Patch processing for relational learning vector quantization

Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer

CITEC centre of excellence, Bielefeld University, 33615 Bielefeld - Germany
xzhu@techfak.uni-bielefeld.de

Abstract. Recently, an extension of popular learning vector quantization (LVQ) to general dissimilarity data has been proposed, relational generalized LVQ (RGLVQ) [10, 9]. An intuitive prototype based classification scheme results which can divide data characterized by pairwise dissimilarities into priorly given categories. However, the technique relies on the full dissimilarity matrix and, thus, has squared time complexity and linear space complexity. In this contribution, we propose an intuitive linear time and constant space approximation of RGLVQ by means of patch processing. An efficient heuristic which maintains the good classification accuracy and interpretability of RGLVQ results, as demonstrated in three examples from the biomdical domain.

1 Introduction

Learning vector quantization constitutes a popular supervised classification algorithm which represents classes in terms of prototypical vectors [11]. Since prototypes can directly be inspected in the same way as data points, it opens the way towards intuitive data analysis, unlike many black box alternatives such as feedforward networks or support vector machines. Prototype-based approaches are beneficial if human insight is crucial such as in the medical domain [16, 2].

LVQ itself has been proposed on heuristic grounds and its mathematical investigation is difficult. Several alternatives have been developed which maintain the intuitive adaptation and classification scheme of LVQ but which can be derived from an objective function: important examples are generalized LVQ (GLVQ) which is based on the accumulated hypothesis margin of the classifier [17], or robust soft LVQ, which is based on probabilities in Gaussian mixture models [18]. While resulting in high-quality classification, basic LVQ and its extensions are restricted to Euclidean vector spaces.

In modern application scenarios, data are becoming more and more complex and dedicated dissimilarity measures are often used for their processing. Examples include dynamic time warping for time series, alignment for symbolic strings, graph or tree kernels for complex structures, the compression distance to compare sequences based on an information theoretic ground, and similar. These settings do not allow a vectorial representation of data at all, rather, data are given implicitly in terms of pairwise dissimilarities or relations; we refer to a 'relational data representation' in the following when addressing data sets which are represented implicitly by means of pairwise dissimilarities d_{ij} of data; D denotes the corresponding matrix of dissimilarities.

Several popular unsupervised prototype-based clustering methods have been extended to relational data by means of an implicit embedding of data into

pseudo-Euclidean space, see e.g. [8]. Recently, this technique has been transferred to supervised classification by means of GLVQ, resulting in relational GLVQ (RGLVQ) for general relational data matrices [10, 9]. A very powerful technique results which determines prototypical representatives of given relational data such that priorly given categories are met by the cluster labels as much as possible. The scheme depends on the full dissimilarity matrix, thus it requires squared time and linear space complexity for training given n data samples. This makes the technique infeasible for large data sets.

In this contribution, we propose an approximation scheme which relies on a processing of the data in patches and a subsequent compression of the information by means of the learned prototypes. Assuming fixed patch sizes, a linear time and constant space learning technique results which allows us to deal with large dissimilarity data sets in reasonable time. We will demonstrate in three examples from the biomedical domain, that the resulting heuristic maintains the good classification accuracy of RGLVQ, while considerably speeding up the techniques. Now we first shortly review prototype based classification, generalized learning vector quantization and its relational counterpart. Then we explain the principle of patch processing and we demonstrate its performance.

2 Prototype based classification

Assume vectorial data $\boldsymbol{x}^i \in \mathbb{R}^n, i = 1, \ldots, m$ are given. Prototypes $\boldsymbol{w}^j \in \mathbb{R}^n, j = 1, \ldots, k$ decompose data into receptive fields $R(\boldsymbol{w}^j) = \{\boldsymbol{x}^i \ : \ \forall j' \ d(\boldsymbol{x}^i, \boldsymbol{w}^j) \leq d(\boldsymbol{x}^i, \boldsymbol{w}^{j'})\}$ based on the squared Euclidean distance $d(\boldsymbol{x}^i, \boldsymbol{w}^j) = \|\boldsymbol{x}^i - \boldsymbol{w}^j\|^2$. In a classification task, prototypes are equipped with class labels $c(\boldsymbol{w}^j) \in \{1, \ldots, L\}$. A given data point $\boldsymbol{x}^i$ is mapped to the class of its closest prototype. In a training scenario, training data $\boldsymbol{x}^i$ are labeled with priorly known classes $\boldsymbol{y}^i$ and the goal of a learning algorithm is to determine the prototype positions such that the classification error $E = \sum_{i,j:\boldsymbol{x}^i \in R(\boldsymbol{w}^j)} \delta(y^i, c(\boldsymbol{w}^j))$ is as small as possible where δ refers to the standard Kronecker-function.

Generalized learning vector quantization Since a direct optimization of these costs is hard, generalized learning vector quantization (GLVQ) [17] considers the related cost function

$$E_{GLVQ} = \sum_i \Phi \left(\frac{d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) - d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))}{d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) + d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))} \right)$$

where Φ is a differentiable monotonic function such as the hyperbolic tangent, and $\boldsymbol{w}^+(\boldsymbol{x}^i)$ refers to the prototype closest to $\boldsymbol{x}^i$ with the same label as $\boldsymbol{x}^i$, $\boldsymbol{w}^-(\boldsymbol{x}^i)$ refers to the closest prototype with a different label. This way, for every data point, its contribution to the cost function is small iff the distance to the closest prototype with a correct label is smaller than the distance to a wrongly labeled prototype, resulting in a correct classification of the point. It has been shown in [17] that these costs can be linked to the overall hypothesis margin of an LVQ classifier which directly influences its generalization ability.

A learning algorithm can be derived thereof by means of a stochastic gradient descent. After a random initialization of prototypes, data $\boldsymbol{x}^i$ are presented in

random order and adaptation of the closest correct and wrong prototype takes place by means of the update rules

$$\Delta \boldsymbol{w}^+(\boldsymbol{x}^i) \sim \; -\Phi'(\mu(\boldsymbol{x}^i)) \cdot \mu^+(\boldsymbol{x}^i) \cdot \nabla_{\boldsymbol{w}^+(\boldsymbol{x}^i)} d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i))$$
$$\Delta \boldsymbol{w}^-(\boldsymbol{x}^i) \sim \Phi'(\mu(\boldsymbol{x}^i)) \cdot \mu^-(\boldsymbol{x}^i) \cdot \nabla_{\boldsymbol{w}^-(\boldsymbol{x}^i)} d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))$$

where

$$\mu(\boldsymbol{x}^i) = \frac{d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) - d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))}{d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) + d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))},$$
$$\mu^+(\boldsymbol{x}^i) = \frac{2 \cdot d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))}{(d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) + d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))^2},$$
$$\mu^-(\boldsymbol{x}^i) = \frac{2 \cdot d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)}{(d(\boldsymbol{x}^i, \boldsymbol{w}^+(\boldsymbol{x}^i)) + d(\boldsymbol{x}^i, \boldsymbol{w}^-(\boldsymbol{x}^i))^2}.$$

For the squared Euclidean norm, the derivative yields $\nabla_{\boldsymbol{w}^j} d(\boldsymbol{x}^i, \boldsymbol{w}^j) = -(\boldsymbol{x}^i - \boldsymbol{w}^j)$, leading to Hebbian update rules of the prototypes, i.e. they adapt the closest prototypes towards / away from a given data point depending on the correctness of the classification.

Relational generalized learning vector quantization In the following, we assume that data $\boldsymbol{x}^i$ are not explicitly given as vectors, rather pairwise dissimilarities $d_{i,j} = d(\boldsymbol{x}^i, \boldsymbol{x}^j)$ are available. We assume symmetry $d_{ij} = d_{ji}$ and zero diagonal $d_{ii} = 0$. However, we do not require that d refers to a Euclidean data space, i.e. D does not need to be embeddable in Euclidean space, nor does it need to fulfill the conditions of a metric. The following observation constitutes the key to transfer GLVQ to this setting [7, 8]: any such matrix D gives rise to a so-called pseudo-Euclidean embedding, i.e. a real-vector space equipped with a symmetric, but not necessarily positive semidefinite form where vectorial representations $\boldsymbol{x}^i$ give rise to the dissimilarity matrix D when computed based on the bilinear form. Further, assuming prototypes are represented as linear combinations of data points

$$\boldsymbol{w}^j = \sum_i \alpha_{ji} \boldsymbol{x}^i \text{ with } \sum_i \alpha_{ji} = 1,$$

dissimilarities can be computed by means of the formula

$$d(\boldsymbol{x}^i, \boldsymbol{w}^j) = \|\boldsymbol{x}^i - \boldsymbol{w}^j\|^2 = [D \cdot \alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D \alpha_j$$

where $\alpha_j = (\alpha_{j1}, \ldots, \alpha_{jn})$ refers to the vector of coefficients describing $\boldsymbol{w}^j$.

This observation gives rise to an extension of GLVQ to relational data, relational GLVQ (RGLVQ), without actually computing the underlying pseudo-Euclidean embedding of data. We represent prototypes implicitly by means of coefficients α_j and adapt the cost function accordingly:

$$E_{\text{RGLVQ}} = \sum_i \Phi \left(\frac{[D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+ - [D\alpha^-]_i + \frac{1}{2} \cdot (\alpha^-)^t D \alpha^-}{[D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D \alpha^+ + [D\alpha^-]_i - \frac{1}{2} \cdot (\alpha^-)^t D \alpha^-} \right),$$

where as before the closest correct and wrong prototype are referred to, indicated by the superscript + and −, respectively. A stochastic gradient descent directly leads to adaptation rules for the coefficients α^+ and α^-: component l of these vectors is adapted by the rules

$$\Delta\alpha_l^+ \sim -\Phi'(\mu(\boldsymbol{x}^i)) \cdot \mu^+(\boldsymbol{x}^i) \cdot \frac{\partial \left([D\alpha^+]_i - \frac{1}{2} \cdot (\alpha^+)^t D\alpha^+\right)}{\partial \alpha_l^+}$$

$$\Delta\alpha_l^- \sim -\Phi'(\mu(\boldsymbol{x}^i)) \cdot \mu^-(\boldsymbol{x}^i) \cdot \frac{\partial \left([D\alpha^-]_i - \frac{1}{2} \cdot (\alpha^-)^t D\alpha^-\right)}{\partial \alpha_l^-}$$

where $\mu(\boldsymbol{x}^i)$, $\mu^+(\boldsymbol{x}^i)$, and $\mu^-(\boldsymbol{x}^i)$ are as above. The partial derivative yields

$$\frac{\partial [D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^t D\alpha_j}{\partial \alpha_{jl}} = d_{il} - \sum_{l'} d_{l'l}\alpha_{jl'}$$

After every adaptation step, normalization takes place to guarantee $\sum_i \alpha_{ji} = 1$. This way, a learning algorithm which adapts prototypes in a supervised manner similar to GLVQ is given for general dissimilarity data, whereby prototypes are implicitly embedded in pseudo-Euclidean space. We initialize α_{ij} with small random values such that the sum is one. It is possible to take class information into account by setting $\alpha_{ji} := 0$ if $c(\boldsymbol{w}^j) \neq \boldsymbol{y}^i$.

The resulting classifier represents clusters in terms of prototypes for general dissimilarity data. Although these prototypes correspond to vector positions in pseudo-Euclidean space, they can usually not be inspected directly because the pseudo-Euclidean embedding is not computed directly. For inspection, we use an approximation of the prototypes: we substitute a prototype by its K nearest exemplars as measured by the given dissimilarity.

Out-of-sample extension is as follows: given a novel data point $\boldsymbol{x}$ characterized by its pairwise dissimilarities $D(\boldsymbol{x})$ to the data used for training, the dissimilarity to the prototypes is given by $d(\boldsymbol{x}, \boldsymbol{w}^j) = D(\boldsymbol{x})^t \cdot \alpha_j - \frac{1}{2} \cdot \alpha_j^t D\alpha_j$. For an approximation of prototypes by exemplars, obviously, only the dissimilarities to these exemplars have to be computed, i.e. a very sparse classifier results.

3 Patch relational generalized learning vector quantization

RGLVQ relies on the full dissimilarity matrix D and represents prototypes implicitly by means of coefficients α_{ji} referring of the contribution of data point $\boldsymbol{x}^i$ to prototype $\boldsymbol{w}^j$. Thus, the algorithm has squared time complexity and linear space complexity. Patch processing has been proposed as an alternative approximation scheme. It offers a powerful linear time and limited memory approximation for streaming data sets with a direct access to the dissimilarities, e.g. by means of a computation scheme for $d(\boldsymbol{x}^i, \boldsymbol{x}^j)$ [1]. In the article [8], it has been used to speed up relational prototype based clustering. The resulting technique is linear time and constant space. Here we extend it to RGLVQ.

The basic idea of patch processing is: A fixed size of the patches m is chosen, and data are separated into patches. Then the patches of data are processed

consecutively using RGLVQ. Given a dissimilarity matrix, the patch p corresponds to the values of the matrix describing the pairwise dissimilarities along the diagonal: $d(\boldsymbol{x}^i, \boldsymbol{x}^j)$ where $i, j \in \{p \cdot m + 1, \ldots, (p+1) \cdot m\}$. In addition to this part, all previous patches are represented in compressed form by means of the prototypes found before. This way, the patch does not only represent data $\{\boldsymbol{x}^{pm+1} \ldots \boldsymbol{x}^{(p+1)m}\}$ but all data $\{\boldsymbol{x}^1 \ldots, \boldsymbol{x}^{(p+1)m}\}$ either explicitly or implicitly by taking into account the already extracted prototypes. To apply RGLVQ, the dissimilarities of data and these prototypes need to be available. In patch processing, these are retrieved on the fly.

Note that it is not clear how to compute these dissimilarities efficiently if prototypes are of the general form $\boldsymbol{w}^j = \sum_i \alpha_{ji} \boldsymbol{x}^i$: this representation would, eventually, refer to the full dissimilarity matrix. Therefore, after processing a patch, we approximate a prototype by its K-approximation for fixed K. The K-approximation $\boldsymbol{w}^j_K$ of a prototype $\boldsymbol{w}^j$ based on a given set of points E corresponds to the closest K data points in E: $\boldsymbol{w}^j_K := \{\boldsymbol{x}^i \in E \,|\, d(\boldsymbol{x}^i, \boldsymbol{w}^j) \leq d(\boldsymbol{x}^{i'}, \boldsymbol{w}^j)$ for all but K indices $i\}$. This way, the dissimilarity matrix considered in step p corresponds to a fixed size $m + kK$ matrix, k being the number of prototypes.

Exemplars/K-approximated prototypes representing the previous clustering results represent a large set of data. Thus, it is vital to weight their relevance correspondingly. In the patch algorithm, this problem is solved by assigning a multiplicity to these prototypes which corresponds to the size of its receptive field divided by K. This means, we assume that the corresponding prototypes are contained in the data set not only once but multiple times. Note that RGLVQ can easily be extended to deal with sets where data points are equipped with multiplicities. For a points $\boldsymbol{x}^i$ with multiplicity m_i its contribution to the costs is simply multiplied by m_i. Hence the corresponding step width of a gradient descent algorithm is simply multiplied with m_i. The resulting algorithm, Patch RGLVQ, is depicted in Algorithm 1

Algorithm 1 Principled algorithm for patch clustering

init: $E := \emptyset$; ▷ exemplars/K-approximated prototypes
  $m_i := 1$ for $\boldsymbol{x}_i \in E$; ▷ multiplicities
  $p := 1$; ▷ patch number
repeat
  $P_{m,m} := \{d(\boldsymbol{x}_i, \boldsymbol{x}_j)\} \mid i, j \in \{p \cdot m + 1, \ldots, (p+1) \cdot m\}\}$; ▷ patch size m
  $P_{m,|E|} := \{d(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid p \cdot m < i \leq (p+1) \cdot m, \boldsymbol{x}_j \in E\}$;
  ▷ dissimilarities of patch and exemplars
  $P_{|E|,|E|} := \{d(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid \boldsymbol{x}_i, \boldsymbol{x}_j \in E\}$; ▷ dissimilarities of exemplars
  $P := \begin{pmatrix} P_{m,m} & P_{m,|E|} \\ P^t_{m,|E|} & P_{|E|,|E|} \end{pmatrix}$; ▷ full matrix for loop
  $m_i :=$ multiplicities for $\boldsymbol{x}_i \in E$; ▷ multiple points
  $m_i := 1$ for other x_i; ▷ standard points
  perform Patch RGLVQ with multiplicities for P and m_i;
  approximate prototypes by K closest exemplars;
    $E :=$ set of exemplars obtained this way; ▷ new exemplars
    $m_i :=$ size of receptive field/K counted with multiplicities for $\boldsymbol{x}_i \in E$;
  p:=p+1; ▷ next patch
until all dissimilarities are considered

4 Experiments

We evaluate the algorithm for three benchmark data sets where data are characterized by pairwise dissimilarities:

1. The Copenhagen chromosomes data set constitutes a benchmark from cytogenetics [13]. 4,200 human chromosomes from 22 classes (the autosomal chromosomes) are represented by grey-valued images. These are transferred to strings measuring the thickness of their silhouettes. These strings are compared using edit distance with insertion/deletion costs 4.5 [15].
2. The vibrio data set consists of 1,100 samples of vibrio bacteria populations characterized by mass spectra. The spectra contain approx. 42,000 mass positions. The full data set consists of 49 classes of vibrio-sub-species. The mass spectra are preprocessed with a standard workflow using the BioTyper software [14]. As usual, mass spectra display strong functional characteristics due to the dependency of subsequent masses, such that problem adapted similarities such as described in [3, 14] are beneficial. In our case, similarities are calculated using a specific alignment measure as provided by the BioTyper software[14].
3. The *SwissProt* data set consists of 10,988 samples of protein sequences in 32 classes taken as a subset from the full database [4]. The considered subset of the SwissProt database refers to the release 37 mimicking the setting as proposed in [12]. The full database consists of 77,977 protein sequences varying between 30 to more than 1000 amino acids depending on the sequence. The 32 most common classes such as Globin, Cytochrome a, Cytochrome b, Tubulin, Protein kinase st, etc. provided by the Prosite labeling [5] where taken leading to 10,988 sequences. Due to this choice, an associated classification problem maps the sequences to their corresponding prosite labels. These sequences are compared using Smith-Waterman which computes a local alignment of sequences [6]. Popular alternatives could rely on global alignment as provided by Needleman-Wunsch, or linear time heuristics such as BLAST or FASTA [6]. This database is the standard source for identifying and analyzing protein sequences such that an automated classification and processing technique would be very desirable.

These three data sets constitute typical examples of non-Euclidean data which occur in biomedical domains. The dissimilarity measures are inherently non-Euclidean and cannot be embedded isometrically in a Euclidean vector space.

We compare the results of RLVQ for the full dissimilarity matrix and patch processing. For comparison, we report the result of a Nyström approximation of the full dissimilarity matrix. This approximation constitutes a standard low rank approximation of a similarity or dissimilarity matrix, which has been introduced in the context of kernel methods in [19]. For RGLVQ, it has been proposed in the contribution [10]. Like patch processing, it leads to a linear time approximation technique. The setting is as follows in the experiments:

- *Evaluation*: We evaluate the result by means of the classification accuracy obtained in a ten-fold cross-validation with 10 repeats (Chromosomes, Vibrio), or a 2-fold cross-validation with 10 repeats (SwissProt).
- *RGLVQ:* RGLVQ is initialized randomly, and training takes place for 5 epochs. We use 49 (Vibrio), 63 (Chromosomes), and 64 (SwissProt) prototypes evenly distributed among the classes.

	RGLVQ	Nyström-RGLVQ using 10%	Patch - RGLVQ $K=1$	$K=3$	$K=5$
Vibrio	1.0	0.992	0.999	1	1
Chromosomes	0.927	0.782	0.867	0.840	0.828
SwissProt	0.823	0.834	0.833	0.824	0.822
speed-up factor	1	7.6	26.2	20	13.2

Table 1. Results on three data sets: RGLVQ, RGLVQ with Nyström, and Patch RGLVQ are evaluated in a repeated cross-validation. The classification accuracy, and the speedup factor according to the CPU time are reported.

- *Patch processing*: For patch processing, ten patches are chosen. The value K for the K-approximation for patch processing is taken in $\{1, 3, 5\}$.
- *Nyström approximation*: A fraction of 10% of the data is used.
- *Implementation:* For all data sets, we use a 12 Intel(R) Xeon X5690 machine with 3.47GHz processors and 48 GB DDR3 1333MHz memory. All experiments are implemented in Matlab. [1]

For Vibrio and SwissProt, the classification accuracy obtained with a linear time approximation is the same as for full RGLVQ. For Chromosomes, it decreases by 6% using patch approximation as compared to almost 25% for the Nystöm approximation. Interestingly, all patch approximations already yield a high quality when approximating the prototypes by its closest exemplar ($K = 1$). This approximation has the side effect that classes can directly be inspected in terms of this representative exemplar, i.e. interpretable models result. We measure the speed-up of the technique for the SwissProt data set which deals with close to $11,000^2$ entries. Original RGLVQ takes 24481 seconds CPU time (i.e. almost seven hours), which can be accelerated by a factor 26 to 15 minutes using patch processing – the Nyström approximation requires considerably more time.

5 Conclusions

In this contribution, we proposed a linear time constant space approximation scheme for supervised prototype based classification by means of relational LVQ. Apart from a considerable speed-up, training does no longer rely on the full dissimilarity matrix; rather a linear subpart is required depending on the chosen patch scheme and the prototypes. Since the computation of the full matrix often constitutes a major bottleneck for complex dissimilarity measures such as alignment, this fact offers even greater application potential. The method has been demonstrated in three examples from the biomedical domain, among those a large portion of the polular SwissProt data set for proteins.

Acknowledgement Financial support from the Cluster of Excellence 277 Cognitive Interaction Technology funded in the framework of the German Excellence Initiative is gratefully acknowledged.

[1] The Matlab code of the proposed algorithms can be obtained from Xibin Zhu (xzhu@techfak.uni-bielefeld.de) on request.

References

1. N. Alex, A. Hasenfuss, and B. Hammer. Patch clustering for massive data sets. *Neurocomputing*, 72(7-9):1455–1469, 2009.
2. W. Arlt et al., Urine steroid metabolomics as a biomarker tool for detecting malignancy in adrenal tumors. *Journal of Clinical Endocrinology and Metabolism*, 96(12):3775–3784, 2011.
3. S. B. Barbuddhe, T. Maier, G. Schwarz, M. Kostrzewa, H. Hof, E. Domann, T. Chakraborty, and T. Hain. Rapid identification and typing of listeria species by matrix-assisted laser desorption ionization-time of flight mass spectrometry. *Applied and Environmental Microbiology*, 74(17):5402–5407, 2008.
4. B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledge base and its supplement trembl in 2003. *Nucleic Acids Research*, 31:365–370.
5. E. Gasteiger, A. Gattiker, C. Hoogland, I. Ivanyi, R. Appel, and A. Bairoch. Expasy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res*, 31(3784-3788), 2003.
6. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
7. B. Hammer. *Learning with Recurrent Neural Networks*, volume 254 of *Lecture Notes in Control and Information Sciences*. Springer, 2000.
8. B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229–2284, 2010.
9. B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. Prototype-based classification of dissimilarity data. In J. Gama, E. Bradley, and J. Hollmén, editors, *IDA*, volume 7014 of *Lecture Notes in Computer Science*, pages 185–197. Springer, 2011.
10. B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. White box classification of dissimilarity data. In *Hybrid Artificial Intelligent Systems*, accepted, 2012.
11. T. Kohonen, editor. *Self-Organizing Maps*. Springer-Verlag New York, Inc., 3rd edition, 2001.
12. T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15:945–952, 2002.
13. C. Lundsteen, J-Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin g-banded human metaphase chromosomes. *Clinical Genetics*, 18:355–370, 1980.
14. T. Maier, S. Klebel, U. Renner, and M. Kostrzewa. Fast and reliable maldi-tof ms–based microorganism identification. *Nature Methods*, (3), 2006.
15. M. Neuhaus and H. Bunke. Edit distance based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.
16. F.-M. Schleif, B. Hammer, M. Kostrzewa, and T. Villmann. Exploration of mass-spectrometric data in clinical proteomics using learning vector quantization methods. *Briefings in Bioinformatics*, 9(2):129–143, 2008.
17. P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.
18. S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2002.
19. C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.