# Approximation techniques for clustering dissimilarity data

Xibin Zhu, Andrej Gisbrecht, Frank-Michael Schleif and Barbara Hammer *

{xzhu|agisbrecht|fschleif|bhammer@techfak.uni-bielefeld.de}

Bielefeld University - CITEC Centre of Excellence
D-33594 Bielefeld - Germany

November 2, 2011

**Abstract**

Recently, diverse high quality prototype-based clustering techniques have been developed which can directly deal with data sets given by general pairwise dissimilarities rather than standard Euclidean vectors. Examples include affinity propagation, relational neural gas, or relational generative topographic mapping. Corresponding to the size of the dissimilarity matrix, these techniques scale quadratically with the size of the training set, such that training becomes prohibitive for large data volumes. In this contribution, we investigate two different linear time approximation techniques, patch processing and the Nyström approximation. We apply these approximations to several representative clustering techniques for dissimilarities, where possible, and compare the results for diverse data sets.

## 1 Introduction

The amount of digital data doubles roughly every 20 months. Hence automatic tools to deal with large data sets become indispensable for humans to extract relevant information from the data. In this context, clustering constitutes one of the standard techniques to structure and compress large data sets. Algorithms which represent clusters in an intuitive form such as prototype-based techniques offer the possibility to directly inspect the results. Additional functionality such as e.g. topographic mapping as enabled by self-organizing algorithms can provide further inside into the data structure. Because of these facts, prototype based

*This work has been supported by the DFG under grant number HA2719/4-1 and by the Cluster of Excellence 277 Cognitive Interaction Technology funded in the framework of the German Excellence Initiative.

clustering and extensions towards topographic mapping have lost none of its attraction for users from diverse application areas [19].

Not only the size of modern data sets, but also its complexity increases rapidly in modern application areas. Improved sensor technology, for example, leads to very high dimensional measurements corresponding to a very detailed resolution of the available information. At the same time, dedicated data formats such as XML files, network data, graph structures and the like become more and more common. Classical prototype based classification such as k-means clustering, neural gas, or the self-organizing map usually deals with Euclidean vectors only. Hence these algorithms are no longer suited in these settings. While the Euclidean distance yields to almost meaningless values for high dimensionality, a lossless vectorial representation is not even possible for data structures such as sequences, trees, or graph structures.

This fact has led to a variety of extensions of prototype-based techniques to deal with more complex data formats, see e.g. [3]. One prominent interface is offered by a general similarity or dissimilarity matrix: only pairwise similarities or dissimilarities of data have to be defined based on which learning takes place. Various dissimilarity measures are available for dedicated data formats: for example, alignment for sequences [14], functional norms for functional data [26], divergences for probability distributions [27], graph and tree kernels [16], or the compression distance for general symbolic sequences [7]. Hence a formulation in terms of dissimilarities extends the applicability of prototype based techniques to a large variety of modern application areas. Since data are characterized by pairwise relations rather than Euclidean vectors, we refer to 'relational data' in the following.

There exist different principled ways to transfer prototype based clustering towards dissimilarity data: Kernel methods extend standard techniques towards more general data by means of kernelization, see e.g. [29, 6]. This has the drawback that a valid kernel has to be present, i.e. data have to be inherently Euclidean corresponding to a positive semidefinite Gram matrix. Techniques such as proposed in [17] are based on the so-called dual cost function associated to quantization error. It allows an elegant solution using techniques of statistical physics. However, no explicit prototypes are available in this setting. As an alternative, exemplar based techniques restrict prototype positions to data points, such that standard cost functions as the quantization error are still defined for general dissimilarities. Optimization of these costs, however, becomes hard due to the discrete space of feasible solutions. Median clustering determines optima by extensive search [20, 8]. It often leads to only suboptimal solutions due to the search in a very restricted space [8]. Recently, a very promising alternative optimization method has been proposed [10]: Affinity propagation (AP) reformulates the quantization error such that it can be formalized as a factor graph, for which powerful optimization techniques such as the max-sum algorithm are readily available. By relying on log-likelihood values, the algorithm inherently deals with a continuous relaxation of the discrete optimization problem and usually arrives at very good optima of the cost function. We will consider AP as one very promising clustering technique for exemplar based data representation

for general similarities in the following.

AP has the drawback that additional functionality such as neighborhood preservation of the clusters is not available. Recently, Euclidean prototype based topographic mapping as offered by neural gas [23] and the generative topographic mapping [4] have been extended to general dissimilarities. A key technique is an implicit embedding of data in pseudo-Euclidean space [15, 13]. This way, a continuous update of prototypes becomes possible leading to robust solutions of the respective cost function in pseudo-Euclidean space. Using a simple algebraic equation, the explicit computation of the embedding becomes superfluous – update rules which depend on the given dissimilarities only can be derived. We will consider two instantiations of this technique, relational neural gas (RNG) and relational generative topographic mapping (RGTM). These techniques constitute two important schemes to infer a topographic mapping for general dissimilarity data.

All methods, AP, RNG, and RGTM, have the drawback that they rely on the full dissimilarity matrix which is quadratic with respect to the number of data. Hence the techniques have quadratic complexity and they become infeasible for large data sets. Diverse approximations to get around a squared complexity in similar settings are available in the literature: Kernel approaches can be accelerated to linear techniques by means of the Nyström approximation [28] which approximates the full Gram matrix by a low rank approximation. By integrating this approximation into the learning algorithms, an overall linear complexity results. We will show that the Nyström approximation can be extended to dissimilarity data and an integration into RNG and RGTM is possible. A linear time approximation results provided a fixed approximation quality of the matrix. We will show that, depending on the nature of the dissimilarity matrix, reasonable results can be obtained this way.

The Nyström technique has two drawbacks: it requires a representative set of examples for the low rank approximation, such that it cannot be used for online settings where data display a clear trend. Further, in order to arrive at linear techniques, the approximation has to be integrated into the learning algorithm. Hence this method does not constitute an option for clustering techniques where the entries of the dissimilarity matrix are used in a distributed way such as AP.

Patch processing has been proposed as an alternative approximation scheme. It offers a powerful linear time and limited memory approximation for streaming data sets [1]. In the article [15], it has been used to speed up RNG. The resulting technique, patch RNG (PRNG) is linear time. It requires a direct access to the dissimilarities. In this contribution, we transfer this technique to AP and RGTM, resulting in patch AP (PAP) and patch RGTM (PRGTM). We show that, depending on the given setting, good approximation quality can be achieved. Patch processing can even deal with streaming data which display a clear trend, unlike the Nyström approximation.

Now we first introduce the basic prototype based clustering techniques for general dissimilarity data. Then we introduce the basic idea behind the Nyström technique and patch processing and we show how these techniques can be applied to relational clustering. Finally, we compare the techniques using three bench-

| Algorithm | Acronym |
|---|---|
| Affinity Propagation | AP |
| Patch Affinity Propagation | PAP |
| *Neural Gas* | *NG* |
| Relational Neural Gas | RNG |
| Patch Relational Neural Gar | PRNG |
| Nyström Relational Neural Gas | RNG (Ny) |
| *Generative Topographic Mapping* | *GTM* |
| Relational Generative Topographic Mapping | RGTM |
| Patch Relational Generative Topographic Mapping | PRGTM |
| Nyström Relational Generative Topographic Mapping | RGTM (Ny) |

Table 1: Different clustering algorithms and approximations introduced and tested in this contribution; algorithms which are suited for vectorial data only are set italic.

marks from bioinformatics: image data in the context of cytogenetics, mass spectra characterizing bacteria, and a part of the classical SwissProt database of protein sequences. In all cases, we compare the behavior of the techniques for data which are directly accessible form versus a presentation as streaming data. The tested algorithms, its approximations, and the corresponding acronyms are summarized in Tab. 1 for convenience.

# 2 Prototype based clustering for dissimilarity data

We focus on prototype based clustering methods which represent clusters in terms of prototypical representatives. In the standard Euclidean setting, data $\vec{v}_i \in \mathbb{R}^n$ are represented by $K$ prototypes $\vec{w}_j \in \mathbb{R}^n$. The receptive field $R_j$ of a prototype $\vec{w}_j$ consists of all data points $\vec{v}_i$ which are closest to $\vec{w}_j$ as measured by the Euclidean distance $d(\vec{v}_i, \vec{w}_j) = \|\vec{v}_i - \vec{w}_j\|$, breaking ties deterministically. That means,

$$R_j := \{\vec{v}_i \in \mathbb{R}^n \mid d(\vec{v}_i, \vec{w}_j)^2 \leq d(\vec{v}_i, \vec{w}_k)^2 \; \forall k \neq j\} \, .$$

In this setting, the goal of clustering can be formalized as minimizing the quantization error

$$E_{\mathrm{qe}} := \sum_{ij:\vec{v}_i \in R_j} d(\vec{v}_i, \vec{w}_j)^2 \tag{1}$$

to obtain prototypes which are as representative for their receptive field as possible. There exist different classical methods which achieve this goal: a direct optimization by means of a gradient descent as present in online vector quantization, or more advanced methods which take a neighorhood structure into account or which rely on a probabilistic interpretation of the model [19, 10, 4].

The latter techniques often yield much more stable results. Further, the possibly provide additional information such as the ability to visualize the prototypes such as in the generative topographic mapping or a neighborhood structure of the clusters such as in neural gas. We will consider three typical clustering techniques in this context: (I) the generative topographic mapping (GTM) which constitutes a generative statistical model. It models data by means of a constraint mixture of Gaussians induced by a mapping from a low-dimensional latent space. In latent space visualization is possible. (II) the neural gas (NG) which models data by means of representative prototypes which represent data in relation to the rank of its distance. This way a very robust algorithm is obtained which is widely independent from scaling issues. (III) affinity propagation (AP) which reformulates the quantization error as a likelihood function. This can be decomposed as factor graph for which the max-sum algorithm can be used [10].

We shortly describe these three models in the following. Thereby, we refer to distances such as the Euclidean distance, or similarities such as the Euclidean dot product as required. Similarities and dissimilarities can be transferred into each other using classical double centering (see e.g. [25]).

**Generative Topographic Mapping**

In GTM, lattice points $\vec{u}_j$ on a regular lattice in a low dimensional latent space are given. These are mapped to prototypes

$$\vec{w}_j = \Phi(\vec{u_j}) \cdot W \tag{2}$$

by means of a generalized linear regression model. In principle, the base functions $\Phi$ could be chosen as nonlinear functions such that their linear combination has sufficient flexibility to map the low dimensional lattice to appropriate positions. At the same time, the base functions control the degree of topology preservation by the stiffness of (2). This is usually obtained by picking only a small number of base functions. In practice, the base functions $\Phi$ are often chosen as equally spaced Gaussian functions. $W$ denotes the weight parameters of the mapping which are determined during learning.

This mapping induces a constrained mixture of Gaussians in the data space in the following way. Every prototype $\vec{w}_j$ induces a Gaussian which variance is determined by the parameter $\beta$

$$p(\vec{v}|\vec{w_j}, \beta) = (\beta/(2\pi))^{n/2} \exp(-\beta/2 \cdot d(\vec{v}, \vec{w}_j)^2)\,. \tag{3}$$

These Gaussians are combined in a mixture model with equal prior. Training optimizes the data log likelihood. This way the parameters $W$ and $\beta$ are determined. It is possible to derive an expectation maximization (EM) algorithm as detailed in [4]. It in turn optimizes the responsibilities

$$R_{ij}(W, \beta) = p(\vec{v}_i|\vec{w}_j, \beta) / \sum_{j'} p(\vec{v}_i|\vec{w}_{j'}, \beta)\,, \tag{4}$$

and the parameters $W$ and $\beta$ by solving the linear equations

$$\mathbf{\Phi}^t \mathbf{G}_{\text{old}} \mathbf{\Phi} W_{\mathbf{new}}^t = \mathbf{\Phi}^t \mathbf{R}_{\text{old}} \mathbf{V}\,, \tag{5}$$

$$\frac{1}{\beta_{\text{new}}} = \frac{1}{Nn} \cdot \sum_{ij} R_{ij}(W_{\text{old}}, \beta_{\text{old}}) d(\vec{v}_i, \Phi(u_j) W_{\text{new}})\,. \tag{6}$$

$N$ denotes the number of data points, $\mathbf{G}$ is the diagonal matrix with entries $G_{ii} = \sum_j R_{ij}(W, \beta)$, $\mathbf{R}$ is the matrix of responsibilities, $\mathbf{\Phi}$ is the matrix of base functions evaluated at all lattice points, and $\mathbf{V}$ is the matrix of data points.

### Neural Gas

NG extends the quantization error to incorporate neighborhood cooperation:

$$E_{\text{NG}} := \sum_{ij} h_\sigma(k_{ij}) d(\vec{v}_i, \vec{w}_j)^2\,, \tag{7}$$

where $h_\sigma(t) = \exp(-t/\sigma)$ exponentially scales the neighborhood range. $k_{ij}$ denotes the rank of prototype $\vec{w}_j$ with respect to $\vec{v}_i$, i.e. the number of prototypes $\vec{w}_k$ with $k \neq j$ which are closer to $\vec{v}_i$ as measured by the Euclidean distance $d$. Classical NG optimizes this objective by means of a stochastic gradient descent. The neighborhood range $\sigma$ is annealed during training such that, in the limit, the standard quantization error is approximated [23]. There exists a faster batch optimization scheme as introduced in [8] which in turn optimizes prototype locations and assignments similar to an EM scheme:

$$\text{determine } k_{ij} \text{ based on } d(\vec{v}_i, \vec{w}_j)^2\,, \tag{8}$$

$$\text{set } \vec{w}_j := \sum_i h_\sigma(k_{ij}) \vec{v}_i / \sum_i h_\sigma(k_{ij})\,. \tag{9}$$

### Affinity Propagation

AP constitutes an exemplar based clustering method, i.e. prototype locations are restricted to data points $\vec{w}_j \in \{\vec{v}_i \,|\, i = 1, \ldots, N\}$. Further, it deals with similarities $s(\vec{v}_i, \vec{w}_j)$ rather than dissimilarities such as the Euclidean dot product, for example. Obviously, the quantization error can be formulated accordingly.

Since prototypes are located at discrete positions, the quantization error can no longer be optimized by means of gradient techniques. Therefore, the quantization error is first rephrased as

$$E_{\text{qe-ap}} := \sum_i s(\vec{v}_i, \vec{w}_I(i)) + \sum_i \delta_i(I)\,. \tag{10}$$

Note that there is no longer a fixed number of clusters $K$ given. Rather, cluster assignments are defined by means of a function $I : \{1, \ldots, N\} \to \{1, \ldots, N\}$. Every data point picks a prototype by means of this function. Since prototypes

are exemplars contained in the data set itself, this can be written as $\vec{v}_{I(i)}$ for data point $\vec{v}_i$. $\delta_i(I)$ punishes invalid assignments, i.e.

$$\delta_i(I) = \begin{cases} -\infty & \text{if } \exists j \; I(j) = i, I(i) \neq i \,, \\ 0 & \text{otherwise.} \end{cases}$$

During training, these assignments have to be adapted; basically the data points have to negotiate to arrive at a valid assignment function which optimizes the cost function. The number of clusters $K$ is no longer specified a priori.

A trivial valid solution of the cost function would be given by the identity $I(i) = i$, i.e. every data point forms an exemplar. To avoid this trivial solution, costs have to be introduced as soon as a data point becomes an exemplar. This can be achieved by setting the self-similarities $s(\vec{v}_i, \vec{v}_i)$ to non-zero values, indicating the costs of data point $\vec{v}_i$ becoming an exemplar. In the limit, for high costs, only one cluster will be found by AP. In between, different numbers of clusters can be reached. Typically, there are two different strategies to set the self-similarities appropriately: either they are set to a reasonable fixed value such as the median of the given similarities. Alternatively, binary search can take place until a desired number of clusters is reached. We will use the latter strategy in this article.

To solve this novel optimization problem (10), the function $E_{\mathrm{qe-ap}}$ is modeled as factor graph. This can be optimized by means of the max-sum algorithm. In turn, responsibilities

$$r_{ij} := s(\vec{v}_i, \vec{v}_j) - \max_{j' \neq j} \{a_{ij'} + s(\vec{v}_i, \vec{v}_{j'})\}$$

and availabilities

$$a_{ij} := \min\{0, r_{jj} + \sum_{i' \neq i,j} \max\{0, r_{i'j}\}\} \,,$$

$$a_{ii} := \sum_{i' \neq i} \max\{0, r_{i'i}\}$$

are determined leading to assignments $I(i) = \mathrm{argmax}_j(a_{ij} + r_{ij})$ as detailed in [10]. Since the factor graph is cyclic, there is no guarantee to obtain the global optimum or even convergenvce. For this reason, small random values are added to the dissimilarities in every run to avoid cycles. This way, usually, convergence to a fixed point is observed.

Note that AP does not rely on Euclidean similarities. Rather the cost function (10) and the corresponding optimization is valid for every general similarity measure $s$ due to the restriction of prototypes to exemplars. In [10], this fact has been demonstrated by various applications involving microarray data and text, for example.

## 2.1 Extensions to (Dis-)Similarities

Unlike AP, NG and GTM are defined for the Euclidean setting only: they rely on vector operations in the data space $\mathbb{R}^n$. AP, on the contrary, restricts

prototype positions to exemplars. Thus it is defined for every general similarity measure. This property is very interesting since, often, data are not given in vectorial form. Rather, pairwise similarities $s_{ij}$ or dissimilarities $d_{ij}$ of data points numbered $i$ and $j$ are available only, such as biological sequences and alignment distances.

Recently, extensions of NG and GTM have been proposed which can take into account a general dissimilarity matrix [15]. The key observation is given by the fact that a general dissimilarity matrix corresponds to vectorial data embedded in so-called pseudo-Euclidean space. The latter refers to a vector space equipped with a bilinear form which induces the given dissimilarities. Unlike standard Euclidean space, the form can be indefinite. In this space, standard vectorial operations are possible. An implicit calculation of the corresponding updates avoids an explicit computation of the embedding.

**Relational Neural Gas**

Relational neural gas (RNG) as introduced in [15] assumes that a symmetric dissimilarity matrix $D$ with entries $d_{ij}$, describing pairwise dissimilarities of data, is available. As shown e.g. in [25], there exists a so-called pseudo-Euclidean embedding of a given set of points characterized by pairwise symmetric dissimilarities. That means, there exists a vector space with a symmetric bilinear form $\langle \cdot, \cdot \rangle$ and vectors $\vec{v}_i$ such that $d_{ij} = d(\vec{v}_i, \vec{v}_j)^2 = \langle \vec{v}_i - \vec{v}_j, \vec{v}_i - \vec{v}_j \rangle$. Without loss of generality, we can restrict prototypes to the span of the data points in this space:

$$\vec{w}_j = \sum_l \alpha_{jl} \vec{v}_l \text{ with } \sum_l \alpha_{jl} = 1 \,. \tag{11}$$

Under this constraint, dissimilarities can be computed by means of the formula

$$d(\vec{v}_i, \vec{w}_j)^2 = [D^t \alpha_j]_i - \frac{1}{2} D \cdot \alpha_j^t \tag{12}$$

where $[\cdot]_i$ refers to component $i$ of the vector. This allows a direct transfer of batch NG to general dissimilarities:

$$\text{determine } k_{ij} \text{ based on } d(\vec{v}_i, \vec{w}_j)^2 \,, \tag{13}$$

$$\text{set } \alpha_{jl} := h_\sigma(k_{lj}) / \sum_l h_\sigma(k_{lj}) \,. \tag{14}$$

This algorithm can be interpreted as neural gas in pseudo-Euclidean space for a given symmetric dissimilarity matrix $D$. It performs the updates implicitly without referring to the vectorial notation $\vec{v}_i$ in pseudo-Euclidean space. This way the computational complexity is reduced to $\mathcal{O}(N^2)$ instead of $\mathcal{O}(N^3)$ for a full embedding. If the bilinear form is indefinite, convergence is not guaranteed, albeit it is usually observed in practice [15].

**Relational Generative Topographic Mapping**

Relational GTM (RGTM) relies on the same principle as RNG: prototypes are restricted to linear combinations of data points

$$\vec{w}_j = \sum_l \alpha_{jl} \vec{v}_l \text{ with } \sum_l \alpha_{jl} = 1\,. \tag{15}$$

This way, vectorial operations become possible in pseudo-Euclidean space, referring only implicitly to the corresponding embedding [13]. Again, distances are computed by means of (12). Since prototypes are represented indirectly by means of weighting factors, the generalized regression model maps from the latent space to the space of coefficients $\alpha_k = \Phi(\vec{u}_k) \cdot W$ where the restriction $\sum_l [\Phi(\vec{u}_k) \cdot W]_l = 1$ is enforced. Using Lagrange optimization, it can be seen that this restriction is automatically fulfilled for the generic optima obtained by means of EM optimization. Hence, the EM scheme of GTM can directly be transferred to RGTM. Distances are computed by means of (12). The data matrix $\mathbf{V}$ is the identity in the space of coefficients. Typically, GTM is initialized based on the first two eigenvectors of the data. This can directly be extended to dissimilarity data by referring to MDS which corresponds to a PCA in the space of similarities. Detailed formulas for the resulting updates can be found in [13].

## 2.2 Efficient Approximations

These algorithms allow to generalize classical clustering algorithms to general similarities or dissimilarities. However, a principled drawback occurs: since the methods rely on the full similarity or dissimilarity matrix, respectively, their effort is quadratic with respect to the number of data. Even more severely, the full quadratic dissimilarity matrix has to be available to apply these methods. This can lead to a considerable effort for complex dissimilarities such as e.g. alignment of sequences or graph structures.

Because of these facts, the techniques cannot be used for large data sets. In the following, we discuss two principled ways to approximate prototype based clustering techniques for dissimilarities, and we show how these methods can be integrated into the algorithms RNG, RGTM, and AP.

**Nyström Approximation**

The Nyström approximation as presented in [28] substitutes a given Gram matrix $S$ by a low rank approximation such that linear techniques result. As discussed in [12], this principle can be generalized to dissimilarities. By the Mercer theorem, one can find an expansion of the form $s(\vec{w}, \vec{v}) = \sum_{i=1}^{\infty} \lambda_i \Psi_i(\vec{w}) \Psi_i(\vec{v})$ for a given kernel $s$ with eigenfunctions $\Psi_i$ and eigenvalues $\lambda_i$. These values are solutions of the equation $\int s(\vec{w}, \vec{v}) \Psi_i(\vec{v}) p(\vec{v}) d\vec{v} = \lambda_i \Psi_i(\vec{w})$. In the Nyström

approximation, this integral is approximated by sampling:

$$\frac{1}{m} \cdot \sum_k s(\vec{w}, \vec{v}_k)\Psi_i(\vec{v}_k) \approx \lambda_i \Psi_i(\vec{w})\,. \tag{16}$$

Using the matrix eigenproblem $S^{(m)}\mathbf{U}^{(m)} = \mathbf{U}^{(m)}\Lambda^{(m)}$ of the $m \times m$ Gram matrix $S^{(m)}$, we obtain

$$\lambda_i \approx \lambda_i^{(m)}/m\,, \tag{17}$$

$$\Phi_i(\vec{w}) \approx \sqrt{m} \cdot (s(\vec{v}_1, \vec{w}), \ldots, s(\vec{v}_m, \vec{w}))\vec{u}_i^{(m)}\,, \tag{18}$$

where $\vec{u}_i^{(m)}$ is the $i$th column of $\mathbf{U}^{(m)}$. This equation correspond to $m$ rows and respective columns of a given $N \times N$ Gram matrix. The corresponding rows are denoted by $S_{m,N}$ and columns by $S_{N,m}$. Hence we obtain $\tilde{S} = \sum_{i=1}^{m} 1/\lambda_i^{(m)} \cdot S_{N,m} \cdot \vec{u}_i^{(m)}(\vec{u}_i^{(m)})^t S_{m,N}$, where $\lambda_i^{(m)}$ and $\vec{u}_i^{(m)}$ correspond to the $m \times m$ eigenproblem. Hence

$$\tilde{S} = S_{N,m}S_{m,m}^{-1}S_{m,N} \tag{19}$$

where $S_{m,m}^{-1}$ denotes the Moore-Penrose pseudo inverse. This matrix $\tilde{S}$ offers a low rank approximation of $S$.

We will see that the corresponding matrix $S$ is used in algorithms such as RNG and RGTM in the form $S^t\vec{x}_i$ where $\vec{x}_i$ is an $N$-dimensional vector. Then, performing multiplication from right to left, the complexity $\mathcal{O}(m^3 + Nm + m^2 + mN)$ results when computuing $\tilde{s}^t\vec{x}_i$. Hence the computation is of complexity $\mathcal{O}(m^3N)$ instead of $\mathcal{O}(N^2)$ for the original matrix.

Similarly, dissimilarities $D$ can be approximated if $D$ is symmetric. Since $D$ is symmetric, it can be diagonalized. Hence it can be interpreted as operator $d(\vec{v}, \vec{w})^2 = \sum_i \lambda_i \Psi_i(\vec{v}), \Psi_i(\vec{w})$. Thus, the same mathematical treatment as before is possible, the only difference being that eigenvalues are allowed to be negative.

**Nyström relational topographic mapping and Nyström relational neural gas**

For RGTM, this yields an approximation of (12)

$$\begin{aligned} d(\vec{v_i}, \vec{w_j})^2 \approx [D_{N,m}(D_{m,m}^{-1}(D_{m,N} \cdot \alpha_j))]_i \\ -\frac{1}{2} \cdot (\alpha_j^t D_{N,m}) \cdot (D_{m,m}^{-1}(D_{m,N} \cdot \alpha_j)) \end{aligned} \tag{20}$$

which is $\mathcal{O}(m^3N)$. Thereby, initialization of RGTM is done based on a corresponding landmark MDS. Similarly, the Nyström approximation can be integrated into RNG approximating the distance computation(12) in the same way. In both cases, by evaluating the matrix multiplications consecutively, a linear time algorithm results provided the sample size $m$ is fixed. Note that the same

approximation does not lead to a reduction of the computational complexity for AP since the similarity values are distributed in the algorithmic computations rather than treated in matrix form.

The Nyström approximation is exact if the number of samples $m$ is chosen according to the rank of $D$. If a subsample is chosen, bounds on the quality of the Nyström approximation can be derived as presented e.g. in [30]. Note that the quality of the approximation depends on the rank of the approximation as compared to the original matrix. Thus, it is vital that a representative subsample is chosen. We will see in experiments, that the size $m$ can often be chosen as a small set to arrive at good results. The accuracy degrades severely, however, if streaming data are dealt with for which the chosen subset is not representative, as we will show in experiments.

If we assume a fixed size $m$ and a fixed complexity of the algorithm depending on the rank of the approximation (i.e. a fixed number of epochs etc.), a linear time approximation results. Note that the part of the dissimilarity matrix which is required for the Nyström approximation is fixed as soon as the representative subsample is chosen. Thus, the required dissimilarities can be precomputed before applying the algorithm.

## Patch Processing

Patch processing was first introduced to k-means in [9]. It is a simple and efficient strategy for clustering with restricted buffer where data are processed consecutively in patches of predefined size. It has been proposed in [1] in the context of the application of NG to streaming data, and interestingly, it even gives good results this setting. The principled algorithm is depicted in Tab. 2. A fixed size of the patches $m$ is chosen. Then patches of data are processed consecutively. Given a similarity or dissimilarity matrix, the patch $p$ corresponds to the values of the matrix describing the pairwise similarities/dissimilarities along the diagonal: $d(v_i, v_j)$ where $i, j \in \{p \cdot m + 1, \ldots, (p+1) \cdot m\}$. In addition to this patch, all previous patches are represented in compressed form by means of the prototypes resulting from clustering the previous patches. For this purpose, the similarities or dissimilarities of data and these prototypes need to be available. In patch processing, these are computed or retrieved from the dissimilarity matrix on the fly.

Note that it is not clear how to compute these dissimilarities efficiently if prototypes are of a general form, i.e. they correspond to arbitrary vectors in pseudo-Euclidean space as for RNG and RGTM. For an implicit representation of prototypes by means of a linear combination such as (11), the dissimilarity computation would depend on the full dissimilarity matrix. To avoid this problem, we assume that prototypes are approximated by a small number of exemplars. Under this assumption, the pairwise distances of these exemplars as well as the distances of these exemplars and the new patch can be retrieved from the dissimilarity matrix in constant time on the fly. This extended dissimilarity matrix, representing patches and representative exemplars, serves as the input for the clustering step of the next patch.

**init:** $E := \emptyset$; /* exemplars */
$m_i := 1$ for $\vec{v}_i \in E$; /* multiplicities */
$p := 1$; /* patch number */

**repeat:** /********** **dissimilarity matrix for the loop** *********/
$P_{m,m} := \{d(\vec{v}_i, \vec{v}_j)\} \mid i, j \in \{p \cdot m + 1, \ldots, (p+1) \cdot m\}\}$;
/* patch size $m$ */
$P_{m,|E|} := \{d(\vec{v}_i, \vec{v}_j) \mid p \cdot m < i \leq (p+1) \cdot m, \vec{v}_j \in E\}$;
/* dissimilarities patch and exemplars */
$P_{|E|,|E|} := \{d(\vec{v}_i, \vec{v}_j) \mid \vec{v}_i, \vec{v}_j \in E\}$; /* dissimilarities exemplars */
$P := \begin{pmatrix} P_{m,m} & P_{m,|E|} \\ P^t_{m,|E|} & P_{|E|,|E|} \end{pmatrix}$; /* full matrix for loop */
(*1) change $p_{ii}$ for all $i$, if necessary; /* diagonal entries */

/********** **multiplicities** $m_i$ **********/
$m_i :=$ multiplicities stored in $E$ for $\vec{v}_i \in E$; /* multiple points */
$m_i := 1$ for other $v_i$; /* standard points */

/********** **clustering** **********/
(*2) perform patch clustering with multiplicities for $P$ and $m_i$;

/********** $k$ **approximation** **********/
(*3) approximate prototypes by $k$ closest exemplars, if necessary;

/********** **new exemplars** **********/
$E :=$ set of exemplars obtained this way;
$m_i :=$ size of receptive field counted with multiplicities for $\vec{v}_i \in E$;

/********** **next patch** **********/
p:=p+1;

Table 2: Principled algorithm for patch clustering

Exemplars which represent 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 all exemplars which represent clusters of points, which corresponds to the size of its receptive field. This means, we assume that the corresponding exemplars are contained in the data set not only once but multiple times.

Note that patch processing requires constant space, provided the patch size $m$ and the approximation quality of prototypes by $k$ exemplars is fixed. Further, it requires linear time, provided $m$ is fixed and the corresponding metaparameters of the clustering algorithm such as the required number of epochs depend on the size of $m$ only. Unlike the Nyström approximation, it is not clear a priori which linear part of the dissimilarity matrix is used for training, since

the relevant exemplars which represent clusters become available during patch processing only. Thus, we need a way to compute or retrieve the relevant entries of the dissimilarity matrix on the fly. This is easily possible if the dissimilarity is characterized by an algorithmic function such as e.g. Smith-Waterman to align biological sequences by means of a global alignment, or alternatives such as local alignment by Needleman-Wunsch or approximations such as FASTA or BLAST. [14]. Because all relevant information is kept either directly or in compressed form, patch processing can also deal with streaming data, as we will see in experiments.

## Patch relational neural gas and patch relational topographic mapping

Patch relational neural gas (PRNG) has been proposed in [15]. A similar extension of RGTM to patch processing, patch RGTM (PRGTM), is possible. The algorithmic steps which have to be clarified are marked with numbers (*1) to (*3) in the algorithm in Tab. 2.

For PRNG and PRGTM, the diagonal (*1) is simply set to zero. It is easy to extend RNG to multiplicities in step (*2). Assume data point $\vec{v}_i$ has multiplicity $m_i$. Then we exchange the update (14) by

$$\alpha_{jl} := m_l \cdot h_\sigma(k_{lj}) / \sum_l m_l \cdot h_\sigma(k_{lj}) \,.$$

RGTM can be extended to multiplicities by exchanging the update equations (5) and (6) of original RGTM such that they are valid for multiple data points: In (5), the matrices $\mathbf{G}$ and $\mathbf{R}$ which relate to responsibilities and sums of responsibilities, respectively, weight these responsibilities $R_{ij}(W, \beta)$ with $m_i$. In the update (6) for $\beta$, the summand according to data point $\vec{v}_i$ is weighted with $m_i$, and $N$ is the number of data counted with multiplicities $\sum_i m_i$. Similarly, the MDS initialization can be extended to multiple data points.

To account for (*3), we need to approximate the prototypes computed by RNG and RGTM, since these entities refer to all given data points. As proposed in [15] we use a simple approximation of the prototypes by means of a fixed number of exemplars taken from the data processed in the current patch. We simply substitute every prototype by the $k$ closest exemplars of the current patch as measured by the given dissimilarity. These exemplars are counted with multiplicities according to the size of their receptive fields.

## Patch Affinity Propagation

Since AP constitutes an exemplar based clustering method, the transfer of the meta algorithm is immediate: we just repeatedly apply AP to a given patch and the exemplars as provided by the previous patch, counted with multiplicities. Thus, the step (*3) of the algorithm shown in Tab. 2 is trivial since prototypes already correspond to exemplars.

To apply patch processing to AP obtaining patch AP (PAP), we need to extend AP such that it can deal with multiple data points in step (*2). One simple way to do so would exist in a simulation of the update equations provided by standard AP. We can include points according to their multiplicities to the data set. Since these points are exactly identical, their responsibilities and availabilities can be shared. Thus, we only have to compute the corresponding updates of responsibilities and availabilities once.

Unfortunately, this naive procedure is not possible: AP consists of the max-sum algorithm applied to a cyclic factor graph, hence convergence is not guaranteed. In particular, cycles occur if symmetries are present in the problem formulation. Symmetries can be induced, for example, by exactly identical similarities. For this reason, slight noise is added to the similarities in the original code for AP (see http://www.psi.toronto.edu/index.php?q=affinity propagation). If multiple points are present, as in our setting, however, symmetries necessarily occur. Since the updates of these identical points are shared, there is no simple way to cure this problem: the corresponding responsibilities and availabilities are exactly identical since there is only one variable representing the values.

Therefore, we use a different approach to extend AP to multiple points. Assume data point $v_i$ is contained in the data set with multiplicity $m_i$. Then the quantization error is given as follows

$$E_{\mathrm{qe}} = \sum_{ij:\vec{v}_i \in R(j)} m_i \cdot s(\vec{v}_i, \vec{w}_j)\,. \tag{21}$$

Obviously, we obtain the same costs if every point is contained only once in the data set, but similarities are given by the values $\tilde{s}(\vec{v}_i, \vec{v}_j) = m_i \cdot s(\vec{v}_i, \vec{v}_j)$.

Therefore, we can treat the problem to cluster the points $\vec{v}_i$ with multiplicities $m_i$ as a standard problem for standard AP with simple points $\vec{v}_i$ where the similarities are given as $\tilde{s}(\vec{v}_i, \vec{v}_j)$. This way, the usual convergence of AP is observed using small random values to avoid symmetries. The update for the responsibilities becomes

$$r_{ik} := m_i \cdot s_{ik} - \max_{k' \neq k} \left\{ a_{ik'} + m_i \cdot s_{ik'} \right\}, \tag{22}$$

where $s_{ik} = s(\vec{v}_i, \vec{v}_k)$ refers to the original similarities of data points. The update of the availabilities is not changed.

The initialization of the diagonal terms, step (*1) of the algorithm in Tab. 2, should also be adapted accordingly, putting a bias towards points with large multiplicities. We achieve this by a division of the preferences along the diagonal by the respective multiplicities $m_i$.

# 3 Experiments

We evaluate and compare the following algorithms:

- Affinity Propagation (AP) and the approximation Patch Affinity Propagation (PAP),
- Relational Neural Gas (RNG) and the two approximations Patch Relational Neural Gas (PRNG) and Nyström Relational Neural Gas (RNG (Ny)),
- Relational Generative Topographic Mapping (RGTM) and the two approximations Patch RGTM (PRGTM) and Nyström RGTM (RGTM (Ny)).

We test the algorithm for three data sets stemming from biomedical applications.

- The *Copenhagen Chromosomes* data constitute a benchmark from cytogenetics [21]. 4,200 human chromosomes from 21 classes (the autosomal chromosomes) are represented by grey-valued images. These are transferred to strings measuring the thickness of their silhouettes. An example pattern representing a chromosome has the form

  $$11332444222333323322223332233233322226662223311 11 .$$

  The string indicates the thickness of the gray levels of the image. These strings can directly be compared using the edit distance based on the differences of the numbers and insertion/deletion costs 4.5 [24]. Data are labeled by the number of the chromosome, hence an associated classification problem is to label the data according to the chromosome type.

- The *vibrio* data set consists of 1,100 samples of vibrio bacteria populations characterized by mass spectra.[1] The spectra encounter 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 [22]. As usual, mass spectra display strong functional characteristics due to the dependency of subsequent masses, such that problem adapted similarities such as described in [2, 22] are beneficial. In our case, similarities are calculated using a dedicated similarity measure as provided by the BioTyper software [22]. Roughly speaking, spectra are represented by peak lists. These are aligned taking the local shapes into account leading to an overall score value. Details of the alignment can be found in [2]. A typical spectrum is depicted in Fig. 1. For every spectrum, its corresponding vibrio sub-species is known, such that an associated classification task maps the data to this subspecies.

- The *SwissProt* data set consists of 10,988 samples of protein sequences in 32 classes taken as a subset from the SwissProt database [5]. The considered subset of the SwissProt database refers to the release 37 mimicking the setting as proposed in [20]. The full database consists of 77,977

[1] We would like to thank Dr. Markus Kostrzewa, Dr. Karl-Otto Kräuter, Dr. Stephan Klebel and Dr. Thomas Maier, all Bruker Daltonik GmbH, Germany, for providing the Vibrio data and support regarding the analysis of the mass spectra with the BioTyper environment.

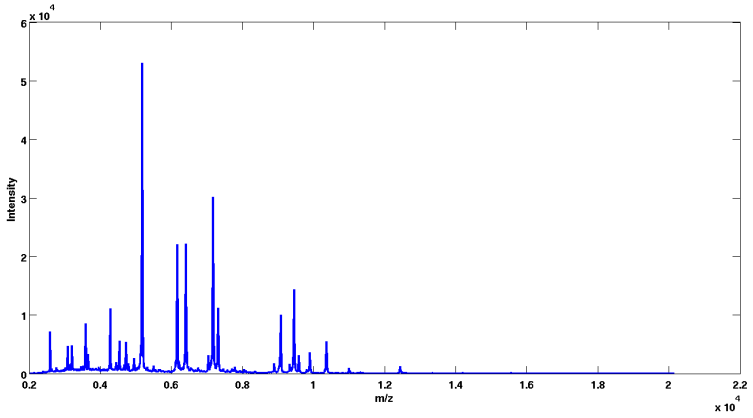


Figure 1: Typical mass spectrum representing a vibrio bacterium, the graph shows the intensity for a band of mass/charge ratios.

protein sequences. A typical protein sequence consists of a string of the form $MSKAKEGDYGSIKKVSGPVVV\ldots$ where the letters refer to the amino acids, and the length of the full sequences varies 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 [11] 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 [14]. Popular alternatives could rely on global alignment as provided by Needleman-Wunsch, or linear time heuristics such as BLAST or FASTA [14]. This database is the standard source for identifying and analyzing protein sequences such that an automated classification and processing technique would be very desirable.

We evaluate the data sets using three different evaluation measures:

- The *quantization error* (1) which measures in how far the prototypes can represent the given data as measured by the averaged dissimilarity of prototypes to points in their receptive field. All methods optimize a cost function which can be related to the quantization error: GTM optimizes the data log-likelihood which, neglecting topological constraints due to the restricted form of the topographic mapping, would boil down to a mixture of Gaussians. Similarly, for small neighborhood $\sigma \to 0$, the cost function of NG directly resembles the quantization error. AP optimizes the quantization error under the restriction that prototypes are exemplars.

  Therefore, the evaluation of the quantization error where the terms $d(\vec{v}_i, \vec{w}_j)$

are taken as dissimilarities, allows to directly evaluate the affect of the approximation techniques on an underlying cost function.

- The *dual quantization error* is given by

$$E_{\mathrm{dualQE}} = \sum_{i=1}^{n} \frac{1}{4 \cdot |R_j|} \sum_{i,i' \,:\, \vec{v}_i \in R_j, \vec{v}_{i'} \in R_j} d(\vec{v}_i, \vec{v}_{i'})^2 \,.$$

  It has been shown in [15], that the quantization error and its dual coincide if prototypes are located at centers of receptive fields in pseudo-Euclidean space $\vec{w}_j = \sum_{i \,:\, \vec{v}_i \in R_j} \vec{v}_i / |R_j|$. Hence the quantization error and its dual are identical for RNG. For RGTM, AP, or approximations of RNG, prototypes are no longer located at the center of gravity due to restrictions of the prototypes or approximations of the dissimilarities, respectively. In such cases, the quantization error can be large, albeit the decomposition of data into receptive fields is hardly affected. In these cases, the dual quantization error allows us to compare the quality of the decomposition of data points into clusters rather than the specific prototype locations.

- Often, clustering or, more precisely, a prototype-based representation of data serves as a first step towards a classification of data in practical applications. If label information is available for the given training samples, prototypes can easily be assigned a label by means of a majority vote in its receptive field. In such cases, it is possible to evaluate the classification error of a given clustering. We test the suitability of approximation techniques for this procedure by referring to the *classification error* obtained by posterior labeling. For classification tasks, typically, the generalization ability is relevant. Therefore we also report the results of a cross-validation for this setting. Obviously, however, there is no reason to assume that the class boundaries of priorly known classes coincide with cluster boundaries. Therefore, this evaluation technique can judge the underlying clustering only to a limited degree.

- For the settings, we also report the *CPU time* in seconds taken for one run on the full data sets on a 24 Intel(R) Xeon X5690 machine with 3.47GHz processors and 48 GB DDR3 1333MHz memory. Thereby, all experiments are implemented in Matlab. [2]

The parameter choices are as follows:

- *Repeats*: Per default, we average every result over ten repeats, reporting the mean value and standard deviation. Due to time issues, less repeats are taken in the following settings: RGTM (only one repeat for the cross-validation), RGTM (Ny with 10%) (only 4 repeats for the cross-validation).

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

- *Cross-validation*: The classification error is evaluated on the full data set, and in a ten-fold repeated cross-validation.
- *Patch sizes*: For patch processing, ten patches are chosen, i.e. every patch is of the size given by the number of data points / 10.
- *k-approximation:* The value $k$ for the $k$-approximation for patch processing is taken in $\{1, 3, 5\}$.
- *Number of clusters:* The number of clusters $K$ is roughly chosen according to the priorly known number of classes: For Chromosomes, we use 60 clusters for AP and RNG and its approximations, and $20 \times 20$ lattice points for RGTM and its approximations, to account for topological constraints of the latter. For the Vibrio data set, we use 50 clusters for AP and RNG, and $20 \times 20$ lattice points for RGTM. For SwissProt, we use 250 Prototypes for AP and RNG, and a $40 \times 40$ lattice for RGTM.
- *Nyström approximation*: A fraction of 1%, 2%, and 10% of the data is used to approximate the full matrix.
- *RGTM*: For RGTM and its approximations, we use the default initialization with MDS (landmark MDS for the Nyström approximation). The number of base functions $F$ is picked according to the data set: We use $10 \times 10$ base functions with bandwidth 1 for the chromosomes and vibrio data and bandwidth 0.2 for Swissprot. 50 epochs are used for EM training.
- *RNG*: For RNG and its approximations, we use an exponential annealing schedule for the neighborhood range starting from $K/2$. 100 epochs are used for training.
- *AP*: For AP, self-similarities are set by binary search starting from the median in repeated runs such that a fixed number of exemplars/prototypes as specified above is obtained. We accept partial deviations thereof, an exact fit of a given number of clusters often requiring additional trials. Usually, 1-8 repeats are necessary to obtain the correct number of exemplars. The necessary number of repetitions is included in the CPU time measurement. Adaptation is done until convergence is observed.

We test the algorithms in two settings: On the one hand, we present the data in epochs subject to random permutations (standard setting). This way, the approximation techniques can access representative data for the approximation. In addition, we present the data in a fixed permutation referred to as *streaming data*; for the latter, data are sorted according to priorly given class labels and presented to the algorithm in this order - this resembles the fact that, for large data sets, it is often not feasible to assume a truly i.i.d. distribution of data; rather data can be addressed as they are stored in (probably distributed) memory; they are often ordered according to specific criteria such as e.g. the class labels. Therefore, it would be advantageous if algorithms could directly deal with the data as stored in the memory and the ordering would

not have much influence on the outcome. For the Nyström approximation and patch processing, respectively, specific ordering might have an influence since the approximation is based on the ordering of the data: landmarks are taken from the first examples for the Nyström approximation, and, similarly, the first patches are restricted to the first data points for patch processing. We will test the feasibility of the approximation in such settings, referred to as 'streaming data'.

Since AP, unlike GTM and NG, requires similarities, we use standard double centering to transform similarities to dissimilarities and vice versa.

## Results for the Chromosomes data

The results of the algorithms on the Chromosomes data set are displayed in Tab. 3 when referring to the full data set, and in Tab. 4 when referring to an evaluation of the classification accuracy in a repeated cross-validation. For the latter setting, we also report the results obtained for streaming data.

We can observe that the results are uniformly best for RGTM. As expected, the quality of all evaluation measures drops down if approximations are used. The overall trend, however, is diverse. This can be traced back to the following phenomena: on the one hand, some approximation results display a uniform decrease in accuracy such as the Nyström approximation with 10% of the data. This can be explained by a bad quality of the resulting dissimilarity matrix. We will specify this issue in a separate paragraph shortly. Hence, in this setting, the Nyström approximation fails.

On the other hand, we can observe a drop down of the quantization error by more than 20% as compared to the best result, but the dual quantization error and the classification error do not follow the trend. This holds in particular for AP and for patch approximations. The reason behind this effect is that prototypes are located at restricted positions in such settings because they are represented by exemplars (a small number of $k$ exemplars for patch processing). This causes the quantization error to increase, albeit the resulting cluster separation is still acceptable as measured by the dual quantization error. For this reason, we suggest that the dual quantization error is more suited to compare the results.

When focussing on the dual quantization error, we can observe that the approximation techniques yield results which are mostly comparable to the original techniques for AP and RNG. More precisely, the increase is less than 3% for PAP as compared to AP, less than 9% for PRNG and RNG (Ny) as compared to RNG except for Nyström approximation using 10%. For RGTM, the increase is less than 8% for the Nyström approximation using 1% of the data, and for patch processing with $k \in \{3, 5\}$. Thus, it seems that the approximation techniques are well suited if appropriate parameters concerning the approximation are taken.

This overall picture is confirmed if the generalization ability of a classifier by posterior labeling is addressed as displayed in Tab. 4. Interestingly, the sensitivity of the approximation techniques to the ordering of the data is quite

| ***Chromosomes*** | accuracy | QE | dualQE | CPUtime |
|---|---|---|---|---|
| **AP** | 0.899(0.001) | 70902.250 (0.000) | 47112.667 (9.406) | 380(73) |
| **PAP** | 0.864(0.001) | 76739.950 (928.209) | 48518.834 (412.783) | 2752(11) |
| **RNG** | 0.902(0.009) | 45751.008 (81.324) | 45751.008 (81.324) | 148(3) |
| **RNG (Ny 10%)** | 0.759(0.132) | 57620.918 (11936.192) | 51801.754 (5479.342) | 123(7) |
| **RNG (Ny 2%)** | 0.810(0.098) | 52415.133 (6287.455) | 49776.813 (4657.611) | 49(2) |
| **RNG (Ny 1%)** | 0.884(0.006) | 46580.623 (232.367) | 46216.928 (104.868) | 48(4) |
| **PRNG (k=1)** | 0.858(0.007) | 76761.575 (520.301) | 49118.806 (318.913) | 63(1) |
| **PRNG (k=3)** | 0.879(0.008) | 61582.019 (432.681) | 47875.753 (248.223) | 83(2) |
| **PRNG (k=5)** | 0.884(0.008) | 58155.238 (383.595) | 47432.292 (148.811) | 99(4) |
| **RGTM** | 0.941(0) | 41407.211 (0.000) | 37737.074 (0.000) | 3262(49) |
| **RGTM (Ny 10%)** | 0.409(0.072) | 428041.085 (1030835.396) | 66159.509 (3041.827) | 512(113) |
| **RGTM (Ny 2%)** | 0.689(0.190) | 60274.937 (1449.945) | 50668.785 (7935.304) | 424(34) |
| **RGTM (Ny 1%)** | 0.885(0.012) | 44763.112 (1929.780) | 41277.147 (419.516) | 416(7) |
| **PRGTM (k=1)** | 0.856(0.010) | 61176.709 (676.005) | 43915.486 (723.462) | 359(18) |
| **PRGTM (k=3)** | 0.869(0.011) | 51346.674 (197.432) | 40622.258 (443.058) | 620(25) |
| **PRGTM (k=5)** | 0.886(0.006) | 48464.344 (308.555) | 39935.687 (361.247) | 1089(63) |

Table 3: Results on the Chromosome data set: The different methods are trained for the full data set and the classification accuracy (accuracy), the quantization error (QE), and the dual quantization error (dual QE) are reported. In addition, we include the CPU time. The standard deviation is given in parentheses.

diverse. While the classification accuracy of patch processing is still more than 84% for streaming data, the Nyström approximation cannot deal with such settings. The letter can be accounted for by the fact that the approximate matrix has a small rank if only a subspace of the full data space is isolated by means of a specific ordering.

| ***Chromosomes*** | accuracy | Streaming data |
|---|---|---|
| **AP** | 0.895 (0.006) | |
| **PAP** | 0.838 (0.005) | 0.783 (0.022) |
| **RNG** | 0.902 (0.003) | |
| **RNG (Ny 10%)** | 0.497 (0.057) | 0.101 (0.053) |
| **RNG (Ny 2%)** | 0.700(0.039) | 0.088(0.034) |
| **RNG (Ny 1%)** | 0.871 (0.009) | 0.088 (0.031) |
| **PRNG (k=1)** | 0.854 (0.006) | 0.621 (0.021) |
| **PRNG (k=3)** | 0.877 (0.004) | 0.747 (0.016) |
| **PRNG (k=5)** | 0.880 (0.006) | 0.763 (0.012) |
| **RGTM** | 0.881 (0.007) | |
| **RGTM (Ny 10%)** | 0.516 (0.066) | 0.431 (0.082) |
| **RGTM (Ny 2%)** | 0.638 (0.081) | 0.582 (0.107) |
| **RGTM (Ny 1%)** | 0.878 (0.027) | 0.756 (0.099) |
| **PRGTM (k=1)** | 0.840 (0.005) | 0.843 (0.007) |
| **PRGTM (k=3)** | 0.858 (0.007) | 0.857 (0.006) |
| **PRGTM (k=5)** | 0.873 (0.005) | 0.871 (0.005) |

Table 4: Results on the Chromosomes data set in a repeated ten-fold cross-validation evaluated by posterior labeling, the standard deviation is shown in parenthesis

Due to the comparably small size of the data set, the speed-up of the approximation techniques is only partially observable. As displayed in Tab. 3, a speed-up up to 3 can be gained for RNG, and more than 7 for RGTM. Surprisingly, PAP needs longer run time than AP. This can be explained by the fact that we also include necessary restarts to obtain a correct number of clusters by binary search.

In summary, RGTM and a patch approximation with sufficient $k$ seems best suited in this setting to arrive at a good cluster structure or classification. The Nyström approximation seems problematic due to very different results for different parameter choices in this setting.

The Chromosomes data set has also been addressed in the approach [18]. In this approach, k-nearest neighbor classification is based on the edit distance to compare the data. Results report a classification error of only about 5%, alternative classifiers such as Hidden Markov Models or feedforward networks account for an error of about 9%. Our experimental setting is slightly different. Still, the classification error is in the same order of magnitude for RNG, albeit the latter does not take into account the class labels during training.

## Results for the Vibrio data

For the Vibrio data set, the overall picture is similar as summarized in Tab. 5 and Tab. 6. RGTM displays the best quantization error and dual quantization error,

| ***Vibrio*** | accuracy | QE | dualQE | CPUtime |
|---|---|---|---|---|
| **AP** | 0.999(0.000) | 594.480(0.000) | 303.755(0.000) | 136(1) |
| **PAP** | 0.992(0.013) | 604.993(2.546) | 304.702(1.218) | 35(3) |
| **RNG** | 0.796(0.025) | 327.899(4.332) | 327.889(4.332) | 18(1) |
| **RNG (Ny 10%)** | 0.895(0.024) | 316.945(2.464) | 316.904(2.497) | 13(1) |
| **RNG (Ny 2%)** | 0.841(0.015) | 333.296(3.892) | 330.847(3.589) | 3(0) |
| **RNG (Ny 1%)** | 0.764(0.029) | 358.924(8.236) | 344.848(5.423) | 3(0) |
| **PRNG (k=1)** | 0.732(0.034) | 661.101(9.277) | 343.007(7.624) | 16(0) |
| **PRNG (k=3)** | 0.864(0.035) | 422.829(8.383) | 319.353(4.389) | 23(0) |
| **PRNG (k=5)** | 0.859(0.019) | 385.893(5.979) | 321.858(2.501) | 34(7) |
| **RGTM** | 0.960(0.000) | 305.637(0.000) | 285.757(0.000) | 111(9) |
| **RGTM (Ny 10%)** | 0.925(0.022) | 308.804(6.096) | 301.493(5.427) | 48(2) |
| **RGTM (Ny 2%)** | 0.946(0.019) | 309.885(5.501) | 293.813(5.332) | 38(1) |
| **RGTM (Ny 1%)** | 0.806(0.025) | 357.398(4.175) | 296.326(8.282) | 35(2) |
| **PRGTM (k=1)** | 0.702(0.103) | 451.467(6.351) | 328.479(20.430) | 166(7) |
| **PRGTM (k=3)** | 0.876(0.027) | 375.971(2.109) | 293.201(2.491) | 234(11) |
| **PRGTM (k=5)** | 0.897(0.030) | 341.489(4.233) | 291.986(3.756) | 309(14) |

Table 5: Results on the Vibrio data set when trained for the full data set. The standard deviation is given in parentheses.

while AP leads to the best classification accuracy. This is an indicator that, in this setting, the priorly known class structures are only partially mirrored in the cluster structure found in the data set. As for Chromosomes, exemplar based techniques lead to an increased quantization error, while its dual is competitive to the results of RNG and RGTM. In this setting, the approximation techniques are uniformly good, leading to an increase of the dual quantization error of less than 6% in all settings but patch RGTM with $k = 1$. In Tab. 6, the classification accuracy obtained for streaming data is displayed in comparison to a random permutation. Again, the Nyström approximation cannot deal with this setting in the same way as patch processing, the latter yielding an accuracy of more than 80% for all but the 1-approximation.

Again, due to the size of the data set, the speed-up of the approximation techniques is only partially visible. It accounts for a factor close to 4 for PAP as compared to AP, up to 6 for approximations of RNG, and more than 3 for RGTM approximations. Generally, the Nyström approximation seems more efficient with respect to the time complexity as compared to patch processing due to the smaller overhead of the technique.

In summary, AP and PAP are preferred if the prior classification task is addressed this way. If the dual quantization error is in the focus, all methods and their approximations seem universally suited. For comparison, the result of an SVM classification (standard Matlab implementation in the bioinformatics toolbox with default parameters, the SVM directly works on the given Gram matrix) leads to 100% classification accuracy in a cross-validation. As reported

| ***Vibrio*** | accuracy | streaming data |
|---|---|---|
| **AP** | 0.999 (0.000) | |
| **PAP** | 0.999 (0.000) | 0.993 (0.004) |
| **RNG** | 0.798(0.012) | |
| **RNG (Ny 10%)** | 0.885 (0.010) | 0.392 (0.097) |
| **RNG (Ny 2%)** | 0.823 (0.012) | 0.179 (0.117) |
| **RNG (Ny 1%)** | 0.715 (0.014) | 0.052 (0.034) |
| **PRNG (k=1)** | 0.729 (0.011) | 0.658 (0.022) |
| **PRNG (k=3)** | 0.842 (0.012) | 0.819 (0.013) |
| **PRNG (k=5)** | 0.858 (0.005) | 0.820 (0.014) |
| **RGTM** | 0.947 (0.005) | |
| **RGTM (Ny 10%)** | 0.939 (0.007) | 0.848 (0.023) |
| **RGTM (Ny 2%)** | 0.781 (0.021) | 0.697 (0.040) |
| **RGTM (Ny 1%)** | 0.547 (0.031) | 0.462 (0.058) |
| **PRGTM (k=1)** | 0.692 (0.029) | 0.645 (0.034) |
| **PRGTM (k=3)** | 0.867 (0.013) | 0.852 (0.017) |
| **PRGTM (k=5)** | 0.903 (0.011) | 0.898 (0.011) |

Table 6: Results on the Vibrio data set when evaluated in a repeated ten-fold cross-validation for randomly permuted data sets (accuracy) and trained for the streaming setting, i.e. data sets presented according to the class labels, the standard deviation is shown in parenthesis

above, AP yields almost the same accuracy albeit not taking class labels into account during training.

## Results for the SwissProt data

For the SwissProt data set, results are reported in Tab. 7. The dual quantization error is best for AP, but the dual quantization error for PAP and RNG and its approximation uniformly less than 10% away. Thereby, no big difference can be observed for the different approximation technique. The RGTM seems universally less suited for the data set, probably because of too strong topological constraints which prevent a good representation of the data in this case. Albeit the dual quantization error is competitive, the classification accuracy varies in the diverse cases. It displays more than 90% accuracy for AP and its approximation, but less than 70% for some of the approximations of RNG. Thus, the link between the priorly given classes and the cluster structure is not clear.

Because of the size of the data set, the speed-up of the approximation techniques can clearly be observed in all cases. It accounts for a factor 2.5 for PAP as compared to AP, up to 6 for approximations of RNG, and up to 10 for approximations of RGTM. The speed-up depends on the parameter choice, smaller values of $k$ for patch processing and a smaller percentage of data points for the Nyström approximation accounting for considerably reduced CPU time.

| ***SwissProt*** | accuracy | crossVal | dualQE | CPUtime |
|---|---|---|---|---|
| **AP** | 0.931(0.001) | 0.925 (0.001) | 3370.656(0.000) | 14162(37) |
| **PAP** | 0.925(0.000) | 0.919 (0.001) | 3451.163(13.659) | 5644(316) |
| **RNG** | 0.883 (0.008) | 0.873 (0.004) | 3476.074(6.427) | 2769(16) |
| **RNG(Ny 10%)** | 0.639(0.168) | 0.660(0.057) | 3742.972(466.172) | 2767(45) |
| **RNG(Ny 2%)** | 0.781(0.010) | 0.840(0.004) | 3582.827(13.728) | 801(22) |
| **RNG(Ny 1%)** | 0.761(0.016) | 0.825 (0.009) | 3712.956(55.872) | 437(27) |
| **PRNG(k=1)** | 0.857 (0.006) | 0.858 (0.003) | 3738.329(16.182) | 863(5) |
| **PRNG(k=3)** | 0.628 (0.029) | 0.633 (0.010) | 3628.645(31.275) | 1285(12) |
| **PRNG(k=5)** | 0.639 (0.018) | 0.635 (0.010) | 3588.068(17.277) | 1712(16) |
| **RGTM** | 0.700(0.000) | 0.702 (0.015) | 3943.099(0.000) | 78518(635) |
| **RGTM(Ny 10%)** | 0.579(0.172) | 0.620 (0.076) | 3937.313(454.717) | 14582(160) |
| **RGTM(Ny 2%)** | 0.841(0.035) | 0.769 (0.010) | 3582.751(89.055) | 8045(57) |
| **RGTM(Ny 1%)** | 0.831(0.041) | 0.630 (0.017) | 3651.053(89.256) | 7247(171) |
| **PRGTM(k=1)** | 0.423(0.039) | 0.398 (0.023) | 4177.034(116.670) | 5059(372) |
| **PRGTM(k=3)** | 0.451(0.034) | 0.412 (0.016) | 4069.374(45.699) | 6755(289) |
| **PRGTM(k=5)** | 0.465(0.049) | 0.388 (0.006) | 4003.668(78.533) | 8917(1852) |

Table 7: Results on the SwissProt data set, the different methods are trained for the full data set and evaluated with the classification accuracy (column accuracy) and the dual quantization error (dualQE), and they are trained in a repeated cross-validation and evaluated by the classification error (column crossVal). The CPU time is measured for one run for the full data set.

In summary, any approximation of AP and RNG seems suited in this case if the goal is a good clustering structure (as measured by the dual quantization error), while AP and PAP lead to the best classification accuracy in this setting. For comparison, an SVM has been trained in a 2-fold cross-validation for the given Gram matrix (standard Matlab implementation in bioinformatics toolbox). It yields 98% classification accuracy. Hence AP and PAP are in the same order of magnitude, albeit they do not take into account the class labels during training. Interestingly, SVM training is quite affected when taking the Nyström approximation for speed-up. The classification accuracy drops down to 86% for 1% data for approximation, and to 63% for 10% data for approximation. These results are in the same order of magnitude as the classification results of the Nyström approximation of RNG.

## Quality of the Nyström approximation

In the experiments, we generally observe the fact that patch processing leads to better results the larger the approximation parameter $k$, which is expected. In contrast, the Nyström approximation does not alway lead to better results if a larger fraction of the data is used for the approximation. In this paragraph, we investigate in how far this observation can be traced back to the quality of

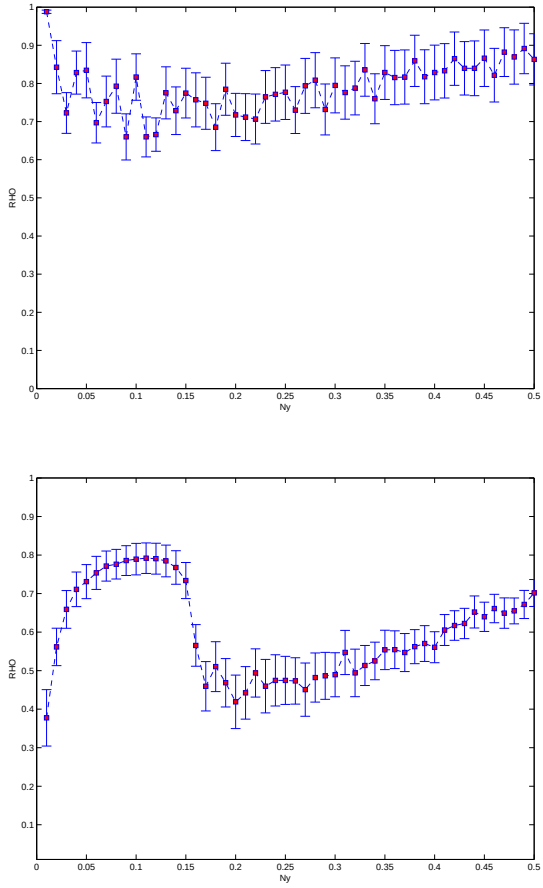


Figure 2: Quality of the Nyström approximation as evaluated by the Spearman correlation of the rows of the approximated matrix and the original one. The approximation is based on a different fraction of the data set as indicated by the x-achses. The graphs show the result for the Chromosomes (top) and Vibrio (bottom).

the Nyström approximation when using different fractions of the data. For this purpose, we sample a fraction ranging from 1% up to 50% (10% for SwissProt) of the data based on which the Nyström approximation is computed. For the clustering techniques, the absolute value of the dissimilarities is generally less relevant as compared to the induced order of the data. Therefore, we compare the resulting Nyström approximation with the original dissimilarity matrix taking Spearman correlation of the rows. We average over ten random selections of the data.

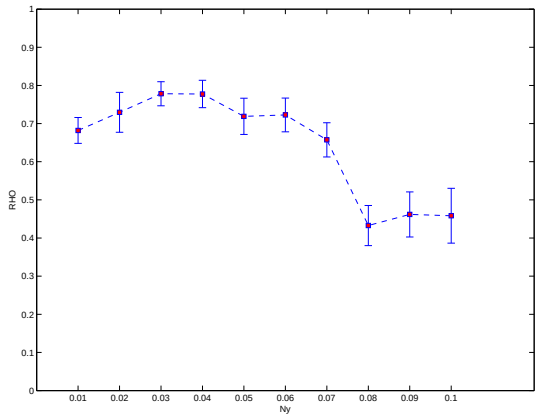


Figure 3: Quality of the Nyström approximation as evaluated by the Spearman correlation for SwissProt.

The results are displayed in Fig. 2 and Fig. 3 for all three data sets. All values are significant corresponding to p-values smaller than 0.1. Interestingly, the graphs are not monotonously increasing. That means, the Nyström approximation in terms of the induced ordering of data points does not necessarily become better the larger the fraction of the data used for approximation. In particular for Chromosomes, approximations with only 1% of the data yield an almost perfect correlation, while this drops down for larger sizes. In contrast, the approximation quality of the Vibrio data set increases for a percentage $\leq 15\%$. This observation can be one reason why the accuracy of the clustering results is also not monotonic with respect to the percentage of data points as detailed above. For SwissProt, the approximation quality seems universally good for a range from 1% to 5% resulting in a high correlation.

## Computational complexity

As already mentioned, the computational complexity of all clustering techniques AP, RGTM, and RNG is $\mathcal{O}(N^2)$. Further, the full dissimilarity matrix $D$ has to be computed, which also has time and space complexity $\mathcal{O}(N^2)$. Where are the current limits for the exact methods? Assuming double precision and standard memory of 12 Gigabytes, a dissimilarity matrix of up to 30,000 objects would currently fit into main memory. Hence the SwissProt data set is already in the order of data sets which only just fit into main memory – it amounts to about 500 Megabytes. Probably the larger bottleneck is given by the computation of the dissimilarity matrix. For the SwissProt data set, its computation took about 8 days (using the Bioinformatics toolbox of Matlab for Smith Waterman).

If either approximation is used, the computational complexity as well as the size of the matrix which is required reduces to a linear part with respect to $N$,

assuming fixed approximation parameters (i.e. a fixed percentage of data for the Nyström approximation and a fixed patch size, respectively. Further, naturally, initialization of RGTM (Ny) has to rely on landmark MDS to transfer this linear complexity to the initialization.) For an approximation with 100 points for the Nyström technique or patch sizes of 100 points, the required space would reduce to about 4.5 Megabytes and a computation time of less than 2 hours for the required dissimilarities.

# 4 Conclusions

We have presented two different ways to speed-up prototype-based clustering of dissimilarity data: the Nyström approximation and patch processing. We showed whether and how these techniques can be included into three representative clustering techniques, affinity propagation, relational neural gas, and relational generative topographic mapping, respectively. Assuming a fixed quality of the approximation, this way, linear instead of squared complexity can be achieved. The corresponding speed up has been presented in a large scale setting, a fraction of the popular SwissProt data set.

As demonstrated in several experiments, the approximation techniques lead to a decrease of accuracy which is, depending on the setting, only in the range of a few percentage. However, the result depends on the chosen evaluation measure, the chosen approximation technique, and the data set. It seems that patch approximation is often suited if $k$ is chosen sufficiently large. The Nyström approximation gives good results if the quality of the approximation of the dissimilarity matrix is sufficient. This can depend on the data set and, surprisingly, it is not necessarily monotonic with respect to the fraction of the data taken into account.

Note that the two approximation techniques presented in this paper are suited for different settings. For the Nyström approximation, it can be specified a priori which parts of the dissimilarity matrix are required for training. To be applicable, however, a representative subsample of the data need to be available a priori, thus it is not suited if streaming data are dealt with. Further, it can only be integrated into algorithms which include the dissimilarity matrix in full matrix form. Affinity propagation, which deals with these values in a distributed way, cannot be accelerated using this technique.

In contrast, patch processing requires a (right from the beginning fixed) linear part of the dissimilarity matrix located along the diagonal, and, in addition, dissimilarities of data and exemplars which are determined during training only. Hence it can be applied only in settings where additional dissimilarities can be retrieved on demand. However, due to its way in which information is compressed, it can also deal with streaming data. Hence it seems particularly suited for life-long adaptation. Further, it constitutes a very simple meta-heuristic which is directly applicable for every clustering scheme which represents clusters in terms of exemplars, and which can deal with multiple data points in an efficient way.

Since both techniques rely on a linear subpart of the full dissimilarity matrix, they can lead to a severe information loss: in theory, it is possible that the most relevant information is located at those parts of the dissimilarity matrix which are not even computed in the approximations. Thus, there cannot exist a non-trivial upper bound on the information loss if no further assumptions on the dissimilarity matrix are assumed. However, in practice, it seems that the approximation schemes are well suited to preserve the relevant information. This is mirrored in an information loss of only a few percentage provided an appropriate approximation technique is chosen.

The best approximation technique as well as the meta-parameters such as the size of the subsamples and the number of exemplars used to approximate the prototypes are not clear a priori and the quality can differ a lot depending on the data setting. Hence it would be desirable to derive characteristics of the dissimilarity matrix which guarantee that a certain approximation technique is well suited in the given setting. This question could be tackled empirically as well as from a theoretical side. It is the subject of ongoing research.

# References


[1] N. Alex, A. Hasenfuss, and B. Hammer. Patch clustering for massive data sets. *Neurocomputing*, 72(7-9):1455–1469, 2009.

[2] 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.

[3] M. Biehl, B. Hammer, S. Hochreiter, S. Kremer, and T. Villmann, editors. *Similarity-based learning on structures, 15.02.09 - 20.02.09*, volume 09081 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.

[4] C. M. Bishop, M. Svensén, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[5] 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 knowledgebase and its supplement trembl in 2003,. *Nucleic Acids Research*, 31:365–370.

[6] R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel som and related laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257 – 1273, 2008. Progress in Modeling, Theory, and Application of Computational Intelligenc - 15th European Symposium on Artificial Neural Networks 2007, 15th European Symposium on Artificial Neural Networks 2007.

[7] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[8] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann. Batch and median neural gas. *Neural Networks*, 19:762–771, 2006.

[9] F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Exploration*, 2(1):51–57, 2000.

[10] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[11] 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.

[12] A. Gisbrecht, B. Mokbel, and B. Hammer. The Nyström approximation for relational generative topographic mappings. In B. Hammer, F. Sha, L. van der Maaten, and A. Smola, editors, *Workshop proceedings of the NIPS workshop on challenges of Data Visualization*, 2010.

[13] A. Gisbrecht, B. Mokbel, and B. Hammer. Relational generative topographic map. In M. Verleysen, editor, *ESANN'10*, pages 277–282. D side, 2010.

[14] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[15] B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229–2284, 2010.

[16] B. Hammer and B. Jain. Neural methods for non-standard data. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks'2004*, pages 281–292. D-side publications, 2004.

[17] T. Hoffmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[18] A. Juan and E. Vidal. On the use of normalized edit distances and an efficient k-nn search technique (k-aesa) for fast and accurate string classification. In *ICPR*, pages 2676–2679, 2000.

[19] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[20] T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8-9):945–952, 2002.

[21] C. Lundsteen, J-Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin g-banded human metaphase chromosomes. *Clinical Genetics*, 18:355–370, 1980.

[22] T. Maier, S. Klebel, U. Renner, and M. Kostrzewa. Fast and reliable maldi-tof ms–based microorganism identification. *Nature Methods*, (3), 2006.

[23] T. Martinetz, S. Berkovich, and K. Schulten. ¨"Neural-gas"¨ Network for Vector Quantization and its Application to Time-Series Prediction. *IEEE-Transactions on Neural Networks*, 4(4):558–569, 1993.

[24] M. Neuhaus and H. Bunke. Edit distance based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

[25] E. Pekalska and R. Duin. *The dissimilarity representation for pattern recognition*. World Scientific, 2005.

[26] J. Ramsay and B. Silverman. *Functional data analysis*. Springer, 2005.

[27] T. Villmann and S. Haase. Divergence-based vector quantization. *Neural Computation*, 23(5):1343–1392, 2011.

[28] 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.

[29] H. Yin. On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural Networks*, 19(6-7):780 – 784, 2006. Advances in Self Organising Maps - WSOM'05.

[30] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nystrom low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1232–1239, New York, NY, USA, 2008. ACM.