$ is frequently imposed (Schneider *et al.* 2009).

In the following we illustrate Matrix Relevance LVQ in terms of a classical data set which is available from (Lichman 2013): We revisit the famous Iris data (Fisher 1936), in which four features are used to represent 150 samples from three different classes, i.e. species of Iris flowers. A simple LVQ system with one prototype per class and a single adaptive $\Omega \in \mathbb{R}^{4\times 4}$ was trained by use of the freely available GMLVQ *beginner's toolbox* (Biehl 2014). An additional z-score transformation was applied, resulting in re-scaled features with zero mean and unit variance in the data set. The resulting LVQ system achieves almost perfect, error-free classification of the training data and very good generalization behavior with respect to test set performance.

Figure 2 visualizes the obtained classifier. The left column of panel (a) displays the prototypes after training while the right column shows the resulting relevance matrix and its eigenvalues. As discussed above, the diagonal elements Λ_{ii} can be interpreted as the relevance of features i in the classification. Apparently, features 3 and 4 are the most important ones in the Iris classification problem. The off-diagonal elements (lower graph) represent the contribution of pairs of different features. Here, also the interplay of features 3 and 4 appears to be important.

In more challenging, realistic data sets, Relevance Matrix LVQ can provide valuable insights into the problem. The formalism has been used to identify most relevant or irrelevant features, e.g. in the context of medical diagnosis problems: The analysis of steroid excretion data for the classification of adrenal tumors facilitated the identification of a discriminative subset of bio-markers as documented in (Arlt *et al.* 2011, Biehl *et al.* 2012). In the analysis of cytokine expressions, GMLVQ revealed the dominant role of two particular markers in early stage Rheumatoid Arthritis, see (Yeo *et al.* 2015) for a detailed presentation from the medical perspective.

Nominally, the relevance matrix comprises on the order of $\mathcal{O}(N^2)$ adaptive parameters in an N-dim. feature space. Hence, one might expect unfavorable over-fitting effects. However, as observed empirically, relevance matrices display a strong tendency to become singlar with very low rank in the course of training, see (Schneider *et al.* 2009) for illustrative examples and (Biehl *et al.* 2015) for a theoretical analysis of this property. This phenomenon can be understood as an intrinsic regularization which limits the complexity of the distance measure and prevents over-fitting in many situations. In addition, the low rank of Λ facilitates the discriminative visualization of labelled data sets in terms of projections onto its leading eigenvectors. As an example, Fig. 2 displays the Iris flower data set.

4. Summary and Outlook

The aim of this paper is far from giving a complete review of the ongoing fundamental and application oriented research in this fascinating area of machine learning. The article provides, at best, first insights and can serve as a starting point for the interested reader.

The presentation is centered on Kohonen's Learning Vector Quantization. Examples for training prescriptions are given and the use of unconventional distance measures is discussed. As an important conceptual extension of LVQ, Relevance Learning is introduced, with Matrix Relevance LVQ serving as an example.

Prototype-based models continue to play a highly significant role in putting forward advanced machine learning techniques. We can only encourage the reader to explore recent developments in the literature. Challenging problems, such as the analysis of functional data, non-vectorial data or relational data, to name only very few, are currently being addressed. At the same time, exciting application areas are being explored in a large variety of domains.

References

Hastie, T., Tibshirani R. & Friedman, J. 2009, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Berlin: Springer

Kohonen, T. 1990, *Self-Organizing Maps.* Berlin: Springer

Martinetz T, Berkovich S & Schulten, K. 1993 *IEEE Trans. Neural Networks*, 4, 558–569.

Biehl, M., Hammer, B., Verleysen, M., & Villmann, T. (eds.) 2009, *Similiarity based clustering.* Berlin: Springer, Lecture Notes in Artificial Intelligence 5400

Biehl, M., Hammer, B., & Villmann, T. 2016, *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2), 92-111

Cover, T. & Hart, P. 1967, *IEEE Trans. Information Theory*, 13, 21-27

Duda, R., Hart, P., & Stork, D. 2001, *Pattern Classification.* Hoboken, NJ: Wiley

Hart P. 1968, *IEEE Trans. Information Theory*, 14, 515–516

Nova, D. & Estévez, P. A. 2014, *Neural Computing and Appl.*, 25, 11–524

Robbins, H. & Monro, S. 1951, *The Ann. of Math. Statistics*, 22, 405

Kohonen, T. 1990, In: *Proc. Int. Joint Conf. on Neural Networks*, Vol. 1, 545–550

Seo, S., Bode, M., & Obermayer, K. 2003, *IEEE Trans. Neural Networks*, 14, 390–398

Sato, A. & Yamada, K. 1996, In: D.S. Touretzky & M.E. Hasselmo (eds.), *Proc. Adv. in Neural Proc. Information Processing Systems 8.* Cambridge, MA: MIT Press, 423-429

Bottou, L. 1998, In: D. Saad (ed.), *Online Learning and Neural Networks.* New York: Cambridge Univ. Press, 9-42

Sra, S., Nowozin, S., & Wright, S. (eds.) 2011, *Optimization for Machine Learning.* Cambridge, MA: MIT Press

Hammer, B. & Villmann, T. 2005, In: M. Verleysen (ed.) *Proc. Europ. Symp. on Artificial Neural Networks.* Evere: d-side publishing, 303–316

Biehl, M., Hammer, B. &Villmann, T. 2014 In: L. Grandinetti, T. Lippert & N. Petkov (eds.), *Proc. BrainComp2013*. Berlin: Springer, Lecture Notes in Computer Science 8603, 110-116

Mahalonobis, P. 1936, In: *Proc. of the National Inst. of Sciences of India*, 2, 49-55

Schölkopf, B. 2001, In: *Proc. Adv. in neural information processing systems*, 301–307.

Biehl, M., Breitling, R., & Li, Y. 2007, In: H. Yin, P. Tino, E. Corchado, W. Byrne & X. Yao (eds.), *Proc. Intelligent Data Engineering and Automated Learning, IDEAL*. Berlin: Springer, Lecture Notes in Computer Science 4881, 880–889.

Golubitsky, O. & Watt, S. 2010, *Int. J. on Document Analysis and Recognition (IJDAR)*, 13, 133–146

Shawe-Taylor, J. & Cristianini, N. 2004, *Kernel Methods for Pattern Analysis*. New York: Cambridge University Press

Schölkopf, B. & Smola, A. 2002, *Learning with Kernels*. Cambridge, MA: MIT Press

Villmann, T., Kästner, M., Nebel, D., & Riedel, M. 2012, In: *Proc. of the International Conference on Machine Learning Applications (ICMLA)*. New York: IEEE, 7-10

Cichocki, A., Zdunek, R., Phan, A., & Amari, S. 2009, *Nonnegative Matrix and Tensor Factorizations*. Hoboken, NJ: Wiley

Mwebaze, E., Schneider, P., Schleif, F.-M., Aduwo, J., Quinn, J., Haase, S., Villmann, T., & Biehl, M. 2011, *Neurocomputing*, 74, 1429–1435

Lange, M. & Villmann, T. 2013, *Machine Learning Reports*, MLR-03-2013

Hammer, B. & Villmann, T. 2002, *Neural Networks*, 15(8), 1059-1068

Schneider, P., Biehl, M., & Hammer, B. 2009, *Neural Computation*, 21, 3532–3561

Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T., & Biehl, M. 2010, *IEEE Trans. Neural Networks*, 21, 831-840

Bunte, K., Schneider, P., Hammer, B., Schleif, F.-M., Villmann, T., & Biehl, M. 2012, *Neural Networks*, 26, 159-173

Biehl, M., Bunte, K., & Schneider, P. 2013, *PLOS ONE*, 8, e59401

Arlt, W., Biehl, M., Taylor, A., Hahner, S., Libe, R., Hughes, B., Schneider, P., Smith, D., Stiekema, H., Krone, N., Porfiri, E., Opocher, G., Bertherat, J., Mantero, F., Allolio, B., Terzolo, M., Nightingale, P., Shackleton, C., Bertagna, X., Fassnacht, M., & Stewart, P. 2011, *J. Clinical Endocrinology and Metabolism*, 44, 1892-1902

Biehl, M., Schneider, P., Smith, D., Stiekema, H., Taylor, A., Hughes, B., Shackleton, C., Stewart, P & Arlt, W. 2012 In: M. Verleysen (ed.) *Proc. Europ. Symp. on Artificial Neural Networks (ESANN)*. Evere: d-side publishing, 423–428

Leo, Y., Adlard, N., Biehl, M., Juarez, M., Samllie, T., Snow, M., Buckley, C. D., Raza, K., Filer, A., & Scheel-Toellner, D. 2016, *Ann. of the Rheumatic Disease*, 75, 763–771

Denecke, A., Wersing, H., Steil, J., & Körner, E. 2009, *Neurocomputing*, 72, 1470-1482

Weinberger, K. & Saul, L. 2009, *J. of Machine Learning Res.*, 10, 207-244

Backhaus, A., Ashok, P., Praveen, B., Dholakia, K., & Seiffert, U. 2012, In: M. Verleysen (ed.) *Proc. Europ. Symp. on Artificial Neural Networks*. Evere: d-side publishing, 411-416

Boareto, M., Cesar, J., Leite, V., & Caticha, N. 2015, *IEEE/ACM Trans. Computational Biology and Bioinformatics*, 12(3), 705-711

Biehl, M., Hammer, B., Schleif, F.-M., Schneider, P., & Villmann, T. 2015, *Proc. Int. Joint Conf. Neural Networks*. New York: IEEE, 8 pages

Biehl, M. 2014, website: `http://www.cs.rug.nl/~biehl/gmlvq`

Fisher, R. 1936, *Annual Eugenics*, 7, 179-188

Lichman, M. 2013, *UCI machine learning repository*, website: `http://archive.ics.uci.edu/ml`.