Prototype-based models in machine learning

Biehl, Michael; Hammer, Barbara; Villmann, Thomas

Published in:
Wiley Interdisciplinary Reviews. Cognitive Science

DOI:
10.1002/wcs.1378

Publication date:
2016

Advanced Review

Prototype-based models in machine learning

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

An overview is given of prototype-based models in machine learning. In this framework, observations, i.e., data, are stored in terms of typical representatives. Together with a suitable measure of similarity, the systems can be employed in the context of unsupervised and supervised analysis of potentially high-dimensional, complex datasets. We discuss basic schemes of competitive vector quantization as well as the so-called neural gas approach and Kohonen's topology-preserving self-organizing map. Supervised learning in prototype systems is exemplified in terms of learning vector quantization. Most frequently, the familiar Euclidean distance serves as a dissimilarity measure. We present extensions of the framework to nonstandard measures and give an introduction to the use of adaptive distances in relevance learning.

How to cite this article:
WIREs Cogn Sci 2016, 7:92–111. doi: 10.1002/wcs.1378

INTRODUCTION

Prototype-based models in machine learning make use of a number of appealing concepts. First of all, the explicit representation of observations, i.e., data, in terms of memorized exemplars or typical representatives is a very intuitive approach, which parallels similar concepts from cognitive psychology and the neurosciences. The same applies to the second major aspect, i.e., the comparison of observations or stimuli with a reference set of prototypes or exemplars in terms of *similarity*. The actual role and significance of prototype or exemplar models and the concept of similarity appear to be topics of ongoing research and debate in the context of cognitive sciences and related disciplines (see, e.g., Ref 1 for further references and a discussion from a machine learning perspective).

The purpose of this paper is neither to review these efforts nor to contribute to them. Instead, we aim at providing an overview of how the above-mentioned and other related concepts can be exploited in the context of analyzing complex datasets. We present and illustrate a few machine learning frameworks, which are formulated in terms of prototypes and employ the concept of similarity-based comparison. Perhaps the most popular of these approaches is the self-organizing map (SOM) as introduced by Kohonen.[2] It constitutes a widely applicable and fundamental self-organizing model that borrows several principles from biological neurons and yields a highly efficient machine learning tool. The aim of the SOM is to find a faithful topology-preserving representation of a given set of stimuli inspired by the topological maps that are apparently present in the cortex.

Kohonen's SOM comprises essentially all relevant model ingredients that the reader will encounter in this review. An SOM is characterized by a number of neurons that react to stimuli from the environment. Mathematically, this can be modeled by assigning a weight vector to each neuron representing the typically expected stimulus of the neuron. A given input is compared to these expectations, and the best matching neuron is identified. This winner-takes-all (WTA) principle can also be motivated in view of the

*Correspondence to: m.biehl@rug.nl

[1]Johann Bernoulli Institute for Mathematics and Computer Science, Faculty of Mathematics and Natural Sciences, University of Groningen, Groningen, The Netherlands

[2]CITEC Centre of Excellence, Faculty of Technology, Bielefeld University, Bielefeld, Germany

[3]Department of Mathematics, University of Applied Sciences Mittweida, Mittweida, Germany

Conflict of interest: The authors have declared no conflicts of interest for this article.

short-term dynamics of biological neurons: A neuron with high potential fires and thereby suppresses its neighbors by means of local inhibition.

The long-term adaptability of neurons and their dendritic connections serve as the motivation for an intuitive learning rule in artificial systems, which results in a topology-preserving map formation based on observed stimuli. For each presented stimulus, the winner is adapted toward the given stimulus, reminiscent of the basic principle of Hebbian learning.[3,4]

In the SOM, artificial neurons are arranged in a regular, most frequently two-dimensional, lattice structure. This introduces a topology that also allows the inclusion of neighborhood cooperativeness in the model: In addition to the winner, neurons in its immediate neighborhood are also adapted. This simple rule yields very powerful, topology-preserving, low-dimensional maps of possibly high-dimensional input stimuli. On one hand, data are compressed and noise is reduced by representing a continuum of stimuli by prototypes. On the other, a topological ordering and potential visualization of the global topology of high-dimensional stimuli is provided in terms of a nonlinear embedding in a regular lattice. Such topological maps can serve several purposes in practice: they allow for data visualization and compression, or they can implement a noise-tolerant association look-up table. Together with posterior labeling or other post-processing techniques, the SOM can also be employed in classification or regression tasks.[2]

Neighborhood cooperativeness can also be introduced without reference to a low-dimensional topology. The so-called neural gas (NG) approach[5] retains most of the concepts of the SOM; however, neighborhood relations are evaluated in terms of ranked dissimilarities in the original input space of stimuli. Thus, the winning neuron and those that are relatively close to it in terms of the dissimilarity measure are adapted to a given stimulus. The resulting framework provides a powerful tool for vector quantization (VQ) and has proven useful in a variety of practical situations.

When omitting the idea of neighborhood cooperation and retaining only the WTA principle, one recovers competitive VQ, arguably the simplest framework for representing data by prototypes. It is closely related to the popular k-means algorithm[4,6,7] and can be viewed as the basis of all prototype-based systems considered here.

SOM, NG, and other VQ frameworks are methods of unsupervised learning. They represent the underlying data distribution without taking into account potential feedback in a supervised learning scenario, where the target could be a particular clustering or classification, for instance.

In fact, the faithful representation of the stimuli may require more resources than the clustering itself, since more information has to be stored. For this reason, it is worthwhile designing methods of self-organized, prototype-based learning for supervised tasks. Motivated by the need for intuitive and fast supervised classification methods in practical applications, learning schemes such as learning vector quantization (LVQ) have been proposed.[2] Compared to SOM or VQ, the WTA principle is amended by explicitly taking into account the label information representing the target.

One might argue that the degree to which inspiration from cognitive or neurosciences plays a role decreases along the lines of the brief overview given above. At the same time, the models appear to become less plausible from a biological point of view. In a sense, purely application-oriented machine learning research can resort to a convenient, pragmatic attitude. While many models and techniques of machine learning have been, to a certain extent, inspired by concepts from other disciplines—once they have proven useful in a practical context—the actual relevance of their counterparts in biology or psychology becomes largely irrelevant from an application point of view. For instance, the great success of artificial feed-forward or recurrent neural networks[3,4,6] in a large variety of application areas is not impaired by the fact that some of the assumptions or hypotheses that inspired their development turned out to be questionable, over-simplifying, or even plain wrong in terms of biological neural systems. In this sense, inspirations from biology or analogies with cognitive psychology can be instructive and very useful. However, they should not be over-interpreted or over-rated when dealing with artificial systems employed in a practical context of machine learning or computational intelligence in a broader sense.

The following section presents and discusses several prototype-based systems of unsupervised learning. For didactical reasons, we follow a somewhat inverted order compared to the above introduction. After presenting the most basic competitive VQ, we introduce the NG approach and conclude the section with a discussion of the SOM in *Self-Organizing Maps*.

The next section introduces and illustrates the basic ideas of supervised prototype systems, mainly in terms of LVQ. To this end, we also compare it with the classical k-nearest neighbor classifier.

In all of the above, simple Euclidean distance in feature space is employed as the measure of dissimilarity. In *Generalized Dissimilarity Measures and Relevance Learning* we extend the framework significantly and discuss how alternative distances and dissimilarity measures can be incorporated into the models. In particular, we introduce the concept of relevance learning and adaptive distances in *Adaptive Distances in Relevance Learning* and conclude with a few remarks in the last section.

PROTOTYPES IN UNSUPERVISED LEARNING

One possible aim of unsupervised learning is the representation of vectorial data by typical representatives, i.e., prototypes, which somehow reflect the frequency of observations in feature space. The goal could be to reduce storage needs, to reveal structures such as clusters in the data, or to pre-process large datasets for further analysis. Unsupervised analysis can provide valuable insights into the nature of the dataset at hand, and it plays an important role in the context of visualization.

In the next subsection we present a very basic scheme for unsupervised VQ which employs the concept of competitive learning.[3,4] The concept is extended in the so-called NG approach by introducing a simple form of neighborhood cooperativeness into the framework of VQ.[8] Finally, *Self-Organizing Maps* presents the SOM, which adds the concept of a topological relation of prototypes.[2] Initially, we restrict the presentation to the use of Euclidean distance in an N-dimensional feature space. The generalization to alternative measures of dissimilarities will be discussed in Some Alternatives and Extensions.

Competitive Vector Quantization

The aim of VQ is usually formulated in terms of a cost function, which guides the training process, i.e., the computation of prototype vectors. It measures the quality of a particular prototype-based representation of a given set of P *feature vectors* $\{\mathbf{x}^\mu \in \mathbb{R}^N\}$, $\mu = 1, 2 \ldots P$, each of them representing a particular *stimulus* or *input*. A popular approach is based on the crisp assignment of any data point to the closest prototype, the so-called winner, in the set $W = \{\mathbf{w}^1, \mathbf{w}^2, \ldots \mathbf{w}^K\}$ in terms of a predefined distance measure. We first restrict the discussion to (squared) Euclidean distance in feature space with $d^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^2$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. A corresponding suitable cost function is given by the so-called quantization error[3,4]:

$$H_{\mathrm{VQ}} = \sum_{\mu=1}^{P} \frac{1}{2} d^2\left(\mathbf{w}^*(\mathbf{x}^\mu), \mathbf{x}^\mu\right). \tag{1}$$

Here, $\mathbf{w}^*(\mathbf{x}^\mu) \in \mathbb{R}^n$ denotes the prototype with the smallest Euclidean distance from a given feature vector $\mathbf{x}^\mu \in \mathbb{R}^N$:

$$d\left(\mathbf{w}^*(\mathbf{x}^\mu), \mathbf{x}^\mu\right) \le d\left(\mathbf{w}^j, \mathbf{x}^\mu\right) \quad \text{for all } j = 1, 2, K, \tag{2}$$

where ties are broken arbitrarily. In words, the quantization error sums up the distances of all individual feature vectors from their respective closest prototype. Hence, it quantifies how *faithfully* the data is represented by the set of prototypes.

Competitive VQ corresponds to a stochastic (or *online*) gradient descent[9] with respect to H_{VQ} and leads to a very intuitive update scheme. At each time step, a single feature vector $\mathbf{x}^\mu$ is selected randomly with equal probability from the dataset. Next, the current winner $\mathbf{w}^*(\mathbf{x}^\mu)$, abbreviated as $\mathbf{w}^*$, is determined according to Eq. (2). Only the winning prototype is updated according to

$$\mathbf{w}^* \leftarrow \mathbf{w}^* + \eta\left(\mathbf{x}^\mu - \mathbf{w}^*\right). \tag{3}$$

The term 'competitive learning' has been coined for this and other training schemes, where prototypes *compete* for updates.[3,8] The above-described algorithm is an example of a WTA scheme. Some extensions to the update of several prototypes at a time will be discussed below.

The so-called learning rate η controls the magnitude of updates. For $\eta < 1$, the prototype is moved even closer to the observed data point. Initially, prototypes could be set identical to randomly selected data points, for instance. All data are presented many times and, eventually, prototypes should be placed in regions of the feature space which display a high density of data points.

As in any stochastic gradient descent, convergence of the prototype vectors has to be guaranteed by employing a time-dependent learning rate, which slowly approaches zero in the course of training.[9] For a discussion of stochastic gradient descent in machine learning, its convergence properties, the role of the step size, and alternative optimization strategies, we refer the reader to, e.g., Ref 10 and references therein.

Competitive VQ is closely related to the well-known k-means algorithm,[4,6,7] which employs all

available data at a time in each update of the system. We refrain from a detailed discussion of this classical method and its relation to density estimation through expectation-maximization schemes.[4,6] Instead, we focus on competitive VQ, which can be interpreted as the conceptual basis of all other algorithms discussed in the following.

Frequently, VQ is confused or even identified with clustering, having in mind that prototypes might represent distinct clusters of data and H_{VQ} relates to the typical within-cluster distances. Indeed, VQ may serve as a basic strategy to detect and identify groups of similar data points. It is important to note, however, that VQ is well defined and can be useful even if there is no cluster structure present in the data. Moreover, the minimization of H_{VQ} does not necessarily coincide with the identification of existing or assumed clusters. Figure 1 shows a few illustrative situations where prototypes represent simple, two-dimensional data with minimal H_{VQ}. Panel (a) displays a single elongated cluster, where the prototypes summarize essential properties and characterize the variation of observed feature vectors without necessarily suggesting intuitively meaningful divisions of the data into groups. In panel (b), the idealized scenario of two nearly spherical overlapping clusters is shown, each of which is represented by one prototype. In panel (c), the clusters are elongated. While the resulting prototype positions minimize H_{VQ}, they would certainly not be interpreted as cluster centers. Obviously, the outcome of VQ depends strongly on the representation of data in feature space, and it can be highly sensitive to the scaling of single features or other, even linear, coordinate transformations. The last example in panel (d) shows a situation in which a seemingly separate smaller cluster is not identified because it does not contribute significantly to the total quantization error.

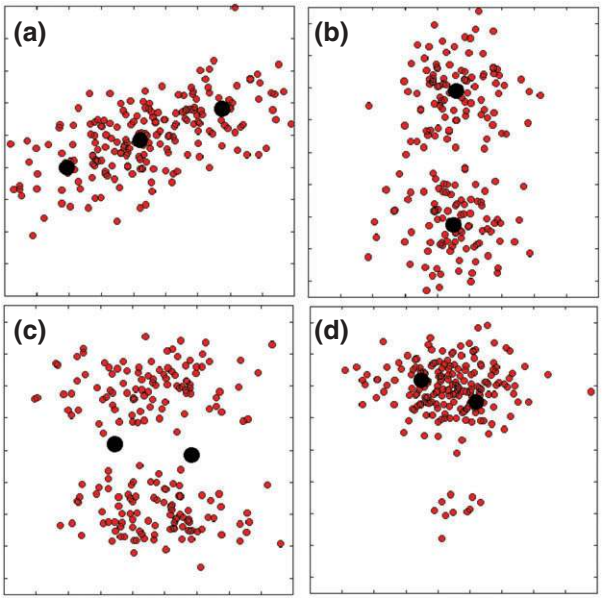

FIGURE 1 | Representation of two-dimensional data points by prototypes. In each panel, 200 data points are displayed as red dots, and prototype positions corresponding to the minimum of H_{VQ} are marked by filled black circles. The subpanels are referred to in *Competitive Vector Quantization*.

It is also important to note that H_{VQ} is a meaningful quality measure only when comparing systems with the same number of prototypes. In general, H_{VQ} will decrease with increasing K. Obviously, placing one prototype on each individual data point always gives the lowest possible quantization error $H_{VQ} = 0$.

A key difficulty of competitive VQ is that the objective function can display many suboptimal local minima. Among other problems, this can lead to a strong dependence of the training outcome on the initial prototype positions. As an extreme example, placing a prototype in an empty region of feature space can prevent it from ever being identified as the winner for any of the data points, thus leaving it unchanged in a WTA training process, leading to a so-called *dead unit*.

Neural Gas Algorithm

To overcome the sensitivity to initial conditions and other difficulties of WTA schemes, concepts of neighborhood cooperativeness were proposed as extensions of the VQ framework. Kohonen introduced a prominent, biologically motivated model, the so-called SOM,[2] which will be presented in the following section. Inspired by Kohonen's approach, Martinetz et al. developed a robust VQ scheme with a rank-based neighborhood cooperativeness, the so-called neural gas (NG).[5]

The main idea of NG is to update several or all prototypes at a time, but according to their rank with respect to the distance from a given sample. For a feature vector $\mathbf{x}^\mu$, the winner rank k_j for each prototype in the set $W = \{\mathbf{w}^j, j = 1, \ldots, K\}$ is determined as

$$k_j(\mathbf{x}^\mu, W) = \sum_{l=1}^{K} \theta\left(d\left(\mathbf{w}^j, \mathbf{x}^\mu\right) - d\left(\mathbf{w}^l, \mathbf{x}^\mu\right)\right), \qquad (4)$$

where $\theta(\cdot)$ denotes the Heaviside step function with $\theta(z) = 0$ if $z \leq 0$ and $\theta(z) = 1$ for $z > 0$. Hence, the winning prototype $\mathbf{w}^*(\mathbf{x}^\mu)$ has rank zero, rank 1 is assigned to the second closest prototype, and so forth.

In the NG algorithm, the prototype ranks $k_j(\mathbf{x}^\mu, W)$ as given by Eq. (4) are determined for a randomly selected single feature vector $\mathbf{x}^\mu$. Then, all prototypes are updated according to

$$\mathbf{w}^j \leftarrow \mathbf{w}^j + \eta \cdot h_\lambda^{\mathrm{NG}}\left(k_j(\mathbf{x}^\mu, W)\right) \cdot \left(\mathbf{x}^\mu - \mathbf{w}^j\right), \qquad (5)$$

where

$$h_\lambda^{\mathrm{NG}}\left(k_j(\mathbf{x}^\mu, W)\right) = \exp\left(-\frac{k_j(\mathbf{x}^\mu, W)}{\lambda}\right) \qquad (6)$$

is the so-called neighborhood function, which decreases with increasing rank. Prototypes that are relatively close to the actual feature vector are shifted considerably toward $\mathbf{x}^\mu$, whereas prototypes far away from the data point are updated only marginally. The parameter λ determines the range of ranks that correspond to significant updates of the prototypes. Note that λ is defined with respect to ranks and does not depend on the magnitude of actual distances in the dataset.

Frequently, λ is set to a large value, typically on the order of K, initially. This results in significant updates of all prototypes and resolves the potential problem of *dead units*. In the course of training, λ can be decreased gradually. Note that in the limit $\lambda \to 0$, the neighborhood function (6) singles out the winning prototype and a WTA prescription is recovered.

For fixed finite λ, the NG update corresponds to a stochastic gradient descent with respect to the cost function[5]:

$$H_\lambda^{\mathrm{NG}} = \sum_{\mu=1}^{P} \sum_{j=1}^{K} h_\lambda^{\mathrm{NG}}\left(k_j(\mathbf{x}^\mu, W)\right) \cdot d^2\left(\mathbf{w}^j, \mathbf{x}^\mu\right). \qquad (7)$$

Interpreting the prototypes as *gas particles* in the high-dimensional data space and the parameter λ as a *viscosity*, the averaged learning dynamic can be seen as a overdamped motion of particles in a potential given by the negative data density $p(\mathbf{x})$. Therefore, the name *Neural Gas Algorithm* has been coined for this particular VQ scheme.[5] It is frequently used for the faithful representation of datasets by a relatively large number of prototypes, the positions of which represent the density of observations in feature space. Application areas for NG and modifications thereof include, among others, the analysis of time series data and sequences, source separation, cluster analysis and visualization, image processing, and robotics.[5,11–16]

Self-Organizing Maps

Kohonen proposed a model of neighborhood cooperativeness between the prototypes, which is motivated by the formation of ordered representations of sensory information in cortical brain areas.[2,8,17,18] In the so-called SOM, an external *neural lattice* A is introduced, most frequently taken to be a regular two-dimensional grid. Nodes of the lattice are called neurons following the biological motivation, and each neuron is identified by its coordinate vector in the grid, e.g., by $\mathbf{r} = (r_1, r_2)^{\mathrm{T}}$ in two dimensions. Thus, a *topological* relation between the neurons is imposed. The by far most popular grid structures considered in the literature are square and hexagonal (equilateral triangular) lattices.[2] Neural weight vectors $\mathbf{w}_\mathbf{r} \in \mathbb{R}^N$ are assigned to each lattice site, forming the set W_A. Each weight vector corresponds to a prototype in the N-dimensional feature space, which is thus associated with a particular lattice site $\mathbf{r}$.

The update rule for prototypes in the SOM is similar to that of NG except for the neighborhood function, which is now based on the position of neurons in the lattice A: Let

$$\mathbf{s}(\mathbf{x}^\mu) = \mathrm{argmin}_{\mathbf{r}\in A}\, d(\mathbf{w}_\mathbf{r}, \mathbf{x}^\mu) \qquad (8)$$

be the actual winning neuron for the feature vector $\mathbf{x}^\mu$. Then all prototypes are updated according to

$$\mathbf{w}_\mathbf{r} \leftarrow \mathbf{w}_\mathbf{r} + \eta \cdot h_\rho^{\mathrm{SOM}}(\mathbf{s}(\mathbf{x}^\mu), \mathbf{r}) \cdot (\mathbf{x}^\mu - \mathbf{w}_\mathbf{r}), \qquad (9)$$

where the neighborhood function

$$h_\rho^{\mathrm{SOM}}(\mathbf{s}, \mathbf{r}) = \exp\left(-\frac{\| \mathbf{s} - \mathbf{r} \|_A^2}{2\rho^2}\right) \qquad (10)$$

is evaluated for two lattice sites in A using the Euclidean norm $\| \mathbf{s} - \mathbf{r} \|_A^2$. Here, the parameter ρ denotes the neighborhood range, controlling the degree of neural cooperativeness imposed by the lattice A. Thus, the topological structure of A directly influences the learning behavior. As for NG, the neighborhood range is usually decreased to zero during learning from a relatively high initial value of ρ. Note that ρ is defined in terms of the lattice spacing in A, while the parameter λ of NG refers to ranked distances in $\mathbb{R}^N$.

The above-discussed basic version of the SOM training process, Eq. (9), cannot be readily interpreted as a proper stochastic gradient descent. However, this is possible for a modification of the update, which retains the basic concepts of Kohonen's algorithm and can be derived from a suitable cost function.[19]

The key feature of the SOM is that it provides a low-dimensional, topology-preserving representation of the data, which can be employed for, e.g., the purpose of visualization. To this end, we can interpret the winner determination, Eq. (8), as a map that assigns any N-dimensional feature vector $\mathbf{x}$ to a specific lattice site $\mathbf{r}$ in A. Under certain conditions, this mapping is topology-preserving in the following sense: feature vectors $\mathbf{x}^{\mu}$ and $\mathbf{x}^{\nu}$ with small distance $d(\mathbf{x}^{\mu}, \mathbf{x}^{\nu})$ in $\mathbb{R}^N$ are mapped either to identical or neighboring neurons $\mathbf{s}$ and $\mathbf{s}'$ in the lattice A. Similarly, neurons $\mathbf{r}$ and $\mathbf{r}'$, which are neighbors in the lattice A, correspond to prototype vectors with small dissimilarity $d(\mathbf{w}_{\mathbf{r}}, \mathbf{w}_{\mathbf{r}'})$ in feature space. For a precise mathematical definition, see Ref 20. Measures for the evaluation of the topology preserving-property of a trained SOM are proposed in Refs 20,21.

The topology-preserving aspect of the SOM facilitates the systematic visual inspection of complex data sets. Several implementations of the SOM, including visualization tools, are available as software packages, e.g., Ref 17. An overview can be found in, e.g., Ref 22; for the following, we employed a freely available MATLAB SOM-Toolbox.[23]

For illustration purposes, we revisit the well-known *Iris* flower data, which was already considered by Fisher.[24] In the dataset as available from the UCI repository,[25] 150 individual flowers are represented by four numerical features (*sepal length*, *sepal width*, *petal length*, *petal width*) and belong to three different classes, i.e., species of Iris flowers (*setosa*, *virginica*, *versicolor*); see Ref 25 for a more detailed description of this classical dataset. An SOM with 4×6 prototypes and lattice sites was trained using the above-mentioned MATLAB toolbox.[23] Note that the unsupervised SOM training does not take into account the known class memberships of the, in total, 150 samples.

As a first example, we discuss the visualization of the trained SOM prototypes in terms of the so-called component planes. The jth component plane of a rectangular SOM lattice A of $(a \times b)$ neurons is an $(a \times b)$-dimensional matrix, where each entry corresponds to a particular lattice site (r_1, r_2) and is given by the jth vector component of the associated prototype vector $\mathbf{w}_{(r_1, r_2)}$. The component planes corresponding to the four features in the Iris dataset are shown in Figure 2 in a color-coded visualization of the matrices. It can be seen, for instance, that prototypes that are neighbors in the lattice also have similar components in the four-dimensional feature space.

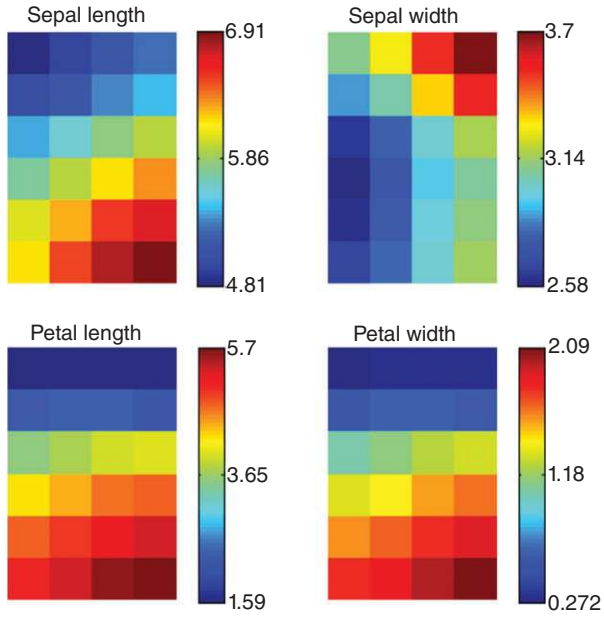

FIGURE 2 | Visualization of the component planes in an self-organizing map (SOM) with 6×4 prototypes, as obtained from the UCI Iris flower data set, see *Self-Organizing Maps* for details.

A very popular and useful tool for the visual inspection of complex data and the detection of clusters is based on the so-called **U**-matrix.[26] It has the same size and structure as the SOM lattice; i.e., for a rectangular lattice A of size $(a \times b)$ we have $\mathbf{U} \in \mathbb{R}^{a \times b}$. The matrix element U_{r_1, r_2} at position $\mathbf{r} = (r_1, r_2)$ is given by the average of the distances $d(\mathbf{w}_{\mathbf{r}}, \mathbf{w}_{\mathbf{r}'})$, where $\mathbf{w}_{\mathbf{r}'}$ denotes the prototypes associated with the neighbor sites $\mathbf{r}'$ of $\mathbf{r}$.[26] In a rectangular lattice, for instance, the coordinates of the neighbor sites of $\mathbf{r}$ satisfy the condition $\| \mathbf{r} - \mathbf{r}' \|_A^2 = 1$. Visualization of the **U**-matrix allows for the identification of clusters, since large matrix elements should mark cluster borders. They correspond to prototypes that are neighbors in the grid A but display relatively large distances in feature space. Figure 3 shows the **U**-matrix obtained from the Iris flower dataset. Its largest elements are shown in orange and red color, implying that the corresponding prototypes relate to the borders of clusters in feature space.

While the label information, i.e., the class memberships, provided in the UCI dataset was not used in the unsupervised training of the SOM, we can exploit this knowledge in the post hoc analysis of our illustrative example system. Figure 3(b), displays for each lattice site the most frequent class membership among the data points, for which the corresponding prototype is the winner. Lattice sites that are displayed without a class label correspond to prototypes

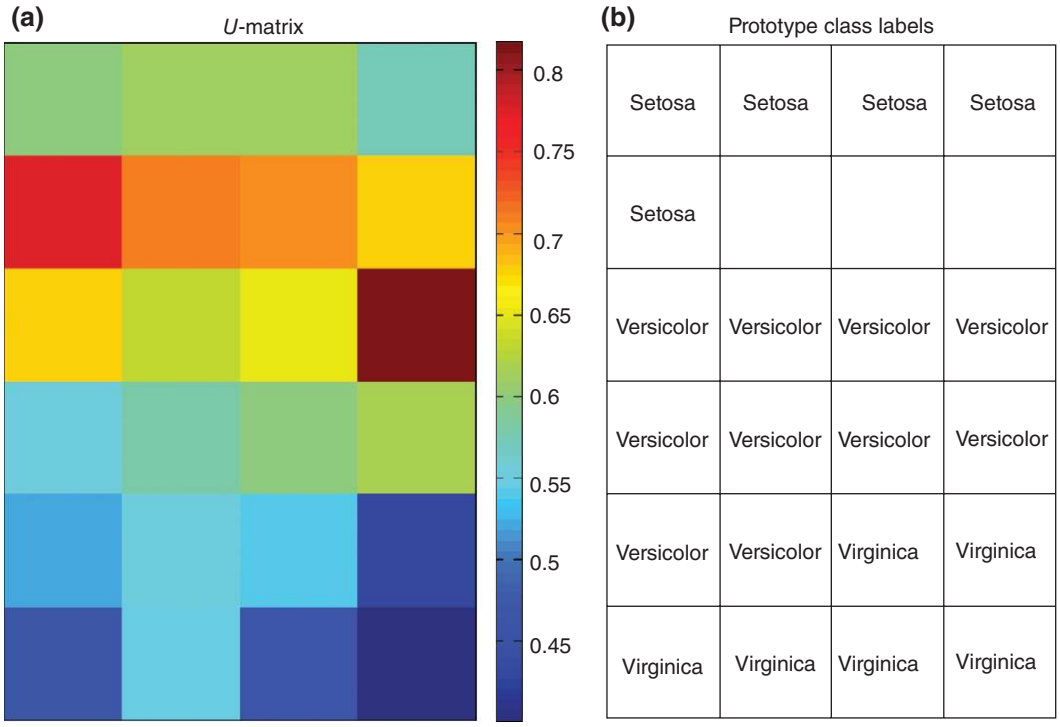

FIGURE 3 | Visualization of the *U*-matrix in an self-organizing map (SOM) with 6×4 prototypes, as obtained from the UCI Iris flower data set. Panel (a) shows the *U*-matrix elements in color-code. For comparison, panel (b) displays the post hoc class labels as assigned to the prototypes by majority vote. Empty sites correspond to prototypes with empty receptive field, which are the winner for none of the samples. See *Self-Organizing Maps* for details.

with empty receptive fields, which represent none of the data points as a *winner*. Topologically, these are obviously located between the class of *Iris setosa* and the other more closely neighbored species *Versicolor* and *Virginica*.

Evidently, the **U**-matrix also suggests a structure that is to a large extent consistent with a cluster of *Iris setosa* and a larger cluster comprising the *Versicolor* and *Virginica* samples. Note also that, although not used in the training, the class memberships are clearly reflected in a spatial ordering in the lattice, relating to the above-discussed property of topology preservation.

The SOM and NG update rules as introduced in Eqs (5) and (9) rely on a random stochastic ordering of the observed data points, such that the results of the trained networks can differ from one run to the next. Practitioners often expect a reproducible result in the sense that exactly the same visualization arises from the same data to be inspected. To this end, one can resort to batch gradient descent, which takes all data into account at once, and employ a unique initialization of the models, e.g., based on the leading principal components of the data.[27]

Intuitive data inspection as illustrated above constitutes one of the prime application scenarios of the SOM. Fields of interest are very diverse, including, for instance, financial analysis, hyperspectral image analysis, geosciences, and the retrieval of documents or other media.[28–31] In addition, the SOM can be used as a preprocessing step in machine learning problems due to its efficient and highly regularized dimensionality reduction and spatial data representation abilities. This is exploited in a variety of areas such as system identification, time series prediction, and robotics.[32–35]

Some Alternatives and Extensions

Here we highlight a few popular alternatives and extensions of unsupervised prototype-based techniques. Emphasis is on techniques that make it possible to transfer these models to settings with more general data structures as often present in practical applications. We focus on the underlying motivation of the models rather than providing their precise mathematical definition in detail.

Generative Topographic Mapping

The original SOM has been proposed on heuristic grounds and its precise mathematical treatment turned out to be a challenge.[36] Hence, some effort has been devoted to alternatives to SOM which lend themselves to a well-founded mathematical analysis and facilitate the incorporation of prior knowledge and the treatment of more complex data structures. A popular alternative to SOM, which is based on a probabilistic generative modelling of the data, is offered by the Generative Topographic Mapping (GTM).[37] GTM models represent observed data by a constrained mixture of Gaussians. The constraints enforce a smooth mapping of the Gaussian centers to a low dimensional lattice in a latent space, i.e., they account for topology preservation and visualisation capability of the model. This probabilistic treatment has the drawback that GTM learning rules are less intuitive as compared to the standard SOM. However, GTM has the benefit of a clear mathematical foundation, and the corresponding learning rules can be derived as data likelihood maximisation. This opens the way to straightforward extensions of GTM to hierarchical variants or models for complex data structures such as tree structured data. The latter are relevant, e.g., when representing logical expressions and symbolic terms. The necessary conceptual extensions can be realized by modifying the underlying probability model, see Refs 38 and 39.

Nonparametric Dimensionality Reduction

SOM is often used as an efficient nonlinear data visualization technique. As such, it competes with alternative strategies to visualize a set of given data points in the plane. The topic of nonlinear dimensionality reduction (NDR) has attracted considerable attention in recent years. Its aim is to project given high-dimensional data to two or three dimensions, with as much as possible of the structure in the data being preserved. A variety of NDR technologies is currently available, see, e.g., Ref 40 for a recent review. In contrast to SOM, many popular NDR techniques are nonparametric. Consequently, they can easily model highly nonlinear effects in the data. Unlike SOM, however, they often require at least quadratic training complexity. Moreover, they neither provide a data compression in terms of typical representatives (like the prototypes in SOM), nor do they facilitate straightforward out-of-sample extensions to novel data points which were not used for training.

SOM for Time Series Data

SOM and NG deal with vectorial data. Often, real-world data display a temporal aspect, as, for instance, high-dimensional sensor streams which are measured over time. In such cases, the goal is to inspect a sensor signal at a given time point, taking into account its temporal context. If and only if the current sensor signal and its temporal context are similar, signals should be mapped to neighboring positions of the SOM. Quite a few SOM models have been proposed that take into account the temporal dynamics of an observed signal.[41–43] Early models often rely on a leaky integration of the sensor signal, such as the so-called temporal Kohonen map or the recurrent SOM. More recent variants incorporate an explicit representation of the temporal context. Such a context representative is attached to every prototype and is learned in a similar way as the prototype position itself. Examples for this principle include recursive SOM, SOM for structured data, or merge SOM. These methods differ in the way in which the temporal context is represented, but they rely on a similar treatment of the temporal dynamics of the signal. A few overview articles are available that summarize these approaches and compare them in terms of their representation capability, see e.g., Refs 41–43.

Median, Kernel, and Relational SOM for General Proximity Data

Digital data often display a complex structure and are of Non-Euclidean nature. Consider, for example, DNA or protein sequences of different length, text fragments, symbolic expressions, or graph structures such as social networks and biological networks. Such datasets have in common that they do not allow for a lossless embedding in a finite-dimensional Euclidean vector space. Rather, data are characterized by problem-specific pairwise dissimilarities such as alignment distances for sequences, edit distances for words, tree kernels, or graph kernels.[44–46]

It is a nontrivial task to extend SOM to non-vectorial data structures. One way to deal with such data consists in an explicit embedding of the signals in a finite-dimensional vector space—however, this results in information loss and is computationally costly. It requires cubic time complexity in the worst case. Therefore, variants of SOM and NG have been designed that can directly deal with data, which are not given as vectors but which are characterized in terms of pairwise similarities or dissimilarities only. We refer to proximities in the following.

The main problem in this context is that no explicit vector space is given in which data are located, and hence it is not clear how to define and adjust prototypes. Several methods have been suggested that address this challenge. Median variants restrict prototype positions to exemplars, i.e., prototypes coincide with selected data points. Then, learning corresponds to a possibly approximate optimization of NG or SOM costs with respect to these discrete parameters.[27,47] Median variants display limited flexibility when only few data points are given, as the space of possible prototype locations is not very rich. For such settings, smooth variants have been proposed, which rely on an implicit embedding of the data in a kernel space or in a more general pseudo-Euclidean space, see, e.g., Refs 44–46. A benefit of these approaches consists in the fact that the embedding of data is only implicit. As a consequence, the computational load is only quadratic compared to the cubic complexity of explicit embeddings. Approximate schemes allow further speedup to linear time complexity.[45]

PROTOTYPES IN SUPERVISED LEARNING

In the previous section we have highlighted a few prototype-based approaches to the unsupervised analysis of potentially high-dimensional and complex data. Frequently, the aim of machine learning is the assignment of feature vectors to well-defined classes or categories. Similarly, in regression problems the task is to assign one or several continuous target values. Supervised machine learning is based on labeled example data and aims at extracting and representing information in terms of a hypothesis of the unknown classification rule or target function, which then can be generalized and applied to novel data in the working phase.

Among the many machine learning frameworks specifically designed for supervised problems, prototype-based schemes constitute a family of very intuitive, easy-to-implement systems of great flexibility. We will limit the discussion to classification problems and focus on LVQ,[2,48,49] a family of algorithms that directly take on the basic ideas of competitive learning as presented in the previous section. For comparison and motivation, we briefly discuss the classical k-nearest neighbor approach,[6,50] which treats all available data as reference exemplars for comparison with new observations.

Note that various modifications of NG and SOM have been suggested for the use in classification or other supervised tasks. We refrain from a discussion of these modifications and refer the reader to the literature. See Refs 17, 51 and 52 for examples of supervised versions of NG and SOM.

From Nearest Neighbor to Nearest Prototype Classifiers

Before discussing LVQ, we provide an intuitive introduction in terms of the simplest and most popular data-driven classifier: the Nearest neighbor (NN) classifier or its extensions to k-nearest neighbor (kNN) schemes.[3,6,50] It is roughly the machine learning counterpart of the idea of exemplars used to represent observations. In this classical approach, a given set of feature vectors is stored as the reference set, together with labels that indicate their membership to one of C classes:

$$\{\mathbf{x}^\mu, y^\mu = y(\mathbf{x}^\mu)\}_{\mu=1}^{P}, \text{ where } y^\mu \in \{1,2,\ldots C\}.$$

An arbitrary feature vector $\mathbf{x}$, representing a query, is classified according to its similarity to (or distance from) the available reference samples. After computing, e.g., its Euclidean distance from all stored feature vectors, $\mathbf{x}$ is assigned to the class of its NNs in the dataset. In the more general kNN classifier, the assignment is given by the result of a suitable voting scheme taking into account the k closest reference vectors.[50]

A kNN classifier is straightforward to implement and requires no further analysis of the example data in a training process prior to the working phase. Theoretical considerations show that kNN can achieve close to Bayes optimal classification if the number of neighbors k is selected appropriately.[6,50] Consequently, kNN methods are applied frequently and serve as a baseline to compare more sophisticated approaches with. As illustrated in Figure 4 (left panel), an NN classifier implements piecewise linear decision boundaries in feature space.

One difficulty of the approach becomes clear immediately. The naive implementation of the kNN scheme requires the computation and sorting of the distances from each available example for each query. Although sophisticated strategies for the minimum search and sorting can reduce the computational needs significantly, the basic problem remains relevant for very large datasets. Perhaps more importantly, the class boundaries can become very complex as every reference sample is taken into account explicitly. Consequently, an NN system can be highly specific to the details of the given data, potentially

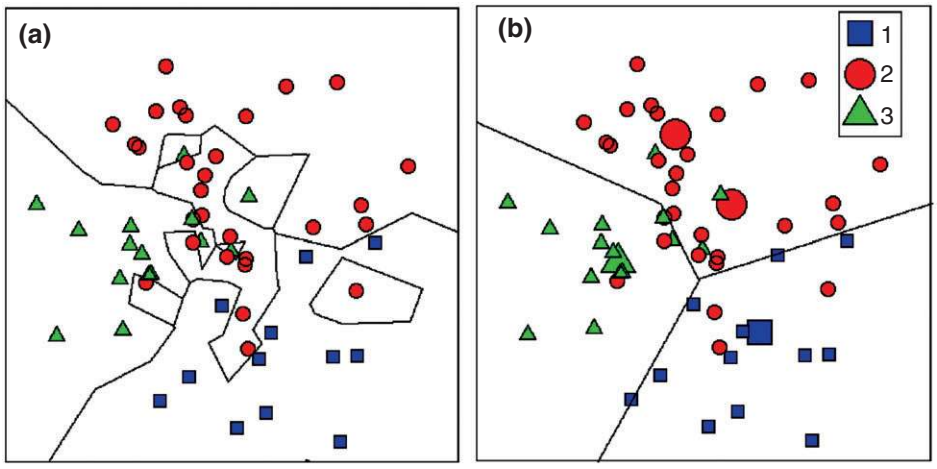

FIGURE 4 | Illustration of the *k*-nearest neighbor classifier (a) and nearest-prototype learning vector quantization (LVQ) classification (b). The same two-dimensional dataset with three different classes and piecewise linear decision boundaries is displayed in both panels, see (b) for a legend. Prototypes are marked by larger symbols in panel (b).

resulting in overfitting effects, i.e., poor ability of the classifier to generalize to novel data. The presence of single, mislabeled examples, for instance, leads to small isolated regions in feature space, which are assigned to a particular class [cf. Figure 4 (left panel)]. The effect can be counterbalanced by proper choice of the neighborhood size k, to a certain extent, but this requires large enough datasets with sufficient density of exemplars in the corresponding regions of feature space. Both problems suggest reducing the number of reference data points. Indeed, the idea of selecting a suitable subset of exemplars was already present in Ref 53 and has been picked up in more recent publications, e.g., Ref 54.

Learning Vector Quantization

A particularly attractive and intuitive approach to prototype-based classification, also suggested by Kohonen,[2,48] is conceptually very similar to unsupervised competitive learning. The so-called LVQ1 algorithm[2,55] includes the essential ingredients of many more sophisticated LVQ updates.

A single feature vector $\mathbf{x}^\mu$ with class label y^μ is sampled uniformly from the given dataset. The currently closest prototype, i.e., the winner $\mathbf{w}^*(\mathbf{x}^\mu)$, abbreviated as $\mathbf{w}^*$, is determined according to Eq. (2) and carries label $y^* = y(\mathbf{w}^*(\mathbf{x}^\mu))$. A WTA update is performed according to

$$\mathbf{w}^* \leftarrow \mathbf{w}^* + \eta\,\Psi\left(y^*, y^\mu\right)\left(\mathbf{x}^\mu - \mathbf{w}^*\right), \tag{11}$$

$$\text{where } \psi(y,\hat{y}) = \begin{cases} +1 & \text{if } y = \hat{y} \\ -1 & \text{else.} \end{cases} \tag{12}$$

In words, the winning prototype is moved closer to the example feature vector if they share the same label and moved away otherwise. One popular initialization strategy is to place prototypes close to the mean vector of the corresponding class in the dataset. Upon repeated presentation of the dataset, prototypes should represent their respective class by assuming *class-typical* positions in feature space.

The striking conceptual similarity to competitive VQ and its plausibility have contributed to the popularity of LVQ1. Note, however, that the algorithm cannot be readily interpreted as a stochastic gradient descent procedure with respect to a simple cost function analogous to H_{VQ} in the unsupervised case.

Many modifications of LVQ training have been suggested in the literature, aiming at faster convergence or better generalization behavior, see Refs 48,49 and 55 and references therein. In particular, cost-function-based approaches have been proposed that allow for training in terms of gradient descent or other optimization methods. Important examples are the so-called generalized LVQ (GLVQ), which is related to the concept of large margin classification[56,57] and robust soft LVQ (RSLVQ), which is based on the statistical modeling of the data.[58,59] The basic ingredients of most LVQ training prescriptions are, however, already apparent in heuristic LVQ1.

The main advantages of prototype-based LVQ are its flexibility and intuitive accessibility. The outcomes of the training process, i.e., the prototypes, are defined in the same space as the observed data themselves. This facilitates discussions with domain experts, as prototypes can be interpreted in the same way as original observations. This is in contrast to

many other frameworks, such as feed-forward neural networks or support vector machines (SVMs).[3,4,60,61]

In fact, as outlined in Ref 2, LVQ was motivated as an approximation of a Bayes classifier based on Gaussian mixture models,[3,6] originally. In this approach, the data densities are approximated in terms of linear combinations (mixtures) of Gaussian basis functions, and the decision boundaries are constructed accordingly. Kohonen's LVQ replaces the density estimation procedure, which turns out problematic in high-dimensional spaces, by the simpler and more robust method of VQ.

LVQ systems have displayed competitive performance in many practical applications. Various conceptual extensions of LVQ classification have been suggested in recent years, for instance, addressing the question of optimal and adaptive distance measures, which are briefly discussed in the following sections.

GENERALIZED DISSIMILARITY MEASURES AND RELEVANCE LEARNING

In the previous sections, we have introduced and illustrated several methods for the unsupervised and supervised training of prototype-based systems which all employed standard Euclidean distance as a measure of dissimilarity in feature space. Obviously, this choice comes to mind naturally, and it is the by far most popular one in the literature. However, it is important to realize that it is certainly not the only or even the best option when designing a dissimilarity-based system. Depending on the application domain and the nature of the problem at hand, other choices may be more suitable and may perform better than the standard Euclidean metric. The choice of an appropriate measure is often crucially dependent on domain-specific knowledge. It constitutes a key step in the design of any prototype-based system and can be decisive for its performance.

Beyond Euclidean Distance

It is an important strength of prototype- and distance-based systems that they are very flexible with respect to the choice of a distance measure. A large variety of measures can be used in order to quantify the dissimilarity of N-dimensional vectors. In the following, we present a few important alternatives to Euclidean metrics. An in-depth discussion, more examples, and further references can be found, e.g., in Refs 62 and 63.

Minkowski Distances

As a straightforward generalization of the standard Euclidean measure, the family of Minkowski distances of the form

$$d_P(\mathbf{x},\mathbf{y}) = \left[\sum_{j=1}^{N} |x_j - y_j|^p\right]^{1/p} \quad \text{for } \mathbf{x},\mathbf{y} \in \mathbb{R}^N \tag{13}$$

fulfill metric properties (for $p \geq 1$) and include the Euclidean metrics as a special case with $p = 2$. By choice of the parameter p, emphasis can be put on smaller or larger deviations $|x_j - y_j|$: In the limit $p \to \infty$, for instance, the Minkowski distance is equivalent to

$$d_\infty(\mathbf{x},\mathbf{y}) = \max_{j=1,\ldots N} |x_j - y_j|,$$

which considers only the component with the largest difference between the two vectors. Employing Minkowski distances with $p \neq 2$ has proven advantageous in several practical applications, see, e.g., Refs 64 and 65.

Mahalanobis Distance

The naive Euclidean measure takes into account all features with the same weight. A major problem of this naive approach is due to the fact that the features in real-world data can display very different magnitudes and variabilities. This is obviously the case if features correspond to quantitatively or qualitatively different properties. Their relative importance in the evaluation of Euclidean distances strongly depends on the actual numerical representation and the choice of unit systems and scales. Potential correlations between different features can also mislead systems relying on Euclidean distance. The presence of highly correlated dimensions of the feature space can dominate the dissimilarity and reduce the influence of important discriminative features.

Statistical properties of the example data can be taken into account explicitly by using a popular dissimilarity measure, the so-called Mahalanobis distance, which was suggested very early.[66] It can be interpreted as a measure comparing two random vectors $\mathbf{x}$, $\mathbf{y}$ that are drawn from the same distribution. The distance

$$d_M(\mathbf{x},\mathbf{y}) = \left[(\mathbf{x}-\mathbf{y})^T C^{-1} (\mathbf{x}-\mathbf{y})\right]^{1/2} \tag{14}$$

takes into account the (empirical) covariance matrix C of the data. Replacing C by the $(N \times N)$ identity

matrix recovers Euclidean distance, while a diagonal approximation of C would rescale all features according to their standard deviation observed in the dataset. The Mahalanobis distance is widely used in the context of the unsupervised and supervised analysis of given datasets, see Ref 6 for a more detailed discussion.

Kernelized Distances

Another route to more general measures is based on the observation that the squared Euclidean distance can be rewritten as

$$d_2(\mathbf{x},\mathbf{w})^2 = [\mathbf{x}\cdot\mathbf{x} - 2\mathbf{x}\cdot\mathbf{w} + \mathbf{w}\cdot\mathbf{w}]. \qquad (15)$$

In so-called kernelized distances, all inner products of the form $\mathbf{x}\cdot\mathbf{w} = \sum_j x_j w_j$ in Eq. (15) are replaced by a suitable kernel function $\kappa(\mathbf{x}, \mathbf{w})$:

$$d_\kappa(\mathbf{x},\mathbf{w})^2 = [\kappa(\mathbf{x},\mathbf{x}) - 2\kappa(\mathbf{x},\mathbf{w}) + \kappa(\mathbf{w},\mathbf{w})]. \qquad (16)$$

A proper kernel function corresponds to an inner product of transformed feature vectors.[60,61,67] Consider the simple example of a quadratic kernel

$$\kappa_{\text{quad}}(\mathbf{x},\mathbf{w}) = [1 + \mathbf{x}\cdot\mathbf{w}]^2.$$

With the definition $\mathbf{x}\cdot\mathbf{w} = \sum_{j=1}^{N} x_j w_j$ of the conventional inner product in N dimensions, $\kappa_{\text{quad}}(\mathbf{x}, \mathbf{w})$ can be rewritten as

$$\kappa_{\text{quad}}(\mathbf{x},\mathbf{w}) = 1 + 2\sum_j x_j w_j + \sum_j x_j^2 w_j^2 + 2\sum_{j,k>j} (x_j x_k)(w_j w_k).$$

We observe that the latter can be interpreted as the inner product of two higher-dimensional vectors:

$$\kappa_{\text{quad}}(\mathbf{x},\mathbf{w}) = \Phi(\mathbf{x})\cdot\Phi(\mathbf{w}) \text{ with}$$

$$\Phi(\mathbf{y}) = \left[1, \sqrt{2}y_1, \ldots, \sqrt{2}y_N, y_1^2, \ldots, y_N^2, \sqrt{2}y_1y_2, \ldots \sqrt{2}y_{N-1}y_N\right],$$

where the $(N^2/2 + 3N/2 + 1)$-dimensional vector $\Phi(\mathbf{y})$ concatenates the constant 1, all original components y_j, their squares y_j^2, and all products $y_j y_k$ with $j \neq k$. Hence, the kernel is associated with a nonlinear mapping $\mathbf{y} \rightarrow \Phi(\mathbf{y})$ to a higher dimensional space, in which the kernel function represents the inner product.

The so-called kernel trick, as used in the context of the SVM, exploits the fact that the transformation need not be specified explicitly. Instead, it is possible, in practice, to start from an appropriate kernel function that implicitly represents a highly nonlinear mapping to high dimensions, provided the kernel satisfies specific mathematical conditions.[60,61]

A particularly prominent example is the Gaussian kernel

$$\kappa_{\text{G}}(\mathbf{x},\mathbf{w}) = \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-\mathbf{w})^2}{\sigma^2}\right),$$

where σ is the kernel width.[60,61]

In SVM training, one takes advantage of the fact that data that is difficult to separate in the original feature space can become linearly separable in the (implicitly defined) high-dimensional transformed space, provided an appropriate kernel is chosen.[60,61] In the above explicit example of the quadratic kernel κ_{quad}, a linear decision boundary in the space of vectors $\Phi(\mathbf{x})$ would obviously correspond to a quadratic separating function in terms of $\mathbf{x} \in \mathbb{R}^N$.

Similarly, kernelized distances can be used to translate nonlinear complex classification tasks into simpler problems in a higher dimensional transformed feature space, see Ref 68 for an example in the context of distance based face recognition.

Divergences

As a last example we discuss the use of statistical divergences as dissimilarity measures. Many practical problems involve the description of observations in terms of densities or histograms. As just two examples, images are often represented by color or other histograms and pieces of text can be characterized by word counts in a *bag of words* framework. Hence, the comparison of sample data amounts to evaluating the dissimilarity of histograms in such cases. Emphasis could be on discriminating multimodal from unimodal histograms or on detecting the presence of characteristic peaks. The Euclidean metric does not take into account the particular nature of these tasks and the efficient comparison of histograms can benefit greatly from the use of measures that were devised specifically for this type of problem.

A large variety of statistical divergences can be considered in this context.[69] Perhaps the most prominent one is the nonsymmetric Kullback–Leibler divergence.[6] Another example would be the so-called Cauchy–Schwarz divergence[69,70]

$$\delta_{\text{CS}}(\mathbf{x},\mathbf{y}) - \frac{1}{2}\log[(\mathbf{x}\cdot\mathbf{x})(\mathbf{y}\cdot\mathbf{y})] - \log[\mathbf{x}\mathbf{y}], \qquad (17)$$

which satisfies $\delta_{\text{CS}}(\mathbf{x}, \mathbf{y}) = \delta_{\text{CS}}(\mathbf{y}, \mathbf{x})$.

In the context of protoytpe-based learning, divergences are frequently employed to quantify the

similarity of an entire population, i.e., density of prototypes with the actual density of observed data, see Refs 71 and 72 for examples in the context of VQ and SOM, respectively. Here, however, we are concerned with situations where an individual feature vector $\mathbf{x}$ itself can be interpreted as a density or histogram, e.g., a color histogram corresponding to an observed image. Then, divergences can be used to measure directly the dissimilarity between individual data points, or data points and prototypes.

Note that meaningful dissimilarities do not necessarily satisfy the properties of a metric in the N-dimensional space. For instance, kernelized distances need not obey the triangular inequality in the original feature space, and many divergences are nonsymmetric. Consider, as an example, the family of γ-divergences

$$\delta_\gamma(\mathbf{x},\mathbf{y}) = \frac{1}{\gamma+1}\log\left[\left(\sum_j x_j^{\gamma+1}\right)^{1/\gamma}\left(\sum_j y_j^{\gamma+1}\right)\right] - \log\left[\left(\sum_j x_j y_j^{\gamma}\right)^{1/\gamma}\right], \tag{18}$$

which reduces to the Cauchy–Schwarz divergence for $\gamma = 1$ and to the well-known Kullback–Leibler divergence in the limit $\gamma \to 0$.[69] For $\gamma \neq 1$, the measure is nonsymmetric with $\delta_\gamma(\mathbf{x}, \mathbf{y}) \neq \delta_\gamma(\mathbf{y}, \mathbf{x})$ if $\mathbf{x} \neq \mathbf{y}$.

Metric properties are, however, not strictly required in prototype-based systems where the dissimilarity measure is used to determine the distance of a specific feature vector from the prototypes only. Pairwise distances between data points are not considered explicitly and, therefore, symmetry is not essential. For example, a nonsymmetric dissimilarity $\delta_\gamma(\mathbf{x}, \mathbf{w})$ could be readily employed in the identification of the winning prototype for a given data point in VQ or LVQ schemes. Obviously, one should use only one of the two versions $\delta_\gamma(\mathbf{x}, \mathbf{w})$ or $\delta_\gamma(\mathbf{w}, \mathbf{x})$ consistently in training and working phase.[62,70] An application example for the use of nonsymmetric divergences in LVQ classifiers is presented in Ref 70. There, γ-divergence based LVQ systems clearly outperform their Euclidean counterpart and, moreover, nonsymmetric choices with $\gamma \neq 1$ are superior to using the symmetric Cauchy–Schwarz divergence.

Prototype Training for Generalized Distances

All training prescriptions discussed in the earlier sections based on Euclidean distance lead to prototype updates where prototypes $\mathbf{w}$ are displaced by a vector proportional to $(\mathbf{x}^\mu - \mathbf{w})$ upon presentation of the feature vector $\mathbf{x}^\mu$. Noting that for $\delta = \frac{1}{2}(\mathbf{w}-\mathbf{x})^2$ we have

$$(\mathbf{x}-\mathbf{w}) = -\frac{\partial}{\partial \mathbf{w}}\delta(\mathbf{w},\mathbf{x}). \tag{19}$$

We can rewrite the gradient descent updates, Eqs. (3), (5), (9), and (11), by inserting the relation (19). For example, the NG updates (5) can be written as

$$\mathbf{w}^j \leftarrow \mathbf{w}^j - \eta \cdot h_\lambda^{\mathrm{NG}}\left(k_j(\mathbf{x}^\mu, W)\right)\frac{\partial \delta\left(\mathbf{w}^j, \mathbf{x}^\mu\right)}{\partial \mathbf{w}^j}, \tag{20}$$

and the WTA update in LVQ1 (11) becomes

$$\mathbf{w}^* \leftarrow \mathbf{w}^* - \eta\, \Psi\left(y^*, y^\mu\right)\frac{\partial \delta(\mathbf{w}^*, \mathbf{x}^\mu)}{\partial \mathbf{w}^*}. \tag{21}$$

Similar forms are obtained for competitive VQ and SOM updates in a straightforward fashion.

We can now replace δ with more general distance measures and readily obtain the corresponding update rules, provided δ is differentiable with respect to the prototype components. Obviously, for the sake of consistency, the evaluation of ranks in NG and the winner identification in VQ, SOM, and LVQ have to be done according to the same generalized dissimilarity δ that is used in the actual update.

Along these lines, a variety of training prescriptions can be constructed that use, for instance, differentiable dissimilarities such as divergences, kernelized distances with differentiable kernels, or other specific measures. Corresponding training prescriptions have been devised in a variety of prototype-based frameworks, e.g., Refs 70, 73 and 74. Nondifferentiable measures can also be considered if differentiable approximations are available, as, for instance, in the case of the *Manhattan distance* given by Eq. (13) with $p = 1$.[75] Cost-function-based approaches can also employ nondifferentiable measures if one resorts to more sophisticated optimization strategies that do not require gradients.[10]

Adaptive Distances in Relevance Learning

In the previous subsection, we discussed a few alternative dissimilarity measures that can be employed in prototype-based learning. In a particular problem, a specific measure could be selected based on domain knowledge or preliminary analysis of the dataset.

The framework of relevance learning constitutes a considerable conceptual extension of dissimilarity-based systems. In this elegant approach, only the parametric form of the dissimilarity or distance is fixed *a priori*. Its parameters are then optimized in a data-driven training phase. It is a plausible and very appealing concept that not only prototype representations may depend on the data encountered in the learning process. Observations should also shape the definition of similarity or dissimilarity.

The basic concept is extremely versatile and can be employed in a variety of supervised and (appropriately constrained) unsupervised learning tasks. We present here only one very clear-cut example of relevance learning, extending the framework of supervised LVQ for classification, cf. *Learning Vector Quantization*.

The framework of matrix relevance LVQ was introduced and extended in Refs 76–78. It has proven useful in a number of applications from the biomedical and other domains.[79–82] Similar to various other schemes,[83–86] it employs a generalization of standard Euclidean distance given by the quadratic form

$$\delta_\Lambda(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T \Lambda (\mathbf{x}-\mathbf{y}) = \sum_{i=1}^{N}\sum_{j=1}^{N}(x_i-y_i)\Lambda_{ij}\left(x_j-y_j\right), \tag{22}$$

with the so-called relevance matrix $\Lambda \in \mathbb{R}^{N\times N}$ of parameters. It is formally reminiscent of the quadratic Mahalanobis distance defined in Eq. (14). However, the matrix Λ is not explicitly computed from the data. Its elements are considered free parameters of the system. Diagonal entries of Λ control the importance of single feature dimensions in the distance and can account for their potentially different scaling. Off-diagonal elements relate to the contribution of pairs of features and enable the system to adapt to interdependencies and correlations among the features.

In order to obtain a meaningful dissimilarity, $\delta_\Lambda(\mathbf{x}, \mathbf{y}) \geq 0$ should hold for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$. This minimal requirement can be guaranteed by an additional parameterization in terms of an auxiliary matrix $\Omega \in \mathbb{R}^{N\times N}$ with

$$\Lambda = \Omega^T\Omega, \quad \text{i.e., } \delta_\Lambda(\mathbf{x},\mathbf{y}) = [\Omega(\mathbf{x}-\mathbf{y})]^2, \tag{23}$$

which also implies the symmetries $\Lambda_{ij} = \Lambda_{ji}$ and $\delta_\Lambda(\mathbf{x}, \mathbf{y}) = \delta_\Lambda(\mathbf{y}, \mathbf{x})$. Hence, δ_Λ can be interpreted as the Euclidean distance after a linear transformation of feature space given by Ω. Note that δ_Λ is only a *pseudo-metric* in $\mathbb{R}^N$, in general: Λ can become singular, implying that $\delta_\Lambda(\mathbf{x}, \mathbf{y}) = 0$ *for* $\mathbf{x} \neq \mathbf{y}$ is possible.

The key idea is to treat the elements of Λ or Ω, respectively, as adaptive parameters in the training process. To this end, the heuristic WTA update of LVQ1 can be extended by a simultaneous modification of prototypes and matrix Ω when presenting example $\mathbf{x}^\mu$ with label y^μ:

$$\begin{aligned}
\mathbf{w}^* &\leftarrow \mathbf{w}^* - \eta\Psi(y^*,y^\mu)\frac{\partial\delta_\Lambda(\mathbf{w}^*,\mathbf{x}^\mu)}{\partial\mathbf{w}^*}\\
&= \mathbf{w}^* + \eta\Psi(y^*,y^\mu)\Omega^T\Omega(\mathbf{x}^\mu-\mathbf{w}^*)\\
\Omega &\leftarrow \Omega - \eta_\Omega\Psi(y^*,y^\mu)\frac{\partial\delta_\Lambda(\mathbf{w}^*,\mathbf{x}^\mu)}{\partial\Omega}\\
&= \Omega - \eta_\Omega\Psi(y^*,y^\mu)\Omega(\mathbf{x}^\mu-\mathbf{w}^*)(\mathbf{x}^\mu-\mathbf{w}^*)^T
\end{aligned} \tag{24}$$

with $\mathbf{w}^*$, y^* and the label-dependent factor $\Psi(y^*, y^\mu) = \pm 1$ as defined in Eq. (11). Here we exploit the idea outlined in *Prototype Training for Generalized Distances* with respect to generalized prototype updates. In addition, the matrix Ω is modified as to increase or decrease the distance $\delta_\Lambda(\mathbf{w}^*, \mathbf{x}^\mu)$ explicitly, depending on the class labels of prototype and sample vector. The matrix Ω can be initialized randomly or as the N-dimensional identity, for instance. The step size η_Ω is frequently chosen as $\eta_\Omega \ll \eta$, which implies that the distance measure changes slowly compared to the prototypes. For the sake of numerical stability and better comparison of different relevance matrices, usually a normalization to $\sum_i \Lambda_{ii} = \sum_{i,j}\Omega_{ij}^2 = 1$ is imposed after each update step.[77]

Cost-function-based schemes can be extended by an adaptive relevance matrix, analogously; see Refs 77 and 87. In addition, a number of modifications and extensions have been suggested, including the use of local matrices, regularization schemes, or the restriction to rectangular matrices Ω. We refer the interested reader to the literature concerning these and many other variants of the basic scheme.[76,78,87] Ready-to-use MATLAB implementations of generalized matrix relevance LVQ (GMLVQ)-based on the GLVQ cost function and several variants thereof are available in Refs 88 and 89.

In the following, we highlight a few attractive features of matrix relevance learning in the context of a well-known classification problem. To this end, we revisit the Iris flower data set with $N = 4$, already introduced in *Self-Organizing Maps* in the context of the SOM. In contrast to unsupervised SOM training,

here we make explicit use of the given class label information.

An LVQ system with one prototype per class and a relevance matrix $\Lambda = \Omega^T\Omega$ with $\Omega \in \mathbb{R}^{4\times 4}$ was trained using the GMLVQ code available from Ref 88. Prior to training, a z-score transformation was performed, resulting in rescaled features with empirical mean $\frac{1}{P}\sum_{\mu=1}^{150} x_j^\mu = 0$ and unit variance in the dataset. The resulting LVQ system achieves very low classification error of the training data. Applying cross-validation strategies,[3,4,6] we observe also very good generalization behavior for this relatively simple classification problem. In particular, the class of *Iris setosa* is perfectly separated from the other two.

Figure 5 shows the classifier as obtained from the entire dataset. In panel (a), the prototype components are displayed. Panel (b) shows the outcome of the training process in terms of the relevance matrix. According to Eq. (22), the diagonal elements Λ_{ii} as displayed in the bar graph can be interpreted as the relevance of feature i for the classification. Apparently, features 3 (petal length) and 4 (petal width) seem to be most important for the discrimination of the classes by GMLVQ. The color-coded off-diagonal elements (lower graph) represent the contribution of pairs of different features. Here, also the interplay of petal length and petal width (element $\Lambda_{3,4} = \Lambda_{4,3}$) appears to be most significant.

In more complex datasets, the analysis of the relevance matrix provides valuable insights into the nature of the problem and the dataset. In several real-world applications, matrix relevances have been used to identify most relevant or irrelevant features.[79,80] In this sense, matrix relevance learning could be used as a heuristic approach to feature selection.

The last aspect we wish to highlight relates to the empirical observation that the relevance matrix Λ displays a tendency to become singular under updates of the form (24) or related training schemes

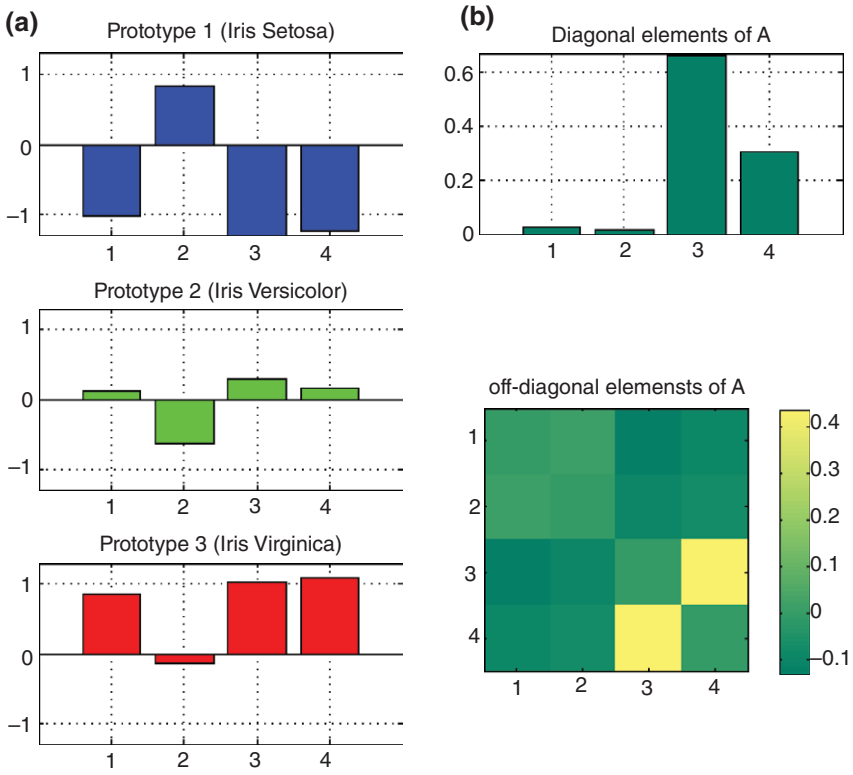

FIGURE 5 | Visualization of the generalized matrix relevance learning vector quantization (GMLVQ) systems obtained for the z-score transformed Iris flower data set, see *Adaptive Distances in Relevance Learning* for details. Panel (a) displays the class prototypes bar plots with respect to the four feature space components corresponding to sepal length (feature 1), sepal width (2), petal length (3), and petal width (4). Panel (b) shows diagonal elements of Λ as a bar plot (upper) and off-diagonal elements in color code (lower).

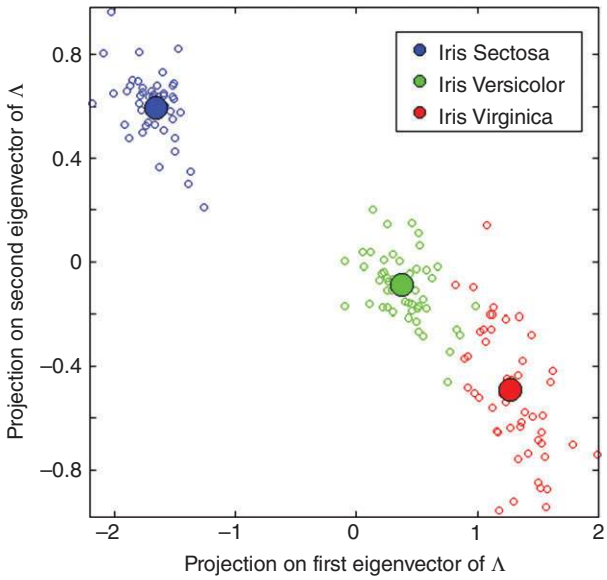

FIGURE 6 | Visualization of the Iris flower data set (small symbols, see legend for class memberships) and corresponding GMLVQ prototypes (large symbols) in the space spanned by the two leading eigenvalues of Λ. See *Adaptive Distances in Relevance Learning* for details.

like GMLVQ. Frequently, the relevance matrix is, after training, dominated by very few eigenvectors; a discussion and examples of this property can be found in, e.g., Refs 76,78 and 80.

Effectively, this reduces the number of degrees of freedom and reduces overfitting effects, which might be expected in high-dimensional spaces where Λ introduces on the order of N^2 adaptive parameters. As a beneficial side effect, the strong singularity of Λ facilitates the low-dimensional discriminative representation and visualization of labeled datasets. Figure 6 shows the visualization of the Iris flower data in terms of their projections onto the two leading eigenvectors of Λ as an example. Note that the cluster structure displayed in this discriminative visualization is consistent with our analysis of the SOM-related U-matrix in *Self-Organizing Maps*: the *Iris setosa* samples form a distinct cluster, well separated from *Versicolor* and *Virginica*.

Structure Metrics in LVQ

The metric learning approaches as introduced in *Beyond Euclidean Distance* have in common that the involved distances are differentiable with respect to the prototypes, such that prototype updates can be based on gradient schemes. For many complex discrete data structures and popular distance measures, this is not the case. Consider, for example, DNA sequences that are compared by pairwise sequence alignment, biological networks that are compared based on graph kernels which count the common substructures of two graphs, or logical terms that are compared based on tree kernels. These proximity measures can act on discrete data structures for which a lossless Euclidean embedding does not exist (e.g., in case of a non-Euclidean proximity such as sequence alignment) or for which a Euclidean embedding is costly to compute (e.g., in case of a structure kernel). Hence, gradient-based LVQ learning prescriptions cannot be used in such settings, since a smooth vectorial embedding of the given data structures is not available.

As for the kernel and relational SOM mentioned above, a few extensions of LVQ have been proposed recently for such situations. These so-called kernel or relational LVQ variants rely on an implicit embedding of data only. They allow efficient training of LVQ schemes, even if data are characterized by pairwise proximities only. A general framework that includes existing popular approaches has been presented recently in Ref 90.

Similar to vector space metrics, proximity measures for structures often incorporate crucial metric parameters. One example is the ground metric (scoring matrix) for sequence alignment[91]: It determines the costs that arise when two symbols a and b are aligned. The overall costs are computed as the sum of all costs for aligned pairs of symbols. The overall result of a neighbor computation is crucially determined by these ground metric parameters, and quite some work has been conducted to determine suitable scoring matrices based on data, e.g., in the context of DNA and protein alignment.[91] Similar to the learning of a quadratic distance for vectorial LVQ, efficient adaptation schemes for such structure metric parameters within relational LVQ have been proposed.[92]

CONCLUDING REMARKS

Prototype-based systems constitute a highly attractive and versatile framework for the unsupervised and supervised analysis of complex, potentially high-dimensional data. The conceptual simplicity and interpretability facilitates efficient exchange with the domain experts and promotes trans-disciplinary collaborations. Together with the concept of (dis-)similarity, they provide a framework in which to develop

novel efficient training schemes which can be useful in a variety of application areas.

This work can only provide a first illustrative introduction and certainly falls short of giving a complete overview and providing a comprehensive literature review. The suggested references can merely serve as a starting point for the exploration of this active area of research.

REFERENCES

1. Jäkel F, Bernhard S, Wichmann FA. Generalization and similarity in exemplar models of categorization: insights from machine learning. *Psychon Bull Rev* 2008, 15:256–271.
2. Kohonen T. *Self-Organizing Maps*. 2nd ed. Berlin, Heidelberg: Springer; 1997.
3. Bishop CM. *Pattern Recognition and Machine Learning*. Cambridge, UK: Cambridge University Press; 2007.
4. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer; 2009.
5. Martinetz TM, Berkovich SG, Schulten KJ. Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Trans Neural Netw* 1993, 4:558–569.
6. Duda RO, Hart PE, Stork DG. *Pattern Classification*. 2nd ed. Hoboken, NJ: John Wiley & Sons; 2001.
7. Lloyd SP. Least square quantization in PCM. *IEEE Trans Inf Theory* 1982, 28:129–137.
8. Ritter H, Martinetz T, Schulten K. *Neural Computation and Self-Organizing Maps: An Introduction*. Computation and Neural Systems Series. Boston, MA: Addison-Wesley Publishing Company; 1992.
9. Robbins H, Monro S. A stochastic approximation method. *Ann Math Stat* 1951, 22:405.
10. Sra S, Nowozin S, Wright SJ. *Optimization for Machine Learning*. Cambridge, MA: MIT Press; 2011, 512 pp.
11. Barros Magalhães Netto S, Corrê Silva A, Acatauassú Nunes R, Gattass M. Automatic segmentation of lung nodules with growing neural gas and support vector machine. *Comput Biol Med* 2012, 42:1110–1121.
12. Estévez PA, Figueroa CJ. Online data visualization using the neural gas network. *Neural Netw* 2006, 19:923–934.
13. Labusch K, Barth E, Martinetz T. Sparse coding neural gas for the separation of noisy overcomplete sources. In: Kurková V, Neruda R, Koutnìk R, eds. *Artificial Neural Networks - ICANN 2008, 18th International Conference, Proceedings, Part II*, Prague, Czech Republic, September 3–6, 2008. Berlin, Heidelberg: Springer; 2008, 788–797.
14. Qin AK, Suganthan PN. Kernel neural gas algorithms with application to cluster analysis. In: *ICPR 2004. Proceedings of the 17th International Conference on Pattern Recognition, 2004*, vol. 4. New York, NY: IEEE; 2004, 617–620.
15. Strickert M, Hammer B. Neural gas for sequences. In: *Proceedings of the Workshop on Self-Organizing Maps (WSOM'03)*. Hibikino, Kitakyushu, Japan: Kyushu Institute of Technology; 2003, 53–57.
16. Walter JA, Martinetz T, Schulten KJ. Industrial robot learns visuo-motor coordination by means of the neural gas. In: Kohonen T, ed. *Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)*, vol. 1. Amsterdam, The Netherlands: North Holland: 1991, 86–96.
17. Kohonen T. *MATLAB Implementations and Applications of the Self-Organizing Map*. Helsinki: Unigrafia Oy; 2014. Available at: http://docs.unigrafia.fi/publications/kohonen_teuvo. (Accessed December 30, 2015).
18. Kohonen T. Learning vector quantization for pattern recognition. Technical Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.
19. Heskes T. Energy functions for self-organizing maps. In: Oja E, Kaski S, eds. *Kohonen Maps*. Amsterdam: Elsevier; 1999, 303–315.
20. Villmann T, Der R, Herrmann M, Martinetz TM. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Trans Neural Netw* 1997, 8:256–266.
21. Bauer H-U, Pawelzik KR. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans Neural Netw* 1992, 3:570–579.
22. Vesanto J. SOM-based data visualization methods. *Intell Data Anal* 1999, 3:111–126.
23. Vesanto J, Alhoniemi E, Himberg J, Kiviluoto K, Parviainen J. Self-organizing map for data mining in MATLAB: the SOM toolbox. *Simul News Europe* 1999, 25:54.
24. Fisher RA. The use of multiple measurements in taxonomic problems. *Ann Eugenics* 1936, 7:179–188.
25. Lichman M. UCI machine learning repository. Available at: http://archive.ics.uci.edu/ml. (Accessed December 30, 2015).
26. Ultsch A, Siemon HP. Kohonen's self organizing feature maps for exploratory data analysis. In: *Proceedings. INNC'90, International Neural Network Conference*. Dordrecht: Kluwer; 1990, 305–308.

27. Cottrell M, Hammer B, Hasenfuss A, Villmann T. Batch and median neural gas. *Neural Netw* 2006, 19:762–771.
28. Bação F, Lobo VS, Painho M. The self-organizing map, the Geo-SOM, and relevant variants for geosciences. *Comput Geosci* 2005, 31:155–163.
29. Laaksonen J, Koskela M, Oja E. Picsom-self-organizing image retrieval with MPEG-7 content descriptors. *IEEE Trans Neural Netw* 2002, 13:841–853.
30. Lagus K, Honkela T, Kaski S, Kohonen T. Websom for textual data mining. *Artif Intell Rev* 1999, 13:345–364.
31. Merényi E, Farrand WH, Taranik JV, Minor TB. Classification of hyperspectral imagery with neural networks: comparison to conventional tools. *EURASIP J Adv Signal Proc* 2014, 71:2014.
32. Barreto GA, Araújo AFR, Ritter HJ. Self-organizing feature maps for modeling and control of robotic manipulators. *J Intell Robot Syst* 2003, 36:407–450.
33. Frota RA, Barreto GA, Mota JCM. Anomaly detection in mobile communication networks using the self-organizing map. *J Intell Fuzzy Syst* 2007, 18:493–500.
34. Souza Júnior AH, Barreto GA, Corona F. Regional models: a new approach for nonlinear system identification via clustering of the self-organizing map. *Neurocomputing* 2015, 147:31–46.
35. Simon G, Lee JA, Cottrell M, Verleysen M. Forecasting the CATS benchmark with the double vector quantization method. *Neurocomputing* 2007, 70:2400–2409.
36. Fort J-C. SOM's mathematics. *Neural Netw* 2006, 19:812–816.
37. Bishop CM, Svensén M, Williams CKI. GTM: the generative topographic mapping. *Neural Comput* 1998, 10:215–234.
38. Gianniotis N, Tiňo P. Visualization of tree-structured data through generative topographic mapping. *IEEE Trans Neural Netw* 2008, 19:1468–1493.
39. Tiňo P, Nabney IT. Hierarchical GTM: constructing localized nonlinear projection manifolds in a principled way. *IEEE Trans Pattern Anal Mach Intell* 2002, 24:639–656.
40. Gisbrecht A, Hammer B. Data visualization by nonlinear dimensionality reduction. *WIREs Data Min Knowl Discov* 2015, 5:51–73.
41. Barreto GA, Araújo AF, Kremer SC. A taxonomy for spatiotemporal connectionist networks revisited: the unsupervised case. *Neural Comput* 2003, 15:1255–1320.
42. Hammer B, Micheli A, Sperduti A, Strickert M. A general framework for unsupervised processing of structured data. *Neurocomputing* 2004, 57:3–35.
43. Hammer B, Micheli A, Sperduti A, Strickert M. Recursive self-organizing network models. *Neural Netw* 2004, 17:1061–1085.
44. Boulet R, Jouve B, Rossi F, Villa N. Batch kernel SOM and related laplacian methods for social network analysis. *Neurocomputing* 2008, 71:1257–1273.
45. Hammer B, Hasenfuss A. Topographic mapping of large dissimilarity data sets. *Neural Comput* 2010, 22:2229–2284.
46. Yin H. On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural Netw* 2006, 19:780–784.
47. Kohonen T, Somervuo P. How to make large self-organizing maps for nonvectorial data. *Neural Netw* 2002, 15:945–952.
48. Kohonen T. Improved versions of learning vector quantization. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, vol. 1. New York, NY: IEEE: 1990, 545–550.
49. Nova D, Estévez PA. A review of learning vector quantization classifiers. *Neural Comput Appl* 2014, 25:511–524.
50. Cover TM, Hart PE. Nearest neighbor pattern classification. *IEEE Trans Inf Theory* 1967, 13:21–27.
51. Hagenbuchner M, Tsoi AC, Sperduti A. A supervised self-organizing map for structured data. In: *Advances in Self-Organising Maps*. Berlin, Heidelberg: Springer; 2001, 21–28.
52. Hammer B, Strickert M, Villmann T. Supervised neural gas with general similarity measure. *Neural Process Lett* 2005, 21:21–44.
53. Hart PE. The condensed nearest neighbor rule. *IEEE Trans Inf Theory* 1968, 14:515–516.
54. Wu Y, Ianakiev K, Govindaraju V. Improved *k*-nearest neighbor classification. *Pattern Recogn* 2002, 35:2311–2318.
55. Biehl M, Ghosh A, Hammer B. Dynamics and generalization ability of LVQ algorithms. *J Mach Learn Res* 2007, 8:323–360.
56. Sato A, Yamada K. Generalized learning vector quantization. In: Mozer MC, Touretzky DS, Hasselmo ME, eds. *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*. Cambridge, MA: MIT Press; 1996, 423–429.
57. Sato A, Yamada K. An analysis of convergence in generalized LVQ. In: Niklasson L, Bodéén M, Ziemke T, eds. *Proceedings of the International Conference on Artificial Neural Networks*. Berlin, Heidelberg: Springer; 1998, 170–176.
58. Seo S, Bode M, Obermayer K. Soft nearest prototype classification. *IEEE Trans Neural Netw* 2003, 14:390–398.
59. Seo S, Obermayer K. Soft learning vector quantization. *Neural Comput* 2003, 15:1589–1604.

60. Schölkopf B, Smola A. *Learning with Kernels*. Cambridge, MA: MIT Press; 2002.
61. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press; 2004, 474 pp.
62. Biehl M, Hammer B, Villmann T. Distance measures for prototype based classification. In: Grandinetti L, Lippert T, Petkov N, eds. *Brain-Inspired Computing*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer International Publishing; 2014, 100–116.
63. Hammer B, Villmann T. Classification using non-standard metrics. In: Verleysen M, ed. *European Symposium on Artificial Neural Networks, ESANN 2005*. Evere, Belgium: d-side publishing; 2005, 303–316.
64. Biehl M, Breitling R, Li Y. Analysis of tiling microarray data by learning vector quantization and relevance learning. In: Yin H, Tino P, Corchado E, Byrne W, Yao X, eds. *Intelligent Data Engineering and Automated Learning, IDEAL 2007*. Lecture Notes in Computer Science, vol. 4881. Berlin, Heidelberg: Springer; 2007, 880–889.
65. Golubitsky O, Watt SM. Distance-based classification of handwritten symbols. *Int J Doc Anal Recognit* 2010, 13:133–146.
66. Mahalanobis PC. On the generalised distance in statistics. *Proc Natl Instit Sci India* 1936, 2:49–55.
67. Schölkopf B. The kernel trick for distances. In: Leen TK, Dietterich TG, Tresp V, eds. *Proceedings Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press; 2001, 301–307.
68. Villmann T, Kästner M, Nebel D, Riedel M. ICMLA face recognition challenge – results of the team 'Computational Intelligence Mittweida'. In: *Proceedings of the International Conference on Machine Learning Applications (ICMLA'12)*. New York, NY: IEEE Computer Society Press; 2012, 7–10.
69. Cichocki A, Zdunek R, Phan A, Amari S-I. *Nonnegative Matrix and Tensor Factorizations*. Hoboken, NJ: John Wiley & Sons; 2009.
70. Mwebaze E, Schneider P, Schleif F-M, Aduwo JR, Quinn JA, Haase S, Villmann T, Biehl M. Divergence based classification and learning vector quantization. *Neurocomputing* 2011, 74:1429–1435.
71. Banerjee A, Merugu S, Dhillon IS, Ghosh J. Clustering with Bregman divergences. *J Mach Learn Res* 2005, 6:1705–1749.
72. Jang E, Fyfe C, Ko H. Bregman divergences and the self organising map. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2008*. Berlin, Heidelberg: Springer; 2008, 452–458.
73. Inokuchi R, Miyamoto S. LVQ clustering and SOM using a kernel function. In: *Proceedings 2004 IEEE International Conference on Fuzzy Systems*, vol. 3. New York, NY: IEEE; 2004, 1497–1500.
74. Lee JA, Verleysen M. Generalization of the L_p-norm for time series and its application to self-organizing maps. In: Cottrell M, ed. *Proceedings. Workshop on Self-Organizing Maps (WSOM)*. Paris: Sorbonne; 2005, 733–740.
75. Lange M, Villmann T. Derivatives of Lp-norms and their approximations. Machine Learning Reports, MLR-03-2013, 2013.
76. Bunte K, Schneider P, Hammer B, Schleif F-M, Villmann T, Biehl M. Limited rank matrix learning, discriminative dimension reduction, and visualization. *Neural Netw* 2012, 26:159–173.
77. Schneider P, Biehl M, Hammer B. Adaptive relevance matrices in learning vector quantization. *Neural Comput* 2009, 21:3532–3561.
78. Schneider P, Bunte K, Stiekema H, Hammer B, Villmann T, Biehl M. Regularization in matrix relevance learning. *IEEE Trans Neural Netw* 2010, 21:831–840.
79. Arlt W, Biehl M, Taylor AE, Hahner S, Libe R, Hughes BA, Schneider P, Smith DJ, Stiekema H, Krone N, et al. Urine steroid metabolomics as a biomarker tool for detecting malignancy in adrenal tumors. *J Clin Endocrinol Metab* 2011, 96:3775–3784.
80. Biehl M, Bunte K, Schneider P. Analysis of flow cytometry data by matrix relevance learning vector quantization. *PLoS One* 2013, 8:e59401.
81. Bunte K, Biehl M, Jonkman MF, Petkov N. Learning effective color features for content based image retrieval in dermatology. *Pattern Recogn* 2011, 44:1892–1902.
82. Denecke A, Wersing H, Steil J, Körner E. Online figure–ground segmentation with adaptive metrics in generalized LVQ. *Neurocomputing* 2009, 72:1470–1482.
83. Backhaus A, Ashok PC, Praveen BB, Dholakia K, Seiffert U. Classifying Scotch Whisky from near-infrared Raman spectra with a radial basis function network with relevance learning. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Verleysen M, ed. Bruges (Belgium), 25–27 April 2012, 411–416.
84. Boareto M, Cesar J, Leite VBP, Caticha N. Supervised variational relevance learning, an analytic geometric feature selection with applications to omic data sets. *IEEE/ACM Trans Comput Biol Bioinform* 2015, 12:705–711.
85. Weinberger KQ, Blitzer J, Saul L. Distance metric learning for large margin nearest neighbor classification. In: Weiss Y, Schölkopf B, Platt J, eds. *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press; 2006, 1473–1480.
86. Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. *J Mach Learn Res* 2009, 10:207–244.

87. Schneider P, Biehl M, Hammer B. Distance learning in discriminative vector quantization. *Neural Comput* 2009, 21:2942–2969.

88. Biehl M. A no-nonsense GMLVQ demo code. Available at: http://www.cs.rug.nl/~biehl/gmlvq. (Accessed December 30, 2015).

89. Biehl M, Bunte K, Schneider P. Relevance and matrix adaptation in learning vector quantization. Available at: http://matlabserver.cs.rug.nl/gmlvqweb/web. (Accessed December 30, 2015).

90. Hammer B, Hofmann D, Schleif F-M, Zhu X. Learning vector quantization for (dis-)similarities. *Neurocomputing* 2014, 131:43–51.

91. Pevsner J. *Bioinformatics and Functional Genomics*. Hoboken, NJ: John Wiley & Sons; 2003.

92. Mokbel B, Paaßen B, Schleif F-M, Hammer B. Metric learning for sequences in relational LVQ. *Neurocomputing* 2015, 169:306–322.

