Clustering Very Large Dissimilarity Data Sets

Barbara Hammer¹ and Alexander Hasenfuss²

¹ University of Bielefeld, CITEC, Germany

² Clausthal University of Technology,
Department of Computer Science, Germany

Abstract. Clustering and visualization constitute key issues in computer-supported data inspection, and a variety of promising tools exist for such tasks such as the self-organizing map (SOM) and variations thereof. Real life data, however, pose severe problems to standard data inspection: on the one hand, data are often represented by complex non-vectorial objects and standard methods for finite dimensional vectors in Euclidean space cannot be applied. On the other hand, very large data sets have to be dealt with, such that data do neither fit into main memory, nor more than one pass over the data is still affordable, i.e. standard methods can simply not be applied due to the sheer amount of data. We present two recent extensions of topographic mappings: relational clustering, which can deal with general proximity data given by pairwise distances, and patch processing, which can process streaming data of arbitrary size in patches. Together, an efficient linear time data inspection method for general dissimilarity data structures results. We present the theoretical background as well as applications to the areas of text and multimedia processing based on the generalized compression distance.

1 Introduction

The availability of electronic data increases dramatically in nearly every aspect of daily life, and it is estimated that the amount of electronic data doubles roughly every twenty months. At the same time, the quality and information content of modern data is constantly improving, and, often, data are represented in dedicated structures such as XML files, sequences, or graph structures. While both aspects allow the access to detailed and specialized information according to the current need of the user, this access is far from trivial and often buried in the sheer amount of information. Because of these problems automatic data mining tools play a major role to visualize and customize data such that humans can more easily access the relevant information at hand.

A multitude of different techniques has been developed in the context of interactive information visualization and data mining [23,33]. One very popular technique is offered by the self-organizing map (SOM) [18]: a lattice of neurons is arranged according to the data similarity such that clustering, browsing, and visualization become possible. This way, SOM can serve as data inspection tool or as preprocessing step for further information processing. In consequence, SOM

has successfully been applied to diverse areas such as robotics, telecommunication, web and text mining, bioinformatics, etc. [18].

With increasing quality and availability of electronic data, new challenges arise: data sets become larger and larger and, often, data are not available a priori, rather, data are stored in a distributed way and can be accessed only in sequential order. This property makes it infeasible to keep all data items in main memory at once or to sift through the data more than once. In consequence, many standard data mining methods cannot be applied in such situations, such as e.g. standard SOMs with online or batch training. Due to this fact, data mining for very large or streaming data has become one central issue of research. Several approaches have been proposed including extensions of k-means clustering and variants with approximation guarantees, heuristics which rely on sampling and according statistical guarantees or grid based methods, such as e.g. CURE, STING, and BIRCH, or iterative compression approaches which process only a fixed subset of the given data at a time [6,10,1,2,20,9,12,32,35].

Another challenge is the fact that more and more electronic data are stored together with dedicated structures such as XML files, temporal or spatial sequences, trees, or graph structures. These data are typically not represented as standard Euclidean vectors and a comparison of data by means of the Euclidean metric is often not appropriate. Rather, problem-adapted specific choices should be used such as alignment distances, tree or graph kernels, etc. Most classical statistical data mining tools such as SOM have been proposed for vectorial data and, hence, they are no longer applicable in such situations. A variety of methods which extend SOM to more general data structures has been proposed: Statistical interpretations of SOM as considered in [11,17,30,31] allow to change the generative model to more general data. The approaches are very flexible but computationally quite demanding. For specific data structures such as time series, recursive models have been proposed [3,14]. Online variants of SOM have been extended to general kernels [34]. However, these versions have been derived for (slow) online adaptation only. The approach [19] provides a fairly general method for large scale application of SOM to non-vectorial data: batch optimization is extended to general proximities by means of the generalized median. Thereby, prototype locations are restricted to data points which restricts the flexibility of the approach compared to the original setting.

Note that large data sets constitute a particular problem for SOMs for general data structures: if data are characterized by pairwise dissimilarities instead of Euclidean vectors, squared complexity is already necessary to store the relevant information in a dissimilarity matrix. In consequence, training algorithms display at least quadratic complexity which is infeasible for large data sets. Only few proposals address this problem such as [4,5]. Here, information is approximated by a small representative subset of the matrix and extended to the full data afterwards. The drawback of this approach is that a representative subset of the dissimilarity matrix has to be available prior to training, which is commonly not the case for streaming data without direct random access.

In this contribution, we present two ideas which extend SOMs to large dissimilarity data sets such that a method with linear time and constant memory results. On the one hand, we extend SOM to data given by pairwise dissimilarities in a way which is similar to relational fuzzy clustering as derived in [15,16]. We transfer these settings to SOM together with an investigation of the underlying mathematics for general data sets. On the other hand, we rely on extensions of clustering to large data sets by means of patch clustering as proposed in [1,10,6]. We extend this scheme to the relational approach resulting in a constant memory and linear time method. We demonstrate two applications to text files and symbolic musical data.

2 Relational Self-Organizing Maps

2.1 Standard Batch SOM and Corresponding Cost Function

Topographic maps constitute effective methods for data clustering, inspection, and preprocessing. Classical variants deal with vectorial data $\mathbf{x} \in \mathbb{R}^n$ which are distributed according to an underlying distribution P in the Euclidean space. The goal of prototype-based clustering algorithms is to distribute prototypes $\mathbf{w}^i \in \mathbb{R}^n$, $i = 1, \dots, k$ among the data such that they represent the data as accurately as possible. A new data point $\mathbf{x}$ is assigned to the winner $I(\mathbf{x})$ which refers to the prototype with smallest averaged distance $\sum_{l=1}^k h_\lambda(\text{nd}(i, l)) \cdot \|\mathbf{x} - \mathbf{w}^l\|^2$. Here, $h_\lambda(t) = \exp(-t/\lambda^2)$ denotes an exponential function with neighborhood λ and $\text{nd}(i, l)$ denotes a priorly chosen neighborhood structure of neurons, often induced by a low dimensional lattice in a low-dimensional Euclidean or hyperbolic space, to achieve greater flexibility of the lattice structure [24]. Prototypes are adapted such that SOM optimizes the cost function [17]

$$E_{\text{SOM}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^k \int \delta_{i, I(\mathbf{x})} \cdot \sum_{l=1}^k h_\lambda(\text{nd}(i, l)) \cdot \|\mathbf{x} - \mathbf{w}^l\|^2 P(d\mathbf{x})$$

Often, this cost function is optimized by means of an online stochastic gradient descent. Alternatively, if data $\mathbf{x}^1, \dots, \mathbf{x}^m$ are available priorly, batch optimization can be done. The corresponding discrete cost function is given by

$$E_{\text{SOM}}(\mathbf{w}, \mathbf{x}) = \frac{1}{2} \sum_{i=1}^k \sum_{j=1}^m \delta_{i, I(\mathbf{x}^j)} \cdot \sum_{l=1}^k h_\lambda(\text{nd}(i, l)) \cdot \|\mathbf{x}^j - \mathbf{w}^l\|^2.$$

This is optimized by an iterative optimization of assignments and prototypes until convergence in Batch SOM, see Algorithm 1. The neighborhood cooperation is usually annealed to $\lambda \rightarrow 0$ during training such that the quantization error is optimized in the limit of small neighborhood size. It has been shown in [8] that this procedure converges after a finite number of steps to a local optimum of the SOM cost function for fixed λ . In the following theoretical considerations, we will always assume a fixed and small neighborhood parameter λ . This consideration approximately corresponds to final stages of training.

Algorithm 1. Batch SOM**input**data $\{\mathbf{x}^1, \dots, \mathbf{x}^m\} \subset \mathbb{R}^n$;**begin** **init** $\mathbf{w}^i$ randomly; **repeat** set $I(j) = \operatorname{argmin}_i \sum_{l=1}^k h_\lambda(\operatorname{nd}(i, l)) \cdot \|\mathbf{x}^j - \mathbf{w}^l\|^2$; set $\mathbf{w}^i = \sum_j h_\lambda(\operatorname{nd}(I(j), i)) \mathbf{x}^j / \sum_j h_\lambda(\operatorname{nd}(I(j), i))$; **until** convergence; **return** $\mathbf{w}^i$;**end.****2.2 Pseudo-Euclidean Embedding of Dissimilarity Data**

In the following, we restrict the presentation to an overview of the key ingredients, omitting a few details and all proofs, which can be retrieved from [13]. Here, we deal with the application of SOM to settings where no Euclidean embedding of the data points is known. We will more generally deal with data points x^i which are characterized by pairwise dissimilarities $d_{ij} = d(x^i, x^j)$ which are symmetric and 0 if $x^i = x^j$. D denotes the corresponding matrix of dissimilarities. It can hold that no isometric Euclidean embedding can be found, i.e. SOM for Euclidean data is not applicable in this case. Discrete data such as DNA sequences or strings and alignment of those constitute one example.

For every matrix D , the so-called pseudo-Euclidean embedding exists, i.e. a vector space with symmetric bilinear form $\langle \cdot, \cdot \rangle$ (which need not be positive definite) and embeddings $\mathbf{x}^i$ of the points x^i can be found such that $d(x^i, x^j) = \langle \mathbf{x}^i - \mathbf{x}^j, \mathbf{x}^i - \mathbf{x}^j \rangle$ [26]. More precisely, define the centering matrix $J = I - 1/m\mathbf{1}\mathbf{1}^t$ with identity matrix I and the vector $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^m$. Define the generalized Gram matrix associated to D as $G = -\frac{1}{2} \cdot J D J$. Obviously, this matrix is symmetric and, thus, it can uniquely be decomposed into the form $G = Q \Lambda Q^t$ with orthonormal matrix Q and diagonal matrix of eigenvalues Λ with p positive and q negative entries. Taking the square root of Λ allows the alternative representation in the form

$$G = X I_{pq} X^t = Q |\Lambda|^{1/2} \begin{pmatrix} I_{pq} & 0 \\ 0 & 0 \end{pmatrix} |\Lambda|^{1/2} Q^t$$

where I_{pq} constitutes a diagonal matrix with p entries 1 and q entries -1 , i.e. $X = Q_{p+q} |\Lambda_{p+q}|^{1/2}$ where only $p + q$ nonzero eigenvalues of Λ are taken into account. We can define the symmetric bilinear form in $\mathbb{R}^{p+q}$

$$\langle \mathbf{x}, \mathbf{y} \rangle_{pq} = \sum_{i=1}^p x_i y_i - \sum_{i=p+1}^{p+q} x_i y_i.$$

Then, the columns of X constitute vectors $\mathbf{x}^i$ with pairwise dissimilarities $d_{ij} = \langle \mathbf{x}^i - \mathbf{x}^j, \mathbf{x}^i - \mathbf{x}^j \rangle_{pq}$. This embedding is referred to as pseudo-Euclidean embedding of the data points. The values $(p, q, m - p - q)$ are referred to as signature.

Since the algorithm of Batch SOM (Algorithm 1) relies on vector operations and the computation of dissimilarities only, this algorithm can directly be used for arbitrary dissimilarity data embedded in pseudo-Euclidean space: given D , we can compute corresponding vectors $\mathbf{x}$ such that $\langle \mathbf{x}^i, \mathbf{x}^j \rangle_{pq} = d_{ij}$ holds for all data points. For these vectors and the corresponding dissimilarities, Batch SOM can directly be applied whereby the bilinear form $\langle \mathbf{x} - \mathbf{w}, \mathbf{x} - \mathbf{w} \rangle_{p,q}$ takes the role of the squared Euclidean norm $\|\mathbf{x} - \mathbf{w}\|^2$ in the algorithm.

However, there are two drawbacks of this naive procedure: on the one hand, we rely on the pseudo-Euclidean embedding the computation of which is costly (cubic complexity) and the interpretation of which is not clear. The prototypes are points in this pseudo-Euclidean space and probably depend on the chosen embedding, i.e. another vectorial embedding which shares the properties of the pseudo-Euclidean one could probably lead to fundamentally different prototypes. On the other hand, a connection of the SOM algorithm in pseudo-Euclidean space to the SOM cost function or, more desirable, a cost function which is independent of the concrete embedding is not clear as well as connected properties such as convergence of the algorithm.

Now we first reformulate batch SOM in pseudo-Euclidean space such that it becomes independent of the concrete pseudo-Euclidean embedding and it is based on the dissimilarity matrix D only. Afterwards, we discuss a connection to a cost function and questions concerning the convergence of the algorithm.

2.3 Relational SOM

The key observation to reformulate batch SOM independent of a concrete embedding of points consists in the fact that prototypes are located on special positions of the vector space only. Prototypes have the form

$$\mathbf{w}^i = \frac{\sum_j h_\lambda(\text{nd}(I(j), i)) \mathbf{x}^j}{\sum_j h_\lambda(\text{nd}(I(j), i))} = \sum_j \alpha_{ij} \mathbf{x}^j$$

where $\sum_j \alpha_{ij} = 1$. Further, one can compute that for every linear combinations $\sum_j \alpha_{ij} \mathbf{x}^j$ and $\sum_j \alpha'_{ij} \mathbf{x}^j$ with $\sum_j \alpha_{ij} = 1$ and $\sum_j \alpha'_{ij} = 1$ and symmetric bilinear form $\langle \cdot, \cdot \rangle$ with $d_{ij} = \langle \mathbf{x}^i - \mathbf{x}^j, \mathbf{x}^i - \mathbf{x}^j \rangle$ the following is valid:

$$\left\langle \sum_i \alpha_i \mathbf{x}^i - \sum_i \alpha'_i \mathbf{x}^i, \sum_i \alpha_i \mathbf{x}^i - \sum_i \alpha'_i \mathbf{x}^i \right\rangle = (\alpha')^t D \alpha - \frac{1}{2} \cdot \alpha^t D \alpha - \frac{1}{2} \cdot (\alpha')^t D \alpha'$$

These observations immediately allow us to reformulate batch SOM in pseudo-Euclidean space independent of the concrete vector embedding of points. The key issues are to substitute prototype locations $\mathbf{w}^i = \sum_{ij} \alpha_{ij} \mathbf{x}^j$ by coefficient vectors $\alpha_i = (\alpha_{i1}, \dots, \alpha_{im})$ and to substitute the computation of dissimilarities

Algorithm 2. Relational SOM

inputsymmetric dissimilarity matrix with zero diagonal $D \in \mathbb{R}^{m \times m}$;**begin****init** $\alpha_{ij} \geq 0$ such that $\sum_j \alpha_{ij} = 1$;**repeat** compute $\text{dist}_{ij} = [D\alpha_i]_j - \frac{1}{2} \cdot \alpha_i^t D\alpha_i$; set $I(j) = \operatorname{argmin}_i \sum_{l=1}^k h_\lambda(\text{nd}(i, l)) \cdot \text{dist}_{jl}$; set $\alpha_{ij} = h_\lambda(\text{nd}(I(j), i)) / \sum_j h_\lambda(\text{nd}(I(j), i))$;**until** convergence;**return** α_{ij} ;**end.**

of prototypes and data points by means of the above formula. The resulting algorithm, Relational SOM, is displayed as (Algorithm 2).

This algorithm is equivalent to vectorial SOM under the identity $\mathbf{w}^i = \sum_{ij} \alpha_{ij} \mathbf{x}^j$ whereby it does not rely on an explicit embedding but on the pair-wise dissimilarities only. This shows that the winner assignments of SOM in pseudo-Euclidean space are independent of the concrete embedding.

2.4 Convergence

Since relational SOM is equivalent to batch SOM in pseudo-Euclidean space, it follows immediately that the algorithm converges towards a fixed point of the algorithm whenever the embedding is in fact Euclidean: in this situation, standard SOM is implicitly performed in the (Euclidean) space and the corresponding guarantees of the standard batch SOM as e.g. revisited in [8] hold. If the pseudo-Euclidean embedding is a non-Euclidean embedding, i.e. negative eigenvalues occur, the guarantees of standard batch SOM do not hold and one can indeed find situations where divergence of the algorithm is observed, i.e. the algorithm ends up in cycles. Batch SOM in Euclidean space relies on a subsequent optimization of the underlying cost function with respect to prototype locations and winner computations. Since an optimum is picked at both steps, the cost function decreases in every epoch in the Euclidean setting and because of the finite number of different possible winner assignments, convergence of the algorithm can be observed after a finite number of epochs.

For the non-Euclidean setting, it turns out that the subsequent computation of prototypes and winner assignments does not necessarily compute optima in the two steps. While it can easily be seen that the winner assignment as given in SOM is optimal, the choice of prototypes as generalized mean of the data points according to the winner assignment is not: it can happen that a saddle point

Fig. 1. Example of points in pseudo-Euclidean space for which relational clustering does not converge to a fixed point. It is indicated by arrows which points cyclically change their cluster assignments.

with worse cost function value instead of an optimum is picked if the negative directions of the bilinear form are dominant in this situation. Then an optimum would lie at the borders of the solution space since it would be beneficial to push prototypes away from the data with respect to the negative directions of the pseudo-Euclidean space. An efficient determination of the global optimum, however, is not possible because non-convex quadratic programming is NP hard [29,25]. Further, even if optima at the borders of the solution space could be computed easily in this setting, their usefulness can be doubted and numerical problems are likely to occur. Thus, the choice of the prototypes as generalized mean, which optimizes the problem only with respect to the positive directions of the pseudo-Euclidean space, seems a reasonable compromise in this situation.

Due to this fact, divergence can occur. Consider, for example, pseudo-Euclidean space with signature $(1, 1, 0)$ and the points as displayed in Fig. 1. For small enough neighborhood range λ , the cluster assignment of four points changes in every epoch as indicated by the arrows. Note, however, that this setting heavily relies on a large contribution of the negative axes to the bilinear form. In practice, the positive directions usually by far outweigh the negative directions such that this problem seems unlikely in practice: we did not observe divergence of the algorithm for real life data sets in a single experiment so far.

2.5 Dual Cost Function

The SOM cost function as introduced above constitutes a reasonable evaluation measure of batch SOM in pseudo-Euclidean space, albeit cases exist at least in theory where no local optimum of this cost function is actually found by batch SOM in this setting as discussed above. This cost function relies on the pseudo-Euclidean embedding and has no meaning if only the dissimilarity matrix D is available. Fortunately, one can substitute the SOM cost function by a so-called dual function for every fixed point of relational SOM which does not depend on the concrete pseudo-Euclidean embedding but which relies on D and the winner assignments only. More precisely, the following equality holds: assume a fixed point of relational SOM and batch SOM in pseudo Euclidean space, respectively,

is found, characterized by winner assignments $I(j)$ and corresponding prototypes $\mathbf{w}^i$. Then, we find $E_{\text{SOM}}(\mathbf{w}, \mathbf{x}) = E_{\text{SOM}}^\vee(I, \mathbf{x})$ where

$$E_{\text{SOM}}^\vee(I, \mathbf{x}) := \sum_i \frac{1}{4 \sum_j h_\lambda(\text{nd}(I(j), i))} \sum_{jj'} h_\lambda(\text{nd}(I(j), i)) h_\lambda(\text{nd}(I(j'), i)) d_{jj'}$$

The dual SOM cost function relies on winner assignments only and it measures the dissimilarities of data points within clusters as defined by the winner assignments, averaged over the prior topological neighborhood structure. As such, it is independent of a vectorial embedding of data but constitutes a valid cost function for clusterings based on dissimilarity data only. Since it coincides with the standard SOM cost function for fixed points of the algorithm, we can conclude that also the value of the standard batch SOM cost function is independent of the concrete vectorial embedding of data for fixed points of the algorithm. Further, for the euclidean case, also the overall structure of these two cost functions coincides in the sense that a one-one correspondence of global optima of the SOM cost function and its dual holds. For non-Euclidean settings, winner assignments can correspond to saddle points of the SOM cost function.

We would like to mention that the identification of the dual SOM cost function gives rise to another (at least theoretical) possibility to deal with non-Euclidean dissimilarities D . In analogy to relational k-means clustering as discussed e.g. in [28,21], the shape of the dual SOM cost function is approximately independent of constant shifts of off-diagonal elements of D for small neighborhood λ and common lattice structures nd . More precisely, if $\tilde{d}_{ij} = d_{ij} + d_0 \cdot (1 - \delta_{ij})$, d_0 denoting a positive constant and δ_{ij} the Kronecker delta, one can find for any fixed winner assignment I

$$E^\vee(I, \tilde{d}_{ij}) - E^\vee(I, d_{ij}) = \frac{1}{4} \left(\sum_{ij} h(I(j), i) - k \right)$$

where k denotes the number of prototypes. If we assume that the neighborhood function $\text{nd}(i, j)$ is nonnegative and it equals 0 if and only if $i = j$, then the term $\sum_{ij} h(I(j), i)$ converges to m for small λ , i.e. the difference of the two cost functions becomes a constant, and hence local optima can be observed at exactly the same places. Since I is discrete, this already holds for approximations of the limit setting.

Since every symmetric dissimilarity measure becomes Euclidean for large shifts of the off-diagonal elements, one could in principle first use the minimum shift which is necessary to make data Euclidean, and apply standard SOM in Euclidean space or its relational counterpart, afterwards, whereby convergence is guaranteed. Unfortunately, as demonstrated in [13], the numerics can become more difficult for shifted data, since differences in the dissimilarities are more and more annihilated such that relational SOM for non-Euclidean data can lead to better results than relational SOM for the corresponding shifted Euclidean data. Hence, the direct application of relational SOM might be advisable albeit convergence is not guaranteed in general.

3 Patch Processing

The time and space complexity of relational SOM constitutes one major drawback of the method: space complexity is linear, and time complexity is quadratic with respect to the number of data points (assuming a constant number of prototypes and epochs of the method). Note that the information which represents the data, the dissimilarity matrix D , itself is quadratic with respect to the number of data, i.e. every method which is linear necessarily has to pick only subsets of the full matrix D , neglecting possible relevant information of D .

Obviously, its complexity makes relational SOM unsuitable for large data sets. Therefore, we propose an approximative method which yields to an intuitive linear time and constant space algorithm, using patch clustering as introduced for Euclidean SOM in [1]. Our assumption is that single data points are available in consecutive order. Further, we assume the existence of a method to compute the pairwise dissimilarity of data on demand. These assumptions are quite reasonable if a very large (possibly distributed) data set is dealt with for which some problem specific dissimilarity measure is used. It is in general infeasible to precalculate the full dissimilarity matrix D in this case, such that methods which reduce the necessary information on demand to only a linear part of matrix D are favourable.

The main idea of the proposed method is to process data in patches which fit into main memory. Every patch is compressed by means of the resulting prototypes and their multiplicities resp. a limited-space approximation thereof. This information serves as additional input for the next patch computation such that all already seen data are implicitly taken into account in every step.

More precisely, we process m data points in n_p patches of priorly fixed size $p = m/n_p$. We assume divisibility of m by n_p for simplicity. A patch P_t is represented by the corresponding portion of the dissimilarity matrix D :

$$P_t = (d_{sl})_{s,l=(t-1) \cdot p+1, \dots, t \cdot p} \in \mathbb{R}^{p \times p}$$

which represents the dissimilarities of points $(t-1) \cdot p + 1, \dots, t \cdot p$. For every patch, the prototypes of the former patch are added with multiplicities according to the size of their receptive fields, and relational SOM is applied to this extended patch. Thereby, it is no longer possible to represent prototypes by means of the full coefficient vectors α_i since this would require linear space for storage and, in addition, computing the dissimilarity of prototypes and data points would eventually require the full (quadratic) dissimilarity matrix D - both demands are infeasible for large data sets. Therefore, we use an approximation of the prototype representation by a finite number K of coefficients with maximum contribution and (since these can be more than K) which are closest to the respective prototype. Formally, a K -approximation refers to the K indices $j_1, \dots, j_K$ corresponding to points $\mathbf{x}^{j_1}, \dots, \mathbf{x}^{j_K}$ with smallest dissimilarity to $\mathbf{w}^i$.

This definition gives rise to a formalization of extended patches which are processed in turn in patch processing and which specify the parts of the matrix D necessary for the algorithm. Assume the current patch P_t is considered. Assume

N_{t-1} refers to the index set of the K -approximation of all prototypes obtained in the previous step. When considering k prototypes, the size of this set is $|N_{t-1}| = k \cdot K$. For the next round of patch processing, dissimilarity clustering is applied to the points corresponding to the indices in N_t and the data from the current patch, i.e. we need the following part of the dissimilarity matrix

$$P_i^* = \begin{pmatrix} d(N_{t-1}) & d(N_{t-1}, P_t) \\ d(N_{t-1}, P_t)^t & P_t \end{pmatrix}$$

where $d(N_{t-1}) = (d_{uv})_{u,v \in N_{t-1}}$ denotes the inter-dissimilarities of points from the K -approximation, and $d(N_{t-1}, P_t) = (D_{uv})_{u \in N_{t-1}, v=(t-1) \cdot p+1, \dots, t \cdot p}$ denotes the dissimilarities of points in the K -approximation and the current patch. We refer to P_i^* as extended patches.

Based on these data handling techniques, patch relational SOM can be defined as iterative processing of patches enriched by the K -approximation of prototypes from the previous patch. The prototypes contribute to the new clustering task according to the sizes of their receptive fields R_i , i.e. a prototype $\mathbf{w}^i$ is counted with multiplicity $|R_i|$. Correspondingly, every point $\mathbf{x}^j$ in N_{t-1} contributes with multiplicity $m_j = |R_i|/K$. It is straightforward to extend relational SOM to deal with multiplicities m_j of point $\mathbf{x}^j$. The only change concerns the update of the coefficients α_{ij} , which is enriched by the multiplicities. Taking this as subroutine, patch relational SOM can be performed as shown in Algorithm 3.

After processing, a set of prototypes together with a K -approximation is obtained which compresses the full data set. An inspection of prototypes is easily possible by looking at the points which are closest to these prototypes.

Note that the algorithm runs in constant space if the size p of the patches is chosen independently of the data set size m . Similarly, under this assumption, the fraction of the distance matrix which has to be computed for the procedure is of linear size $\mathcal{O}(m/p \cdot p) = \mathcal{O}(m)$ and the overall time complexity of patch clustering is of size $\mathcal{O}(m/p \cdot p^2) = \mathcal{O}(mp) = \mathcal{O}(m)$, assuming constant p . Hence, a linear time and constant space algorithm for general dissimilarity data results which is suited for large data sets, if constant patch size is taken according to the available space. Note that this algorithm, unlike alternatives such as [4,5], does not require a representative subpart of the matrix D prior to training, rather, data can be processed in the order as they are accessed, i.i.d. and non i.i.d. cases leading to virtually the same final result of the algorithm due to the incorporation of the statistics of all already seen data in every patch.

4 Experiments

We present two applications of relational SOM with underlying hyperbolic lattice structure [24]. In both cases, data are treated as symbolic strings, for which general dissimilarity measures such as alignment or n-gram kernels are available. In this contribution we rely on a dissimilarity measure which has been derived from information theoretic principles and which can be considered as an efficient

Algorithm 3. Patch Relational SOM

begincut the first patch P_1 ;apply relational NG to $P_1 \rightarrow$ prototypes W_1 ;compute the K -approximation N_1 of W_1 ;update multiplicities m_i of N_1 ;set $t = 2$;**repeat**cut the next patch P_t ;construct extended patch P_t^* using P_t and N_{t-1} ;set multiplicities of points in P_t to $m_i = 1$;apply relational NG with multiplicities to $P_t^* \rightarrow$ prototypes W_t ;compute K -approximation N_t of W_t ;update multiplicities m_i of N_t ; $t \rightarrow t + 1$;**until** $t = n_P$ **return** prototypes W_{n_P} ;**end.**

approximation of a universal dissimilarity measure which minorizes every reasonable dissimilarity. The normalized compression distance (NCD) [7] is defined as

$$\text{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

where x and y are strings, xy its concatenation, and $C(x)$ denotes the size of a compression of the string x . For our experiments, bzip2 was used.

4.1 Mapping MIDI Files

In this application, classical music represented in the MIDI format should be mapped. The data set consists of a selected subset of 2061 classical pieces from the *Kunst der Fuge* archive. To extract relevant strings which represent the MIDI files, a directed graph was extracted based on pitch differences and relative timings as intermediate step. As discussed in [22], the representation is widely invariant to shifts of the overall pitch and scalings of the overall tempo, thereby emphasizing on the melody underlying the musical piece and abstracting from the specific orchestration. The paths of the graph are then concatenated and represented in byte code, leading to a string representation suited for NCD. A relational hyperbolic SOM with 85 neurons was trained. Figure 2 shows the projection of the hyperbolic grid into the Euclidean plane. Obviously, the composers sharing a common style or epoch are mostly situated close to each other.

4.2 Mapping Newsgroups

The goal of this applications is to map 183,546 articles from 13 different news-groups in a semantically meaningful way by means of a relational HSOM. The texts were preprocessed by removing stop words and applying word stemming [27]. Patch processing was applied using relational HSOM with 85 neurons and a 3 approximation of the prototypes. Since the full dissimilarity matrix would occupy approx. 250 GB space, it is no option to process the data at once or precalculate the full dissimilarity at once. Instead, 183 patches of around 1000 texts were taken and the distances of these patches were precalculated, resulting in only around 12 MB space. In addition, since patch processing requires the dissimilarities of the extended patches, around $1000 \cdot 85 \cdot 3$ dissimilarities had

Fig. 2. Relational Hyperbolic Self-Organizing Map for 2061 classical pieces which are compared by symbolic preprocessing and NCD. Labeling of the neurons is based on majority vote.

Fig. 3. Visualization of 183,546 newsgroup articles using patch relational HSOM, displaying the topographic arrangement of the biggest newsgroups on the map. In this case, patch processing reduces the required space of the full dissimilarity matrix from approx. 251 GB to only around 13 MB.

to be computed on demand for every patch. This way, the whole computation could be performed on a common desktop computer in reasonable time.

The outcome is depicted in Fig. 3. Clearly, the data arrange on the lattice according to their semantic meaning as mirrored by the corresponding newsgroup, such that visualization and browsing become possible.

5 Conclusions

We have presented an extension of SOM which is capable of mapping huge dissimilarity data sets in constant space and linear time such that visualization and data inspection become possible for the user. The main ingredients are an extension of SOM to a relational setting which corresponds to the implicit application of SOM in pseudo-Euclidean space, and a decomposition of the full clustering procedure into separate pieces by means of patch processing. Interestingly, the method is quite intuitive and can be linked to the standard SOM cost function or its dual, such that extensions such as supervision [13] are easily possible.

References

1. Alex, N., Hasenfuss, A., Hammer, B.: Patch clustering for massive data sets. *Neurocomputing* 72(7-9), 1455–1469 (2009)
2. Badoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: *Proc. STOC*, pp. 250–257 (2002)
3. De, G., Barreto, A., Araujo, A.F.R., Kremer, S.C.: A Taxonomy for Spatiotemporal Connectionist Networks Revisited: The Unsupervised Case. *Neural Computation* 15(6), 1255–1320 (2003)
4. Belongie, S., Fowlkes, C., Chung, F., Malik, J.: Spectral partitioning with indefinite kernels using the Nyström extension. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) *ECCV 2002. LNCS*, vol. 2352, pp. 531–542. Springer, Heidelberg (2002)
5. Bezdek, J.C., Hathaway, R.J., Huband, J.M., Leckie, C., Kotagiri, R.: Approximate data mining in very large relational data. In: Dobbie, G., Bailey, J. (eds.) *Proc. Australasian Database Conference*, pp. 3–13 (2006)
6. Bradley, P.S., Fayyad, U., Reina, C.: Scaling clustering algorithms to large data sets. In: *Proc. KDD*, pp. 9–15. AAAI Press, Menlo Park (1998)
7. Cilibrasi, R., Vitanyi, M.B.: Clustering by compression. *IEEE Transactions on Information Theory* 51(4), 1523–1545 (2005)
8. Cottrell, M., Hammer, B., Hasenfuss, A., Villmann, T.: Batch and median neural gas. *Neural Networks* 19, 762–771 (2006)
9. Domingos, P., Hulten, G.: A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. In: *Proc. ICML*, pp. 106–113 (2001)
10. Farnstrom, F., Lewis, J., Elkan, C.: Scalability for clustering algorithms revisited. *SIGKDD Explorations* 2(1), 51–57 (2000)
11. Graepel, T., Obermayer, K.: A stochastic self-organizing map for proximity data. *Neural Computation* 11, 139–155 (1999)
12. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large datasets. In: *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 73–84 (1998)
13. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets, Technical Report IFI-01-2010, Clausthal University of Technology (2010)
14. Hammer, B., Micheli, A., Sperduti, A., Strickert, M.: Recursive self-organizing network models. *Neural Networks* 17(8-9), 1061–1086 (2004)
15. Hathaway, R.J., Bezdek, J.C.: Nerf c-means: Non-Euclidean relational fuzzy clustering. *Pattern Recognition* 27(3), 429–437 (1994)
16. Hathaway, R.J., Davenport, J.W., Bezdek, J.C.: Relational duals of the c-means algorithms. *Pattern Recognition* 22, 205–212 (1989)
17. Heskes, T.: Self-organizing maps, vector quantization, and mixture modeling. *IEEE TNN* 12, 1299–1305 (2001)
18. Kohonen, T.: *Self-Organizing Maps*. Springer, Heidelberg (1995)
19. Kohonen, T., Somervuo, P.: How to make large self-organizing maps for non-vectorial data. *Neural Networks* 15, 945–952 (2002)
20. Kumar, A., Sabharwal, Y., Sen, S.: A simple linear time $(1+\epsilon)$ -approximation algorithm for k-means clustering in any dimensions. In: *Proc. IEEE FOCS*, pp. 454–462 (2004)
21. Laub, J., Roth, V., Buhmann, J.M., Müller, K.-R.: On the information and representation of non-Euclidean pairwise data. *Pattern Recognition* 39, 1815–1826 (2006)

22. Mokbel, B., Hasenfuss, A., Hammer, B.: Graph-based Representation of Symbolic Musical Data. In: Torsello, A., Escolano, F., Brun, L. (eds.) GBRPR 2009. LNCS, vol. 5534, pp. 42–51. Springer, Heidelberg (2009)
23. Nisbet, R., Elder, J., Miner, G.: Handbook of Statistical Analysis and Data Mining Applications. Academic Press/Elsevier (2009)
24. Ontrup, J., Ritter, H.: Hyperbolic self-organizing maps for semantic navigation. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14, pp. 1417–1424. MIT Press, Cambridge (2001)
25. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP hard. *Journal of Global Optimization* 1, 15–22 (1991)
26. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition – Foundations and Applications. World scientific, Singapore (2005)
27. Porter, M.F.: An algorithm for suffix stripping. *Program* 14(3), 130–137 (1980)
28. Roth, V., Laub, J., Kawanabe, M., Buhmann, J.M.: Optimal cluster preserving embedding of nonmetric proximity data. *IEEE TPAMI* 25(12), 1540–1551 (2003)
29. Sahni, S.: Computationally related problems. *SIAM Journal on Computing* 3(4), 262–279 (1974)
30. Seo, S., Obermayer, K.: Self-organizing maps and clustering methods for matrix data. *Neural Networks* 17, 1211–1230 (2004)
31. Tino, P., Kaban, A., Sun, Y.: A generative probabilistic approach to visualizing sets of symbolic sequences. In: Kohavi, R., Gehrke, J., DuMouchel, W., Ghosh, J. (eds.) Proc. KDD 2004, pp. 701–706. ACM Press, New York (2004)
32. Wang, W., Yang, J., Muntz, R.R.: STING: a statistical information grid approach to spatial data mining. In: Proc. VLDB, pp. 186–195 (1997)
33. Wong, P.C., Thomas, J.: Visual Analytics. *IEEE Computer Graphics and Applications* 24(5), 20–21 (2004)
34. Yin, H.: On the equivalence between kernel self-organising maps and self-organising mixture density network. *Neural Networks* 19(6), 780–784 (2006)
35. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, pp. 103–114 (1996)