Topographic Mapping of Dissimilarity Datasets

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von
Alexander Hasenfuß

genehmigt von der
Fakultät für Mathematik/Informatik und Maschinenbau
der Technischen Universität Clausthal

Tag der mündlichen Prüfung
22. September 2009

Die Arbeit wurde angefertigt am Institut für Informatik an der Technischen Universität Clausthal.

Dekan der Fakultät:	Prof. Dr. J. Dix
Berichterstatterin	Prof. Dr. B. Hammer
Mitberichterstatter	Prof. Dr. M. Verleysen
	Prof. Dr. Th. Villmann

Abstract

A great challenge today, arising in many fields of science, is the proper mapping of datasets to explore their structure and gain information that otherwise would remain concealed due to the high-dimensionality. This task is impossible without appropriate tools helping the experts to understand the data. A promising way to support the experts in their work is the *topographic mapping* of the datasets to a low-dimensional space where the structure of the data can be visualized and understood.

This thesis focuses on Neural Gas and Self-Organizing Maps as particularly successful methods for prototype-based topographic maps. The aim of the thesis is to extend these methods such that they can deal with real life datasets which are possibly very huge and complex, thus probably not treatable in main memory, nor embeddable in Euclidean space. As a foundation, we propose and investigate a fast batch scheme for topographic mapping which features quadratic convergence. This formulation allows to extend the methods to general non-Euclidean settings in two ways, on the one hand by restricting prototype locations to data points, leading to so-called median variants. On the other hand, continuous prototype updates become possible by means of an equivalent formulation of the methods in terms of pairwise dissimilarities only and the notation of generalized relational prototypes, leading to so-called relational variants. Since the methods rely on the standard cost functions of Neural Gas and Self-Organizing Maps (in the version of Heskes), further extensions to incorporate auxiliary information in terms of labels, and to control the magnification exponent of the resulting prototype distribution become possible. The dependency of the models on prototypes allows to include an intuitive patch processing scheme which turns the basic algorithms into a framework which requires only constant memory and linear time complexity also for the case of general (quadratic) dissimilarity matrices. The suitability of these methods is demonstrated in several experiments including an application to text data for which the dissimilarity matrix requires almost 250 GB storage capacity.

To Baba,
with love and thanks

Acknowledgements

I would like to thank my supervisors Barbara Hammer and Thomas Villmann for simply being the best. I am deeply grateful to them for all.

No love, no friendship can cross the path of our destiny
without leaving some mark on it forever.
– Francois Mauriac (1885 - 1970)

Contents

Chapter 1

Introduction

In many fields of science today, massive datasets are collected that have to be examined. A task that is utterly impossible without appropriate automated tools helping the experts to understand the data. For example in biomedical sciences, climate research, experimental physics, astronomical research, or life sciences, obtained data has to be grouped, arranged, and illustrated to make it accessible for visual exploration (Keim et al., 2008; Simoff et al., 2008). A promising way to support the experts in their work is the *topographic mapping* of the datasets to a low-dimensional space where the structure of the data can be visualized and understood. Moreover, those automated tools should present an interpretable visualization of prototypical data points to the experts. Then the experts can better control the exploration of the data space in the framework of an interactive tool, initiate further experiments with different settings based on their experience, or manually classify the data to integrate expert knowledge to the system, for instance.

These requirements are in a perfect way satisfied by prototype-based methods like Neural Gas and Self-Organizing Maps. Numerous successful applications of those methods have been reported in literature as, for instance, substantiated by the extensive collection of over 7000 references in the *Bibliography of Self-Organizing Map (SOM) Papers* (Kaski et al., 1998; Oja et al., 2003; Pöllä et al., 2007).

Beyond that, a particular challenge in visual analytics is the handling of large datasets, since especially in this case it is difficult to provide the results in a reasonable time. Unfortunately, the standard formulations of Neural Gas and SOMs are based on a learning scheme featuring a rate of convergence that is not sufficient for a fast adaptation of the prototypes. Therefore, a fast processing scheme for Neural Gas and SOMs is introduced in the first part of this thesis that accelerates the convergence considerably. In addition, useful extensions for the accelerated variants are presented, which incorporate supplementary information about data into the learning process, actively control the learning process with regard to the prototype distribution, and handle even much bigger datasets.

Up to now, we were discussing prototype-based methods which operate in Euclidean spaces. But in modern science today, it has become beneficial to apply special metrics to measure the pairwise proximities between the objects being regarded, since Euclidean representations were found to be not descriptive enough to map the whole structure of the data. Examples for non-Euclidean measures are, for instance, *alignment distances* from bioinformatics, *normalized compression distance* from algorithmic information theory, or *geodesic distances* from graph theory. These special suited dissimilarity measures yield very powerful representations, but in general they originate from non-Euclidean spaces, hence no vector representa-

tion is available to work with. The application of the popular standard methods k-Means, Neural Gas and Self-Organizing Maps that operate in Euclidean spaces is therefore not possible.

A couple of mining tools for non-Euclidean datasets have been proposed: First of all, the popular kernel approach that has gained a lot of attention in the last decade. By linearizing non-linearities in a kernel-defined feature space, these methods can handle various complex structured data (Filippone et al., 2008). Kernel functions proposed for complex structured data are, for instance, Edit distance-based kernel functions for strings and graphs (Neuhaus and Bunke, 2006), Fisher Kernel (Saunders et al., 2003) and String Kernel (Lodhi et al., 2002) for text processing, Graph Kernel (Kashima et al., 2004; Bach, 2008), Kernel on pointsets (Parsana et al., 2008), and Alignment Kernel from bioinformatics (Qiu et al., 2007). Later on in this thesis, we will demonstrate that our proposed prototype-based algorithms can easily be kernelized and hence are able to make use of the large collection of available kernel functions.

Another popular approach working on dissimilarity data is based on *Deterministic Annealing* (Rose, 1998) that was utilized for clustering by Hofmann and Buhmann (1999), and for a stochastic formulation of Self-Organizing Maps by Graepel and Obermayer (1999).

All these approaches have some drawbacks regarding their applicability for the interpretable visualization of data: Either they do not provide interpretable outcomes, need too much computational efforts, work only on special classes of dissimilarity data, or provide unstable results. But by now there are no *simple*, *robust*, and *efficient* methods for arbitrary dissimilarity datasets.

At this point, the thesis at hand ties up by building the bridge between the standard prototype-based methods, and the non-Euclidean spaces. An extension of Neural Gas and Self-Organizing Maps is introduced which can directly work on arbitrary dissimilarity datasets. Since the size of dissimilarity datasets is quadratic in the number of datapoints, these extended methods are also quadratic in their basic form and thus they are not applicable to huge datasets. Therefore, we propose a fast and intuitive processing scheme with a linear time and constant space complexity which constitutes one of the few prototype-based methods for very large dissimilarity datasets.

In the first part of the thesis, we lay the technical foundation of the thesis and present the standard prototype-based methods. We introduce a fast processing scheme and show its convergence. Furthermore, we propose extensions to incorporate supplementary information about data into the learning process, to actively control the learning process with regard to the prototype distribution, and to handle very large or streaming datasets using constant memory.

The second part of the thesis deals with dissimilarity data and how to reformulate the prototype-based methods to work directly on these kind of data. A first direct approach is built on the formal notion of cost functions by restricting the flexibility of prototype locations. As an alternative, full flexibility can be achieved by the so-called *relational prototypes*. We first discuss how to define these prototypes in non-vectorial spaces. Then, based on the possibility to express distances between relational prototypes and data points by only the given dissimilarities between data points, we derive relational variants of the prototype-based standard methods. Finally, we introduce an approximation scheme to transfer the efficient patch processing scheme to the non-Euclidean scenario. We demonstrate its applicability in a real-life setting where around 180.000 data objects are processed corresponding to a full dissimilarity matrix with a size of 250 gigabytes.

Finally, it should be mentioned here that the thesis at hand is based on several scientific research articles to which the author has contributed in the past four years. The thesis is therefore also meant as an extension and an unified presentation of the following articles:

Refereed Journals

- *Batch and Median Neural Gas*
  Cottrell, M., Hammer, B., Hasenfuss, A., and Villmann, T.
  Neural Networks 19:762–771, 2006
- *Magnification Control for Batch Neural Gas*
  Hammer, B., Hasenfuss, A., and Villmann, T.
  Neurocomputing 70:1225–1234, 2007
- *Patch Clustering for Massive Data Sets*
  Alex, N., Hasenfuss, A., and Hammer, B.
  Neurocomputing 72:1455–1469, 2009

Edited Books

- *Median Topographic Maps for Biomedical Data Sets*
  Hammer, B., Hasenfuss, A., and Rossi, F.
  In *Similarity-based Clustering and its Application to Medicine and Biology*
  Springer, 2009

Refereed Conferences

- *Magnification Control in Relational Neural Gas*
  Hasenfuss, A., Hammer, B., Geweniger, T., and Villmann, T.
  In *Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN 2008)*, pp. 325–330, 2008
- *Patch Relational Neural Gas - Clustering of Huge Dissimilarity Datasets*
  Hasenfuss, A., Hammer, B., and Rossi, F.
  In *Artificial Neural Networks in Pattern Recognition (ANNPR 2008)*
  Springer LNCS 5064:1–12, 2008
- *Topographic Processing of Very Large Text Datasets.*
  Hasenfuss, A., Boerger, W., and Hammer, B.
  In *Proc. ANNIE 2008*, 2008.
- Single Pass Clustering and Classification of Large Dissimilarity Datasets
  Hasenfuss, A., and Hammer, B.
  In *Artificial Intelligence and Pattern Recognition (AIPR 2008)*
  pp. 219–223, 2008
- *Relational Topographic Maps*
  Hasenfuss, A., and Hammer, B.
  In *Advances in Intelligent Data Analysis VII, Proc. IDA 2007*
  Springer LNCS 4723:93–105, 2007
- *Neural Gas Clustering for Dissimilarity Data with Continuous Prototypes*
  Hasenfuss, A., Hammer, B., Schleif, F.-M., and Villmann, T.
  In *Computational and Ambient Intelligence (IWANN 2007)*
  Springer LNCS 4507:539–546, 2007

- *Thinning Mesh Animations*
  Winkler, T., Drieseberg, J., Hasenfuss, A., Hammer, B., and Hormann, K.
  In *Proceedings of Vision, Modeling, and Visualization (VMV 2008)*, 2008

- *Relational Neural Gas*
  Hammer, B., and Hasenfuss, A.
  In *KI 2007: Advances in Artificial Intelligence*
  Springer LNAI 4667:190–204, 2007

- *Topographic Processing of Relational Data*
  Hammer, B., Hasenfuss, A., Rossi, F., and Strickert, M.
  In *Proceedings of 6th Int. Workshop on Self-Organizing Maps (WSOM 2007)*

- *Intuitive Clustering of Biological Data*
  Hammer, B., Hasenfuss, A., Schleif, F.-M., Villmann, T., Strickert, M., and Seiffert, U., In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2007)*, pp. 1877–1882, 2007

- *Accelerating Relational Clustering Algorithms with Sparse Prototype Representation*, Rossi, F., Hasenfuß, A., and Hammer, B.
  In *Proceedings of 6th Int. Workshop on Self-Organizing Maps (WSOM 2007)*

- *Magnification Control for Batch Neural Gas*
  Hammer, B., Hasenfuß, A., and Villmann, T.
  In *Proceedings of the 14th European Symposium on Artificial Neural Networks (ESANN 2006)*, pp. 7–12, 2006

- *Supervised Batch Neural Gas*
  Hammer, B., Hasenfuss, A., Schleif, F.-M., and Villmann, T.
  In *Artificial Neural Networks in Pattern Recognition*
  Springer LNAI 4087:33–45, 2006

- *Supervised Median Clustering*
  Hammer, B., Hasenfuss, A., Schleif, F.-M., and Villmann, T.
  In *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2006)* 16:623–632, 2006

- *Batch Neural Gas*
  Cottrell, M., Hammer, B., Hasenfuß, A., and Villmann, T.
  In *Proceedings of the 5th Int. Workshop on Self-Organizing Maps (WSOM 2005)*
  pp. 275–282, 2005

Chapter 2

Introduction to Topographic Mapping

A great challenge today, arising in many fields of science, is the proper mapping of datasets to explore their structure and gain information that otherwise would remain concealed, buried due to the high-dimensionality of the data. In this chapter we are concerned with the principles of topographic mapping that provides solutions to this problem.

Usually, a *topographic mapping* is defined between a high-dimensional input space and a low-dimensional map space. The mapping is expected to preserve neighbourhoods to a considerable extent, that is neighbourhoods from input space shall be mapped to neighbourhoods in map space, and vice versa. This informal description may be sufficient for the present to follow the discussion, later on we shall discuss a more formal framework.

Since the map space is low-dimensional, the mapped data can be visualized easier and often explored better than in the original space. Moreover, the map space often contains a discrete map structure on which the input data is mapped what leads to particular comfortable and plastic visualizations.

We start by considering some very popular projection techniques that do not exactly focus on a topology-preserving mapping of data but nevertheless shall give a first impression of how data can be projected — and last but not least they are presented here for historical reasons.

Principal Component Analysis One of the most popular and successful projection technique today is surely the *Principal Component Analysis* (PCA) that dates back to the early work of Pearson (1901) and Hotelling (1933). As the name suggests, PCA's objective is to determine the most important directions of a given Euclidean dataset in terms of the variance. The method is based on a linear projection to a subspace using the covariance matrix of data as the linear function that maps data to the corresponding subspace spanned by its eigenvectors. That way, the variance of the dataset is optimally preserved by the mapping.

Naturally, by further projecting to the subspace spanned by only a few eigenvectors corresponding to the largest eigenvalues, one gets the desired dimension reduction. Note that in this way also the pairwise distances between the data points are preserved as good as possible in the projection (cf. Gower, 1966). PCA can also be linked to the field of neural networks, where Oja's rule (Oja, 1982, 1992, 2008) is a mathematical formalization of Hebbian learning for a single neuron that

was shown to perform a principal component analysis on the input data after convergence. The converged neuron projects input data to the subspace spanned by the largest eigenvector of the covariance matrix.

Obviously, the drawback of PCA is its linearity. PCA is not able to provide topology-preserving maps for non-linear datasets and is therefore not suited for topographic mapping. Most of the times one will end up with meaningless results after projecting non-linear data — a situation that practitioners usually face when messing around with real data. Despite this fact, PCA enjoys great popularity in practice as attested by the numerous publications of the last 40 years from various fields, for an overview see e.g. (Jolliffe, 2002).

A generalization of PCA that overcomes the limitations of linearity are the *Principal Curves* and *Principal Surfaces* which provide approximations of data manifolds by parameterized smooth curves and surfaces (Hastie, 1984; Hastie and Stuetzle, 1989). The projection of the data onto the parametric curve or surface is the best approximation in sense of the mean squared error. Unfortunately, there is no general method to learn the parametric representation of higher dimensional data (Chang and Ghosh, 1999). The original definition of Principal Surfaces has also limitations, e.g. problems with self-intersecting data, for an overview see (Ozertem, 2008). Chang and Ghosh (2001) proposed a unifying probabilistic model that overcomes several issues.

Multidimensional Scaling In contrast to PCA where vectorial data from a Euclidean space is given, the *Multidimensional Scaling* (MDS) techniques are concerned with pairwise distances between data objects. Their objective is to find suitable points in a Euclidean space, corresponding to the given pairwise distances, such that those distances are preserved as good as possible by the Euclidean points. Obviously, most of the times some distortion will occur, because in general arbitrary distances are not isometrically embeddable into Euclidean spaces (Indyk and Matoušek, 2004), and even if they are, the dimension of the chosen Euclidean space might be too low.

There are many different approaches of MDS, for an excellent overview see (Borg and Groenen, 2005). However, these various variants can be classified into three groups: *Classical Scaling* (Torgerson, 1952) the original technique is strongly related to the above introduced PCA as shown by Gower (1966). Essentially, it is a linear model and features the same outcome as PCA when applied to Euclidean distances. Another class is formed by *Metric MDS* techniques (Torgerson, 1958; Sammon, 1969) which rely on some cost function measuring the distortion error of the embedding. These techniques therefore try to preserve the (exact) distances. In contrast, methods from the third class, the *Non-Metric MDS* techniques (Kruskal, 1964), do not focus on the distances itselves they rather try to keep the ranking of distances.

Once again, the focus of multidimensional scaling techniques is not explicitly topology-preservation. Their objectives are preservation of distances or preservation of ranks regarding distances. Nevertheless, sometimes later on in the experiments we shall rely on MDS to visualize and explore representative objects that are given only by their pairwise distances in between.

In the next section, we are concerned with the basics of prototype-based methods. These methods are based on representative prototypes that are located in data space with the aim to characterize a given data distribution as good as possible in terms of an error measure. Since prototypes can serve as average representatives for surrounding data points, the prototype-based methods are very intuitive, practi-

tioners are able to investigate important characteristics of the data in certain regions of the data space by analyzing the corresponding representatives. All topographic mapping techniques presented later will therefore be based on prototypes. Having justified our decision to focus exclusively on prototype-based methods, we will now give a brief introduction about what they are. Moreover, we will introduce two important prototype-based methods, namely *Self-Organizing Maps* and *Neural Gas*, which shall play a central role throughout the remaining chapters.

2.1 Prototype-based Methods

Throughout this work, we are exclusively concerned with methods relying on representative prototypes which are situated within data space in a way that they approximate a given data distribution best. These prototype-based approaches offer very intuitive learning techniques and their outcomes are meaningful in a way that they can be interpreted and visualized directly. Practitioners like e.g. biologists, medical scientists, or social scientists are often interested in prototypical individuals for further analysis which bear characteristical properties of the data. In this spirit, prototype-based approaches are superior to any other technique and the right tools to choose. This shall be our motivation to concentrate on these fruitful approaches leaving aside other models.

Historically issues of that kind were dealt with as part of a far more general framework in quantization theory (Gersho and Gray, 1992). That's why we will rely on some nomenclature, definitions, algorithms, and results from that field in the following. However, the objectives in quantization theory are often different from those in machine learning what justifies an independent treatment of the topic. Only the basics can be transferred, the bigger part has to be built on top anew. A very recommended reading about all the different facets of quantization theory is the survey of Gray and Neuhoff (1998). In what follows, we give some basic definitions borrowed from quantization theory and introduce the concept of prototype-based representation in conjunction with central definitions.

Regarding the representation of data by prototypes, the quality of the representation is usually measured in terms of the *quantization error* that subsumes the quadratic distances of the data to the nearest prototype. More precisely, let the data be given by a probability distribution over a manifold $V \subseteq \mathbb{R}^d$, described by the probability density function p. Given a finite collection $W = (w_i)_{i\in\{1,\ldots,n\}}$ from $\mathbb{R}^d$ of representative *prototypes*, the prototypes are requested to characterize the data manifold best when measured by the quantization error

$$Q(W) = \frac{1}{2}\sum_{i=1}^{n}\int_V \chi_i(W,v)\cdot\|w_i - v\|^2\, p(v)\, dv, \tag{2.1}$$

where $\chi_i(W,v)$ is one iff $v \in \{x \in \mathbb{R}^d \,:\, \|w_i - x\| = \min_{w_j\in W}\|w_j - x\|\}$, and zero else. In this context the set of prototypes W is also called a *vector quantizer*. Note that once in a while in the theoretical discussion, we shall also consider countable sets of prototypes.

For a finite set of data points $V = \{v_1, v_2, \ldots, v_m\}$, drawn with respect to the underlying probability distribution, the quantization error is estimated by the *intra-class variance*

$$\hat{Q}(W) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\chi_i(W,v_j)\cdot\|w_i - v_j\|^2, \tag{2.2}$$

and we shall also refer to the intra-class variance as the (empirical) quantization error (Graf and Luschgy, 2002) from now on.

The prototypes induce a *Voronoi diagram* $\{V_i(W) : w_i \in W\}$ of $\mathbb{R}^d$, where $V_i(W) = \{x \in \mathbb{R}^d : \|w_i - x\| = \min_{w_j \in W} \|w_j - x\|\}$ is the *Voronoi region* generated by $w_i \in W$. By our definition, the points on a shared border of adjacent prototypes belong to all participating prototypes. The Voronoi diagram is a covering of $\mathbb{R}^d$, that is $\cup_{w_i \in W} V_i = \mathbb{R}^d$. Note that the Voronoi region of a prototype is also called its *receptive field* for historical reasons.

We further define the *winner index* $s(v, W)$ to be

$$s(v, W) = \operatorname*{argmin}_i \|w_i - v\|, \tag{2.3}$$

if it is well-defined. If there are different w_i with the same distance to v, ties shall be broken deterministically, e.g. by consulting the natural order of the indices. Here and elsewhere we shall omit the set of prototypes W in the notation, leaving $\chi_i(v)$ and $s(v)$, when it can be done without ambiguity.

Naturally the question arises how to algorithmically determine an appropriate set of prototypes for given data. In 1957 Lloyd introduced a prototype-based algorithm minimizing the quantization error given a finite set of data that is nowadays one of the most popular methods operating under the name *k-means*. After randomly initializing the prototypes, it determines in every iteration which points are located inside the Voronoi region of each prototype and moves the corresponding prototype to the barycenter of those points. These two operations are repeated until a stopping criterion holds. Interestingly, the original method was only published as a technical report first, the official journal publication follows not until 1982, see (Lloyd, 1982). Meanwhile, numerous extensions and variations of k-means have been published, milestones often cited are e.g. a special online variant by MacQueen (1967), a variant particularly efficient for clustering Hartigan and Wong (1979), and k-means with a growing set of prototypes doubling the number in each step by Linde et al. (1980). All these k-means derivates determine a set of prototypes, but all of them share the common drawback that they easily get stuck in local optima of the cost function they try to optimize. Furthermore, closely linked to this disadvantage is their sensitivity to initialization of the prototypes. Thus, they often provide only suboptimal solutions in form of prototypes not faithfully representing given data. Later on, we shall introduce alternative methods overcoming these issues.

Having introduced the basic concepts how to represent data by prototypes in input space, we are now concerned with a first – and perhaps the most popular – prototype-based technique of topographic mapping, the *Self-Organizing Map*. The representation of data is no longer done only by prototypes situated in input space, the data is mapped to prototypes in a lower dimensional structure hopefully carrying over much topological information.

2.2 Self-Organizing Maps

Motivated by biological self-organization processes that emerge in neural cell structures, Kohonen (1982, 2001) introduced the concept of *Self-Organizing Maps* (SOM), modeled as a special artificial neural networks in conjunction with a heuristical learning rule. Essential part of the concept is the *map*, a finite structure with a fixed topology whose elements shall serve as representatives of the data later on. The

term *self-organization* refers to the ability of the method to reorganize its initial map and adapt it to the topological characteristics of given input data autonomously. Each element of the map is assigned to a prototype located in input space which is updated in a vector quantization style but under influence of the neighbourhood structure dictated by the map. Those assignments provide the connections between each element of the map and a corresponding Voronoi region of the input space. By transferring collective properties of the data within the Voronoi regions in form of e.g. labels, elements of the map can reflect characteristics of certain regions of the input space, providing a low-dimensional visualization of high-dimensional data.

Despite its simple and intuitive principle, the Self-Organizing Map has shown a robust performance in practice and has proved to be well-suited for many applications. It has raised great interest in different communities and started a large flow of successive literature. For an extensive collection of over 7000 references see e.g. the *Bibliography of Self-Organizing Map (SOM) Papers* (Kaski et al., 1998; Oja et al., 2003; Pöllä et al., 2007).

Note that Self-Organizing Maps can also be interpreted as discrete approximations of Principal Surfaces (cf. page 6) as pointed out by Mulier and Cherkassky (1995). A related concept are the Elastic Nets, see (Willshaw and von der Malsburg, 1976; von der Malsburg and Willshaw, 1977; Durbin and Willshaw, 1987), that show similar states of convergence but their ordering process is different (Claussen and Schuster, 2002).

As mentioned before, the Self-Organizing Map is based on a predefined map structure with a fixed topology. We refer to this structure in the following as a *lattice*. The lattice is usually chosen as a simple, low-dimensional structure, like a rectangular grid, a torus, or a hexagonal grid, to keep the advantage of easy visualization. Also more advanced structures like periodical grids in hyperbolic space featuring an exponential growth of elements are sometimes considered (cf. Ritter, 1999) but require proper visualization techniques for non-Euclidean structures.

Besides grids also other shapes are possible as lattice structures, e.g. trees are used in the TreeSOM approach (Sauvage, 1997). The lattice can also grow during processing, see the Growing SOM (Bauer and Villmann, 1997) and the Growing Hierarchical Tree SOM (Forti and Foresti, 2006).

The lattice structure needs not to be constructed explicitly for training. We only need to know the pairwise distances between the lattice elements induced by the neighbourhood structure. Here, usually the shortest path distance is chosen, i.e. the shortest path length in a graph-theoretical sense where every arc often has length one by definition.

More precisely, abstracting from any biological and neural point of view, we describe the Self-Organizing Map method as follows:

Formal Description of Self-Organizing Maps

Assume that there is input data given by a probability distribution over a manifold $V \subset \mathbb{R}^d$ that is described by a probability density function p. As a first ingredient, we introduce a finite collection of prototypes $W = (w_i)_{i \in \{1,\dots,n\}}$ from $\mathbb{R}^d$ that serves as a vector quantizer in input space. The second component is a lattice structure featuring the same number of elements as W. We identify the elements of the lattice structure with the prototypes of W and denote them also as $w_1, w_2, \dots, w_n$, what can be done formally by a bijective mapping. In doing so, each prototype is in a way distinctly connected to an element of the lattice, and vice versa. Since we are only interested in the pairwise distances of lattice elements, let $g(w_i, w_j)$ denote the pairwise distance of elements w_i and w_j in the given lattice structure.

One can imagine the training process of the method as a simultaneous vector quantization *and* non-linear projection of the sampled input data whereas both techniques influence each other coupled by the connection of prototypes and lattice elements. More formally, the online training algorithm of a Self-Organizing map is given as follows:

Training of Self-Organizing Maps The prototypes $W = (w_i)_{i \in \{1,\dots,n\}}$ from R^d are initialized randomly in input space and are updated in each epoch by the rule

$$\Delta w_i \;=\; \varepsilon_t \cdot h_\lambda(g(w_{s(v)}, w_i)) \cdot (v - w_i) \tag{2.4}$$

for all prototypes $w_i \in W$. Thereby, $g(\cdot,\cdot)$ denotes the pairwise distances in the lattice and $w_{s(v)}$ denotes the winning prototype for the sampled data point v as defined in (2.3). The parameter $\lambda > 0$ controls the *neighbourhood range* through the *neighbourhood function* $h_\lambda(x) = \exp(-x/\lambda)$. The update is governed by a decreasing *step size* $\varepsilon_t \in (0, 1]$ that usually obeys the condition $\sum_{t=1}^{\infty} \varepsilon_t = \infty$ and $\sum_{t=1}^{\infty} \varepsilon_t^2 < \infty$, following the work of Robbins and Monro (1951) who introduced similar conditions in their work on stochastic approximation.

During processing the initially large neighbourhood range λ is driven asymptotically to zero changing the characteristics of the updates in every step.

In what follows, we intent to give a brief summary of the theoretical properties that were derived for Self-Organizing Maps. Unfortunately, despite its algorithmic simplicity and intuitivity, the method has shown to be quite resistant against a mathematical treatment. For an overview of the latest achievements see (Cottrell et al., 1994, 1998; Hammer and Villmann, 2003; Fort, 2006). Briefly, the state-of-the-art is a broad theoretical understanding of the one-dimensional case but only few theoretical results about the multi-dimensional case are known.

Theoretical Properties of Self-Organizing Maps Due to the dynamics of λ, the Self-Organizing Maps show two phases during training (cf. Sadeghi, 2001). In a first phase, the *self-organization phase*, the initially disordered prototypes and therefore also the disordered map elements become in some way organized reflecting topological properties of the input space. In the second phase, the *convergence phase*, the prototypes converge to their final destinations driven in a fashion of a special case of the Robbins-Monro algorithm (cf. Sadeghi, 1998), but for finite datasets they do not minimize exactly the quantization error as pointed out by Rynkiewicz (2006) based on previous work by Fort and Pagès (1995, 1996).

However, it must be stated here that almost all theoretical results about SOM were derived only for the one-dimensional case, i.e. one-dimensional input space and a one-dimensional lattice in shape of a chain. These results transfer also to settings where lattice space and data space are separable, i.e. the distributions are independent for each dimension. There is not much theoretical evidence about the general multidimensional case. Complete proofs of convergence in the general case are not known so far.

Moreover, it has been shown that in general for a continuous input distribution there does not exist any cost function that is minimized by the Self-Organizing Map in a gradient descent fashion (cf. Erwin et al., 1992).

Heskes and Kappen (1993); Heskes (1996, 1999) demonstrated for the general case that by slightly changing the winner definition in the update rule (2.4), nominating a prototype as winner that minimizes the weighted average distance (also

referred to as the *local error*) to the sampled data point instead, that is

$$s^*(v, W) \;=\; \underset{i=1}{\overset{n}{\operatorname{argmin}}} \sum_{k=1}^{n} h_\lambda(g(w_i, w_k)) \|w_k - v\|^2 \,, \tag{2.5}$$

the modified Self-Organizing Map performs a stochastic gradient descent on the cost function

$$E_{\mathrm{SOM}}(W) = \frac{1}{2} \sum_{i=1}^{n} \int h_\lambda(g(w_{s^*(v)}, w_i)) \cdot \|w_i - v\|^2 \, p(v) \, dv. \tag{2.6}$$

Finishing this brief summary of the theoretical properties of SOM, it should be mentioned here that despite the lack of theoretical evidence Self-Organizing Maps perform very robust and reliable in practical applications, surely one good reason for their popularity in numerous fields (Kohonen, 2001).

It should also be noted that Bishop et al. (1998a,b) proposed a concurrent method, the *Generative Topographic Mapping* (GTM). GTM is a probability density model which describes the data distribution by latent variables utilizing a grid in a low dimensional latent space and a set of non-linear basis functions that are combined in data space governed by an adapted transformation. The model is a probabilistic reformulation of Kohonen's SOM appropriate to overcome disadvantages of the Self-Organizing Map. The GTM model comes up with an explicit continuous density model of the input data not only a discrete one as in case of SOM, a provable convergence in contrast to SOM, and a cost function signalizing the quality of the map. Nevertheless, a decade after its introduction the GTM has not yet displaced the original Self-Organizing Map, maybe owing to its less intuitive principle. The intention of our thesis is topographic mapping by prototype-based methods, so we leave it at that brief discussion of GTM.

In the following section, we are concerned with the topology preservation property of Self-Organizing Maps, a crucial aspect regarding the visualization ability of a projection method.

Topology Preservation in Self-Organizing Maps

One of the most important issues concerning SOM is the topology preservation feature that is needed for a meaningful projection of (intrinsic) high-dimensional input data into the lower-dimensional lattice space. Since in general the projection will take place from a higher to a true lower-dimensional space, naturally some topological information will be lost in the process due to disrupted neighbourhoods, or new neighbourhoods will be created where there were none before. This is usually referred to as the *dimension conflict*. But also a mismatch in the topology of data manifold and the fixed lattice structure will impose errors. As discussed by Venna and Kaski (2006) these kinds of errors influence the measures *trustworthiness* and *discontinuity* of the map. Obviously, there is a tradeoff between those measures, thus projection methods cannot optimize both objectives at the same time.

For analyzing the quality of the mapping in terms of topology preservation, one has to define appropriate measures in a mathematically sound way. This issue is not trivial, there were various formal frameworks proposed measuring the topology preservation of a Self-Organizing Map, for a comprehensive overview see (Bauer et al., 1999; Villmann, 2004). The most promising approach is the *topographic product* (Bauer and Pawelzik, 1992) defined on rectangular and hexagonal lattice

structures and the more general *topographic function* (Villmann et al., 1997; Villmann, 1997).

In the context of Self-Organizing Maps, we can formally define a *topology preserving mapping* as a pair of suitable forth and back mappings between data and lattice space that are both continuous in the topological sense, i.e. they are neighbourhood preserving. If a topology preserving mapping between a data manifold and a lattice structure exists, we call the lattice a *topology preserving map* of the manifold. Later on, we will also refer to this kind of lattice as a *perfectly* topology preserving map to emphasize its ultimate perfection in contrast to topographic maps with some topological defects that shall be discussed later. For a formal discussion of topology preservation we need to define suitable neighborhoods on data and lattice spaces to obtain topological spaces.

Topographic Function To measure the quality of a topographic map provided by a SOM, Villmann et al. (1997) considered a discrete lattice space and the discrete space of prototypes. They defined appropriate metrics on lattice and prototype space which induce suitable topologies. Between these discrete topological spaces a mapping Ψ was defined in the following way: The forth mapping assigns input data $v \in V$ to the lattice element corresponding to its winning prototype $w_{s(v)}$. It is considered *neighborhood preserving* iff the images of adjacent points $v_i, v_j \in V$ are also adjacent according to the Chebyshev distance in the lattice. The back mapping, i.e. the bijection between lattice elements and prototypes assigning each lattice element to its corresponding prototype in input space, is said to be *neighborhood preserving* iff images of adjacent elements of the lattice measured by the Euclidean distance are also adjacent in data space.

Based on these metrics inducing the topological spaces, the *topographic function* Φ was introduced that provides a measure of the degree of topology preservation of the mapping Ψ. The definition relies on the existence of an induced Delaunay triangulation of the prototypes in input space what presumes some kind of density criterion to be fulfilled that is discussed later on in more detail.

The topographic function Φ depends on a parameter that ranges in the interval $[-(n-1), n-1]$ of the number of prototypes n, with a special meaning for $\Phi(0)$. With help of the topographic function it is now possible to quantify the quality of the topographic mapping. If it holds $\Phi(0) = 0$, the mapping is perfectly topology preserving. The largest value k^+ for which $\Phi(k^+) \neq 0$ and the smallest value k^- for which $\Phi(k^-) \neq 0$ are indicators to distinguish local and global dimensional conflicts (cf. Villmann, 2004).

However, Fort (2006) pointed out that by then all proposed measures were based on a discrete setting utilizing a topology based on Voronoi tesselations. He also stated that proving results about self-organization seemed to be very difficult in the discrete framework, instead, an asymptotic framework was suggested and briefly sketched.

Concerning the above introduced SOM variant of Heskes and Kappen (1993) that performs a gradient descent on a cost function, Heskes (1996) analysed the average transition time from disordered maps to fully ordered configurations in the modified SOM variant. In this context, it was pointed out that topological defects of the map, like kinks in one-dimensional maps or twists in two-dimensional maps, correspond to local minima of the error potential, whereas the global minima corresponds to a perfectly ordered configuration. That way, topological conflicts are observable in this variant, provided that there is no conflict of intrinsic dimension-

ality in principle.

As discussed above, one serious drawback of Self-Organizing Maps is their fixed lattice structure that is very well suited for visualization but lacks of flexibility in terms of topology. Another type of methods trades easy visualization in for a more flexible lattice structure that is driven by data and not fixed beforehand. The approach is based on so called *Topology Representing Networks* which shall be discussed in the following section.

2.3 Topology Representing Networks

In this section, we briefly introduce a topographic mapping approach (Martinetz and Schulten, 1994) that in contrast to Self-Organizing Maps does not depend on a predefined fixed lattice structure. It constructs its lattice structure according to the given data topology by applying a competitive Hebbian scheme.

More precisely, let $M \subseteq \mathbb{R}^d$ be a manifold which we suppose for the present to be given explicitly. Suppose further that there is a finite collection of prototypes $W = (w_i)_{i\in\{1,\ldots,n\}}$ from $\mathbb{R}^d$ lying in the manifold. We shall show later how prototypes can be distributed homogeneously over a manifold. We name $V_i^{(M)}(W) = V_i(W) \cap M$ the *masked Voronoi region* of prototype w_i. Because of $w_i \in M$ and of course $w_i \in V_i(W)$ the intersection is never empty. Moreover, since the Voronoi diagram $V(M)$ covers $\mathbb{R}^d$, the masked Voronoi diagram covers M, i.e. it is $M = \bigcup_{i=1}^n V_i^{(M)}(W)$.

We define a neighbourhood relation of prototypes on the manifold by a non-empty intersection of the corresponding masked Voronoi regions. That way, a masked Voronoi diagram generates a so-called *induced Delaunay triangulation*, a graph structure on W that is given by the adjacency of the masked Voronoi regions, i.e. there is an edge $w_i \overset{e}{\text{---}} w_j$ iff $V_i^{(M)}(W) \cap V_j^{(M)}(W) \neq \emptyset$. As shown by Martinetz and Schulten (1994, Theorem 2), the induced Delaunay triangulation is a perfectly topology preserving map of M. That means, neighbourhoods in the manifold are mapped to neighbourhoods in the graph structure, and vice versa. It is to be understood here, that obviously the larger the number of prototypes the finer is the resolution of the map. Note that by definition, also one single prototype $w_i \in M$ is a perfectly topology preserving map!

At this point the question arises how to determine the induced Delaunay triangulation algorithmically. This is not as easy as the construction of the standard Delaunay triangulation via the Voronoi regions. Usually, the manifold M is only accessible by a probability distribution defined on $\mathbb{R}^d$, i.e. we can only sample its points. Martinetz and Schulten (1994) therefore proposed an iterative Hebbian learning scheme that always strengthen the connections between first and second winner prototype for each sampled point.

Martinetz and Schulten (1994) defined prototypes $W \subseteq M$ to be *dense* in M if for every $v \in M$ the triangle formed by v and the first and second winner prototype regarding v lies completely in M, i.e. $\Delta v w_{s(v)} w_{s'(v)} \subseteq M$. For instance, in convex datasets the prototypes are always dense according to the above definition. Moreover, it was shown that the generated connective structure of a TRN converges to the induced Delaunay triangulation if the prototypes are lying dense in the manifold. The resulting graph structure was also shown to be path-preserving in the manifold M, i.e. two points w_i, w_j are connected by a path in the graph structure if and only if they are connected by a path in the manifold M.[1]

[1] A *path* in a manifold X from x_1 to x_2 is a continuous map $\phi : [a, b] \to M$ such that $\phi(a) = x_1$ and $\phi(b) = x_2$.

Thus, under the constraint that the prototypes obey a density criterion regarding the underlying data manifold, applying a competitive Hebbian scheme by always connecting the first and second ranked prototypes leads to a graph structure that represents the induced Delaunay triangulation of the data manifold. That way, a perfect topology preservation is obtained. The outcoming lattice can be seen as a discrete, path preserving representation of the data manifold. In this context, we call that approach a *topology representing network*.

Martinetz and Schulten (1994) discussed loosely in this context that a homogeneous distribution of prototypes can be made dense by increasing the number of prototypes. Obviously, if the data manifold M has a smooth boundary ∂M, e.g. it is convex, a sufficiently large number of prototypes will be dense in M. Note that this aspect is of great importance for the algorithmic usage, later on we will present a technique capable of generating a homogeneous distribution of prototypes and therefore constituting the foundation of the discussed approach.

It should be noted here that there is a related framework introduced by Fritzke (1995) and Bruske and Sommer (1995) which relies on a growing strategy. The number of prototypes is increased by time and topological connections are drawn using the same Hebbian scheme as above. Connections as well as prototypes fade away during the iteration process if they no longer match the topology best. Later on Fritzke (1997) applied his approach also successfully to the processing of non-stationary distributions. For some of the latest developments in this direction, namely the Growing Hierarchical Tree SOM, see also (Forti and Foresti, 2006).

One important question concerning the usability of the Topology Representing Network approach has remained open: We haven't discussed yet how to get suitable prototypes lying in the manifold and capturing the subtleties of the topology. Fortunately, there is a reliable vector quantization technique distributing its prototypes exactly as desired, the so-called *Neural Gas* method. In what follows we will introduce the principles of Neural Gas.

2.4 The Neural Gas Algorithm

As motivated in the last section, the foundation of Topology Representing Networks are prototypes which lie dense in the data manifold. As stated by Martinetz and Schulten (1994), a homogeneous distribution of prototypes can always be made dense by simply increasing the number of prototypes. For that reason, methods are sought after that are capable of distributing prototypes homogeneously in the data manifold. In the following, we will present the very robust and reliable vector quantization method *Neural Gas* that can serve as a basis for Topology Representing Networks (Martinetz and Schulten, 1994). Besides being part of Topology Representing Networks, it also offers great advantage over the popular k-means algorithm and can be used as a substitute (Martinetz et al., 1993).

Neural Gas (NG), introduced by Martinetz et al. (1993), is a vector quantization technique that aims to construct a finite collection $W = (w_i)_{i\in\{1,\ldots,n\}}$ of representative prototypes from $\mathbb{R}^d$ for a given data manifold $M \subset \mathbb{R}^d$. Once in a while in the theoretical discussion, we shall also consider countable sets of prototypes.

In the following, let the data be given by a probability distribution over a manifold $V \subseteq \mathbb{R}^d$, described by the probability density function p. The prototypes are then requested to characterize the data manifold best when measured by the quantization error (2.1). Technically, Neural Gas utilizes a stochastic gradient descent on a cost function that is based on a ranking scheme, relating the strength of pro-

totypes updates to their ranks regarding the distances to the sampled data point. Due to its importance for the theory of Neural Gas and all upcoming parts of this work, we would like to emphasize the following definition of ranks:

Definition 2.4.1 (Ranking of Prototypes) Let $W = (w_i)_{i\in\{1,\dots,n\}}$ from $\mathbb{R}^d$ be a finite collection of prototypes. The *rank* $k_i(W, v)$ of a prototype w_i relative to data point $v \in \mathbb{R}^d$ is given by the *rank function* $k_i : \left(\bigtimes_{i=1}^{n} \mathbb{R}^d\right) \times \mathbb{R}^d \to \{0, 1, \dots, n-1\}$ defined by

$$k_i(w_1, w_2, \dots, w_n, v) = |\{w_k \,:\, \|w_k - v\| < \|w_i - v\|\}|. \tag{2.7}$$

Moreover, we require the function to be bijective. If necessary, ties $\|w_k - v\| = \|w_i - v\|$ shall be broken deterministically. For convenience, we denote also the rank function as $k_i(W, v)$ in the following. □

We would also like to emphasize the importance of Neural Gas for the thesis at hand. It shall serve us as a reference throughout in a way that new techniques are introduced and discussed on its basis. In the majority of cases, these introduced techniques are easily transferable to Self-Organizing Maps or k-Means due to the close relationship between the models. So most of the times, we present only the Neural Gas variant of a technique and settle back by simply referring to the analogy of the procedure. Having set the further course of action, we present the Neural Gas algorithm in the following.

Neural Gas Algorithm

For input data given by a probability distribution that is described by a probability density function p, Neural Gas performs a stochastic gradient descent on the cost function

$$E_\lambda(W) = \sum_{i=1}^{n} \int h_\lambda(k_i(W, v)) \cdot \|w_i - v\|^2 \, p(v) \, dv, \tag{2.8}$$

through the update rule

$$\Delta w_i \;=\; \varepsilon_t \cdot h_\lambda(k_i(W, v)) \cdot (v - w_i) \tag{2.9}$$

for all prototypes $w_i \in W$. The parameter $\lambda > 0$ controls the *neighbourhood range* through the *neighbourhood function* $h_\lambda(x) = \exp(-x/\lambda)$. The update is governed by a decreasing *step size* $\varepsilon_t \in (0, 1]$ that usually obeys the condition $\sum_{t=1}^{\infty} \varepsilon_t = \infty$ and $\sum_{t=1}^{\infty} \varepsilon_t^2 < \infty$, following the work of Robbins and Monro (1951) who introduced similar conditions in their work on stochastic approximation.

During processing the initially large neighbourhood range λ is driven asymptotically to zero changing the characteristics of the cost function in every step. Due to the dynamics of λ, Neural Gas is not sensitive to initialization, and most local optima are supposed to arise only later in the process. Therefore the neighbourhood dynamics prevents the method from getting stuck too early in suboptimal states. Note that for a vanishing neighbourhood range the cost function Eq. (2.8) becomes the quantization error (2.1), since $C(\lambda) \to 1$ and $h_\lambda(k_i(W, v)) \to \chi_i(W, v)$ hold for $\lambda \to 0$.

For all experiments in this work the initial neighborhood range λ_0 was chosen as $n/2$ as suggested by Martinetz et al. (1993), where n is the number of neurons used. The neighborhood range λ_t was decreased exponentially with the number of

adaptation steps t according to $\lambda_t = \lambda_0 \cdot (0.01/\lambda_0)^{t/t_{\max}}$ as it was suggested in the original work of Martinetz et al. (1993). The value $t_{\max}$ is the number of epochs of a training run.

In what follows we are concerned with an inherent but counterintuitive characteristic of vector quantizers. The final prototypes do not exactly reproduce the input data density, instead an asymptotic power law holds. This inherent characteristic can pose undesired effects in practical applications and has therefore to be kept in mind while applying vector quantizers.

2.5 The Magnification Effect

It was first demonstrated by Zador (1963, 1982) and stated more precisely by Graf and Luschgy (2000) that vector quantization techniques aiming for a minimization of the distortion error feature the inherent characteristic that the final prototype density ϱ does not exactly match the data density p. Instead, the relation between those densities asymptotically obeys the power law

$$\varrho(w) \sim p(w)^{\alpha}, \tag{2.10}$$

with $\alpha \neq 1$ in general. In this context, the exponent α is called the *magnification exponent*. In a more general setting, the magnification characteristic of a certain method is called the *magnification factor*.

The magnification effect might seem counterintuitive at first feel, but it can be easily observed in experiments when sparsely sampled regions of the input space with low probability draw prototypes from dense regions. This is particularly striking in case of outliers in real data.

In general, the magnification effect is an inherent characteristic of the different methods and not only vector quantizers suffer from this discrepancy. Although, for a variety of popular methods, the magnification follows a power law with exponent different from one (see Villmann and Claussen, 2006).

As it was shown by Zador (1982) in a more general setting, for vector quantizers minimizing the quadratic distortion error Eq. (2.1), like the popular k-means, it holds

$$\alpha = \frac{\hat{d}}{\hat{d}+2}, \tag{2.11}$$

where $\hat{d}$ denotes the intrinsic data dimensionality (see also Graf and Luschgy, 2000).

Martinetz et al. (1993) proved that for a small neighbourhood range $0 < \lambda \ll n$ the Neural Gas algorithm also features a magnification exponent of $\alpha = \hat{d}/(\hat{d}+2)$, what is consistent with the result of Zador (1982) in (2.11).

A magnification factor of one relates to an optimal information transfer and yields maximum entropy of the mapping in an information-theoretical sense. The prototypes are then exactly adjusted according to the underlying data distribution – the data density equals the prototype density. In that case, the amount of information, which is conserved replacing the points in the receptive fields by their representative prototypes, is maximized.

It also means that all prototypes have the same probability over the data distribution to become winner. For a given data distribution P and magnification factor

one, it then holds

$$\lim_{|W|\to\infty} \left| \int_{V_i(W)} v\, dP(v) \; - \; \int_{V_j(W)} v\, dP(v) \right| \; = \; 0 \quad \text{for all } w_i, w_j \in W,$$

where $V_i(W)$ and $V_j(W)$ are the corresponding Voronoi regions of the final prototype locations $w_i, w_j \in W$ after learning, respectively.

A magnification factor one is specific for approaches explicitly optimizing the information transfer or related measures (cf. Linsker, 1989).

The reader might be in doubt about the impact of the magnification effect in real-life data, because for high-dimensional data, Neural Gas is approximately information optimal, since $d/(d+2) \to 1$ for $d \to \infty$. But almost always the intrinsic dimensionality of real data is very low, i.e. $\hat{d} \ll d$. So there is a good justification to deal with the magnification effect in the real world.

Later on, subsequent to the presentation of each upcoming method, we will discuss possibilities to modify the methods in a way that the magnification exponent changes. That results in an altered distribution of the final prototypes and opens the field of *magnification control* allowing for arbitrary control of the magnification.

As discussed above, Neural Gas tends to shift prototypes towards regions where data is sparsely sampled. This inherent behaviour might lead to unwanted effects if, for example, unwanted outliers occur in the dataset (Hodge and Austin, 2004). Here, magnification control can help to suppress the influence of the outlying data points.

In practice also the opposite effect is of great interest. For example, if the focus lies on rare events that should be covered by prototypes (Merényi and Jain, 2004; Villmann et al., 2003; Villmann and Heinze, 2000), magnification control allows to emphasize regions of low density.

In the next chapter, we will discuss how to accelerate the introduced prototype-based methods and also how to apply them efficiently on very large datasets.

Chapter 3

Fast Topographic Mapping of Euclidean Data

This chapter is concerned with fast variants of prototype-based methods that can be applied if the given Euclidean dataset is finite. They show a very fast convergence compared to the original formulations and can be used as a replacement in all applications where the whole dataset is available in advance. Later on, we will discuss also variants that are capable of dealing with very large datasets, and also with streaming data or datasets that are gathered over time.

The proofs and concepts in this section shall play an important role for this work, because all following parts are based on quite similar reasonings and can be traced back to the proofs and concepts presented here. That's why we will give more details here and keep things shorter later on by referencing to this chapter.

3.1 Batch Processing for Prototype-based Methods

The original Neural Gas method optimizes its cost function by applying a stochastic gradient descent method, that in principle needs many iterations until convergence (Martinetz et al., 1993). However, if a finite dataset $\{v_1, v_2, \ldots, v_m\}$ is given, Neural Gas can be formulated as a faster batch variant as introduced by Cottrell et al. (2005, 2006).

In the finite setting, the cost function of Neural Gas Eq. (2.8) becomes

$$E_\lambda(W) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot \|w_i - v_j\|^2 . \tag{3.1}$$

In every iteration of a batch approach, all prototypes are updated at once taking into account the whole dataset. The optimal locations of the prototypes must therefore be calculated in an efficient manner, that's why we are looking for an analytic solution.

Obviously,

$$\frac{\partial}{\partial w_i} E_\lambda(W) = \sum_j h_\lambda(k_i(W, v_j))(w_i - v_j) \stackrel{!}{=} 0$$

cannot be solved explicitly because of the rank function. As a consequence, there is no way to directly derive update rules for the optimal prototype locations.

For this reason, we introduce a set of hidden variables k_{ij} to replace the rank function $k_i(W, v_j)$, where $\{k_{ij} : i \in \{1, 2, \ldots, n\}\}$ is a permutation of $\{0, 1, \ldots, n-1\}$ for every $j \in \{1, 2, \ldots, m\}$. This yields

$$\hat{E}_\lambda(W, k_{ij}) = \frac{1}{2} \sum_i \sum_j h_\lambda(k_{ij}) \cdot \|w_i - v_j\|^2 . \tag{3.2}$$

That way, an alternating optimization technique (Hathaway and Bezdek, 2003) can be applied, that, in turn, optimizes the hidden variables k_{ij} for fixed prototype locations w_i, and then determines optimal prototype locations w_i for fixed hidden variables k_{ij}, according to the cost function Eq. (3.2).

In Table 3.1 the Batch Neural Gas algorithm is quoted. In the following, we will discuss in detail how to get the optimal assignments in each step of the alternating optimization. Also a proof of convergence of the algorithm in sense of the cost function is given.

Algorithm 3.1: Batch Neural Gas

Input

Data $V = \{v_1, v_2, \ldots, v_m\} \subset \mathbb{R}^d$

Begin

(* Initialize prototypes and neighbourhood range *)

Init $w_i \in \mathbb{R}^d$ randomly for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Compute Euclidean distances...

$d(w_i, v_j) = \|w_i - v_j\|$

Determine hidden variables as ranks (break ties deterministically)...

$k_{ij} = |\{ l \in \{1, \ldots, n\} : d(w_l, v_j) < d(w_i, v_j)\}|$

Update prototype locations...

$w_i = \sum_j h_\lambda(k_{ij}) \cdot v_j / \sum_j h_\lambda(k_{ij})$

Decrease neighbourhood range...

$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$

endfor;

(* Return representative prototypes *)

Return w_i

End.

Optimal Assignments

The first step of the alternating optimization is the determination of optimal assignments for the hidden variables k_{ij}. Given fixed prototype locations, an optimal assignment for the hidden variables k_{ij}, in terms of the modified cost function $\hat{E}_\lambda$, turns out to be the ranks of the prototypes, that means

$$k_{ij} \;=\; k_i(W, v_j), \tag{3.3}$$

since that way farther situated prototypes will get lower weights, and vice versa.

Assume W are fixed prototype locations. We say an assignment k'_{ij} is *conform with the ranks* $k_i(W, v_j)$ if there is no pair $(k'_{tj}, k'_{t'j})$ with $k'_{tj} < k'_{t'j}$ and $\|w_t - v_j\|^2 > \|w_{t'} - v_j\|^2$. Obviously, $k_{ij} = k_i(W, v_j)$ is conform with the ranks.

Proposition 3.1.1 Let W be fixed prototype locations. Then, the assignment $k_{ij} = k_i(W, v_j)$ is a global minimizer of $\hat{E}_\lambda(W, k_{ij})$, and it is unique up to conformity.

Proof. To verify the claim, we show that all conform assignments are global minimizers of $\hat{E}_\lambda(W, k_{ij})$, and that it holds $\hat{E}_\lambda(W, k_{ij}) < \hat{E}_\lambda(W, k'_{ij})$ for any assignment k'_{ij} not conform with the ranks $k_i(W, v_j)$.

We may assume, without loss of generality, that the fixed prototypes $w_1, w_2, \ldots, w_n$ are subscripted in a way that

$$\|w_1 - v_j\|^2 \leq \|w_2 - v_j\|^2 \leq \ldots \leq \|w_n - v_j\|^2$$

when considering v_j. Let $t < t'$. For any nonnegative η, η' with $\eta < \eta'$ it then holds

$$\begin{aligned} &\|w_t - v_j\|^2 < \|w_{t'} - v_j\|^2 \iff \\ &(\eta' - \eta) \cdot \|w_t - v_j\|^2 < (\eta' - \eta) \cdot \|w_{t'} - v_j\|^2 \iff \\ &\eta \cdot \|w_t - v_j\|^2 + \eta' \cdot \|w_{t'} - v_j\|^2 > \eta' \cdot \|w_t - v_j\|^2 + \eta \cdot \|w_{t'} - v_j\|^2 \end{aligned} \tag{3.4}$$

Since the set of possible assignments is finite, there must exist a global minimizer, an assignment k^*_{ij} with $\hat{E}_\lambda(W, k^*_{ij}) \leq \hat{E}_\lambda(W, k_{ij})$ for all possible assignments k_{ij}.

Let k^*_{ij} be a global minimizer of $\hat{E}_\lambda$ for fixed prototypes W. Suppose k^*_{ij} is not conform with the ranks $k_i(W, v_j)$. By definition of conformity, there must exist at least one pair $(k^*_{tj}, k^*_{t'j})$ where $k^*_{tj} > k^*_{t'j}$ and $\|w_t - v_j\|^2 < \|w_{t'} - v_j\|^2$. Let $(k^*_{tj}, k^*_{t'j})$ be such a pair and set $\eta_{ij} = h_\lambda(k^*_{ij}) = \exp(-k^*_{ij}/\lambda)$. Then, the cost function can be written as

$$\hat{E}_\lambda(W, k^*_{ij}) = \frac{1}{2} \sum_j \left[\sum_{i \neq t, i \neq t'} \eta_{ij} \|w_i - v_j\|^2 + \eta_{tj} \|w_t - v_j\|^2 + \eta_{t'j} \|w_{t'} - v_j\|^2 \right].$$

Since by assumption k^*_{ij} is not conform with the ranks due to the pair $(k^*_{tj}, k^*_{t'j})$, it holds $\eta_{tj} < \eta_{t'j}$ and $\|w_t - v_j\|^2 < \|w_{t'} - v_j\|^2$. Then, according to Eq. (3.4), we obtain

$$\begin{aligned} \hat{E}_\lambda(W, k^*_{ij}) \quad &> \quad \frac{1}{2} \sum_j \left[\sum_{i \neq t, i \neq t'} \eta_{ij} \|w_i - v_j\|^2 + \eta_{t'j} \|w_t - v_j\|^2 + \eta_{tj} \|w_{t'} - v_j\|^2 \right] \\ &=: \quad \hat{E}_\lambda(W, k^{**}_{ij}) \end{aligned}$$

Thus, there is an assignment k^{**}_{ij} with $\hat{E}_\lambda(W, k^{**}_{ij}) < \hat{E}_\lambda(W, k^*_{ij})$, namely the assignment where positions t and t' are transposed compared to k^*_{ij}. But then k^*_{ij} is not a minimizer, what leads to a contradiction. Therefore, the supposition must be false, k^*_{ij} has to be conform with the ranks. Thus, any global minimizer of $\hat{E}_\lambda$ is conform with the ranks.

Obviously, all conform assignments share the same value when mapped by $\hat{E}_\lambda$. Since there must exist a global minimizer that is always a conform assignment, by definition, all conform assignments are global minimizers of $\hat{E}_\lambda$. In particular,

the assignment $k_{ij} = k_i(W, v_j)$ is a global minimizer of the function $\hat{E}_\lambda$ for fixed prototypes, and it is unique up to conformity. □

If ties in the rank determination are broken deterministically, then $k_{ij} = k_i(W, v_j)$ is the only conform assignment, and therefore a *unique global minimizer* of $\hat{E}_\lambda(W, k_{ij})$ for fixed W.

Optimal Prototypes

The second step of the alternating optimization is the determination of optimal prototype locations. It turns out that the optimal prototype locations W for fixed hidden variables k_{ij} are given by the update rule

$$w_i = \frac{\sum_j h_\lambda(k_{ij}) \cdot v_j}{\sum_j h_\lambda(k_{ij})} \qquad \text{for all } i \in \{1, 2, \ldots, n\}. \tag{3.5}$$

This can be seen as follows: Consider the modified cost function $\hat{E}_\lambda$ from Eq. (3.2) with substituted fixed hidden variables. It is differentiable on the whole domain, and the partial derivatives of this modified cost function are given by

$$\begin{aligned} \frac{\partial}{\partial w_i} \hat{E}_\lambda(W, k_{ij}) &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_i \sum_j h_\lambda(k_{ij}) \left\| w_i - v_j \right\|^2 = \sum_j h_\lambda(k_{ij})(w_i - v_j) \\ &= w_i \cdot \sum_j h_\lambda(k_{ij}) - \sum_j h_\lambda(k_{ij}) v_j, \end{aligned}$$

and from $\frac{\partial}{\partial w_i} \hat{E}_\lambda(W, k_{ij}) \stackrel{!}{=} 0$ and $\sum_j h_\lambda(k_{ij}) > 0$ it follows immediately that critical points W^* are given by

$$w_i^* \cdot \sum_j h_\lambda(k_{ij}) - \sum_j h_\lambda(k_{ij}) v_j = 0 \quad \Longleftrightarrow \quad w_i^* = \frac{\sum_j h_\lambda(k_{ij}) \cdot v_j}{\sum_j h_\lambda(k_{ij})}.$$

It is straightforward to verify that the Hessian matrix

$$H(W, k_{ij}) = \left(\frac{\partial^2}{\partial w_i \partial w_j} \hat{E}_\lambda(W, k_{ij}) \right)_{i,j \in \{1,2,\ldots,n\}}$$

is positive definite on the whole domain, because H is a diagonal matrix with positive diagonal entries

$$\frac{\partial^2}{\partial w_i^2} \hat{E}_\lambda(W, k_{ij}) = \sum_j h_\lambda(k_{ij}) \; > \; 0.$$

Therefore, $\hat{E}_\lambda$ is *strictly convex* on the whole domain, and there exists only one critical point W^* that is a *global minimizer* of $\hat{E}_\lambda$.

Hence, it follows that the update rule Eq. (3.5) sets the prototypes directly to the *unique global minimum* of function $\hat{E}_\lambda$.

Convergence

In the following, it is shown that Batch Neural Gas converges to a local minimum of the cost function Eq. (3.2). The proof is based on the work of Bottou and Bengio (1995).

We consider a sequence of alternating optimization steps

$$S = W^{(0)} \longrightarrow k_{ij}^{(0)} \xrightarrow{\lambda^{(1)}} W^{(1)} \longrightarrow k_{ij}^{(1)} \xrightarrow{\lambda^{(2)}} W^{(2)} \longrightarrow k_{ij}^{(2)} \xrightarrow{\lambda^{(3)}} \cdots \longrightarrow \cdots$$

generated by the algorithm, where

$W^{(0)}$ is a randomly initialized collection of prototypes from $\mathbb{R}^d$,

$W^{(t)}$, $t \geq 1$, is set to the unique global minimizer of $\hat{E}_{\lambda^{(t)}}(W^{(t-1)}, k_{ij}^{(t-1)})$ by the update rule Eq. (3.5)

$k_{ij}^{(t)}$, $t \geq 0$, is set to the unique global minimizer $k_{ij}^* = k_i(W^{(t)}, v_j)$ of $\hat{E}_{\lambda^{(t)}}(W^{(t)}, k_{ij}^{(t-1)})$, cf. Eq. (3.3)

$\lambda^{(t)}$, $t \geq 1$, is the monotone decreasing neighbourhood range in step t.

Note that, unlike optimal prototype locations, the optimal assignments k_{ij} in each step are independent of the neighbourhood range λ, because $h_\lambda(\cdot)$ is monotone decreasing for all $\lambda > 0$. Hence, it does not alter the order of the ranks, the optimal assignments stay the same no matter how λ is chosen.

Lemma 3.1.2 Given a sequence S of alternating optimization steps of Batch Neural Gas as defined above. It then holds

$$\hat{E}_{\lambda^{(t)}}(W^{(t)}, k_{ij}^{(t)}) \; > \; \hat{E}_{\lambda^{(t+1)}}(W^{(t+1)}, k_{ij}^{(t+1)})$$

for all $t \geq 0$.

Proof. It holds $\hat{E}_{\lambda^{(t)}}(W^{(t)}, k_{ij}^{(t)}) \; > \; \hat{E}_{\lambda^{(t+1)}}(W^{(t)}, k_{ij}^{(t)})$. That is because $h_\lambda(\cdot)$ is strictly monotone in λ, and therefore also $\hat{E}_\lambda(W, k_{ij})$ for fixed W and k_{ij}.

It also holds $\hat{E}_{\lambda^{(t+1)}}(W^{(t)}, k_{ij}^{(t)}) \geq \hat{E}_{\lambda^{(t+1)}}(W^{(t+1)}, k_{ij}^{(t)})$ for all $t \geq 0$, since the update rule Eq. (3.5) always sets the prototypes directly to the unique global minimum of the cost function $\hat{E}_\lambda$ given (fixed) $k_{ij}^{(t)}$.

Furthermore, it is $\hat{E}_\lambda(W^{(t+1)}, k_{ij}^{(t)}) \geq \hat{E}_\lambda(W^{(t+1)}, k_{ij}^{(t+1)})$ for all $t \geq 0$, because also the assignments $k_{ij}^{(t+1)}$ are chosen as the unique global minimizer k_{ij}^* of $\hat{E}_\lambda$ given (fixed) $W^{(t+1)}$ according to Eq. (3.3). Combining these inequalities we have

$$\hat{E}_{\lambda^{(t)}}(W^{(t)}, k_{ij}^{(t)}) \; > \; \hat{E}_{\lambda^{(t+1)}}(W^{(t+1)}, k_{ij}^{(t+1)})$$

for all $t \geq 0$. □

Theorem 3.1.3 Batch Neural Gas (Algorithm 3.1) converges in terms the cost function Eq. (3.2) while performing alternating optimization steps.

Proof. Assume randomly initialized prototypes $W^{(0)}$ and associated $k_{ij}^{(0)}$ are given, whereby $k_{ij}^{(0)}$ is chosen as the unique global minimizer of $\hat{E}_{\lambda^{(0)}}$ for (fixed) $W^{(0)}$ according to Proposition 3.1.1. Let S be the sequence of alternating optimization steps of the algorithm as defined above.

Because of Lemma 3.1.2 the sequence $(\hat{E}_{\lambda^{(0)}}(W^{(0)}, k_{ij}^{(0)}), \hat{E}_{\lambda^{(1)}}(W^{(1)}, k_{ij}^{(1)}), \ldots)$ generated by S is monotone decreasing. It is also bounded because $\hat{E}$ is strictly positive, thus it is convergent. The Batch Neural Gas (Algorithm 3.1) converges to a local minimum of the cost function Eq. (3.1). □

As demonstrated by Cottrell et al. (2006), Batch Neural Gas features a quadratic rate of convergence because it is interpretable as a Newton optimization method, in contrast to the simple gradient descent of the original Neural Gas.

Following the same ideas, we can easily derive the Batch SOM algorithm (Kohonen, 2001) as shown in (Cottrell et al., 2006). The Batch SOM was analyzed by Fort et al. (2002) drawing the conclusions that it has the advantages of simplicity of the computation, quickness, better final distortion, no adaptation parameter to tune, and deterministic reproducible results. But it also has serious drawbacks like bad organization, bad visualization, too unbalanced classes, and strong dependency on the initialization. Theoretical work for the Batch SOM with Heskes' modification was done by Cheng (1997) proving results about convergence and ordering.

In the following, we will discuss how to control the distribution of the final prototypes to achieve a desired behaviour of the algorithm. This could be e.g. the emphasis of rare events or to get an optimal information-theoretical transfer.

3.2 Magnification Control for Batch Methods

As it has been discussed before in Section 2.5, many prototype-based learning methods possess the characteristic property that there is a discrepancy between the densities of the data distribution and the final prototype distribution after learning. In the following, we give an overview about different techniques to modify the prototype-based methods aiming for an arbitrary control of the magnification factor. An explicit control of the magnification is particularly interesting for applications where rare events should be suppressed or, contrarily, emphasized, or an information optimal transfer should be achieved.

Several approaches have been proposed to control the magnification behaviour of prototype-based learning methods. For a thorough survey on magnification control for prototype-based methods consult (Villmann and Claussen, 2006). All techniques have in common that they modify the original update rules

$$\Delta w_i \;=\; \varepsilon_t \cdot h_\lambda(f(i, W, v)) \cdot (v - w_i)$$

of SOM and Neural Gas in some way to influence the dynamics of the method. Some of the methods are able to provide an arbitrary magnification controlled by parameters.

In literature, there are three important approaches to modify the update rule (cf. Villmann and Claussen, 2006), namely

Localized Learning, where a multiplicative factor is introduced in form of a localized learning rate $\varepsilon_i = \varepsilon(w_i)$,

Winner-relaxing Learning, where a winner relaxing term $R(\mu, \kappa)$ is added to the update rule, and

Concave-convex Learning, where an exponent ξ is introduced to scale the shift factor as $(v - w_i)^\xi$.

In our work, we will stick to the localized learning technique, because it has proven to be the most stable one for Neural Gas and SOM (cf. Villmann and Claussen, 2006) and offers arbitrary magnification control as follows.

Localized Learning in Neural Gas

Given a data distribution density function p, localized learning extends the learning rate in (2.9) by a factor which depends on the local data density:

$$\Delta w_i = \varepsilon_0 \cdot p(w_{s(v_j)})^c \cdot h_\lambda(k_i(W, v_j)) \cdot (v_j - w_i) \tag{3.6}$$

where $\epsilon_0 > 0$ is the learning rate and $s(v_j)$ is the winner index (2.3) for data point v_j. Parameter c controls the magnification exponent, where $p(w_{s(v_j)})^c$ vanishes for $c = 0$ leading to standard Neural Gas. That way, a local learning factor depending on the data density at the winner location is added.

As shown by Villmann (2000), Neural Gas with the modified learning rule (3.6) asymptotically obeys the power law

$$\varrho(w_i) \sim p(w_i)^{\alpha'} \quad \text{with} \quad \alpha' = (c+1) \cdot \alpha = (c+1) \cdot \frac{\hat{d}}{\hat{d}+2}\,, \tag{3.7}$$

where $\hat{d}$ is the intrinsic dimension of the data. Obviously, an information theoretically optimum factor $\alpha = 1$ is obtained for $c = 2/\hat{d}$. Larger values of c emphasize input regions with high data density, whereas smaller values put a focus on regions with rarely sampled data points.

A drawback of the above approach is the need to calculate the density $p(w_{s(v_j)})$ at the winning prototype location in each step. Having a transfer of magnification control to batch variants in mind, it would come in handy to precalculate data densities at the locations of the data points of the finite dataset and rely only on these values. With this motivation in mind, we consider the similar learning rule

$$\Delta w_i = \varepsilon_0 \cdot p(v_j)^c \cdot h_\lambda(k_i(W, v_j))(v_j - w_i)\,, \tag{3.8}$$

where the local density at the location of the sampled data point is taken instead of that at the location of the winning prototype.

Now, the average of the learning rule (3.8) can be formulated as an integral

$$\langle \Delta w_i \rangle \sim \int p(v)^c \cdot h_\lambda(k_i(W, v)) \cdot (v - w_i) \cdot p(v)\, dv. \tag{3.9}$$

Since the magnification factor of localized learning has been derived under the assumption of a continuum of prototypes, where $w_{s(v)} = v$ holds, the average update (3.9) yields exactly the same result as the original one (3.6) proposed by Villmann (2000). Thus, the same magnification factor (3.7) results also for the altered learning rule (3.8).

Moreover, the alternative update (3.8) has the benefit that it performs a stochastic gradient descent on the cost function

$$E_\lambda(W, c) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int_V h_\lambda(k_i(W, v)) \cdot \|w_i - v\|^2 \cdot p(v)^{c+1}\, dv. \tag{3.10}$$

That is because (cf. Hammer et al., 2007b) the derivative of cost function (3.10) is given by

$$\begin{aligned} \frac{\partial E_\lambda(W, c)}{\partial w_l} &= \frac{1}{C(\lambda)} \int_V h_\lambda(k_i(W, v)) \cdot (w_l - v) \cdot p(v)^{c+1}\, dv \\ &\quad + \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \int_V h'_\lambda(k_i(W, v)) \cdot \frac{\partial k_i(W, v)}{\partial w_l} \cdot \|w_i - v\|^2 \cdot p(v)^{c+1} dv\,. \end{aligned}$$

It has been shown in (Hammer et al., 2007b) that the second term vanishes. The first term complies straightforward with the average update rule (3.9). Hence, a stochastic gradient descent on the cost function (3.10) is performed when using update rule (3.8).

Thus, learning schemes which optimize the cost function $E_\lambda(W, c)$ yield a map formation with magnification factor α' as given by Eq. (3.7).

Note that for an application of the learning rule (3.6), the density p of the data distribution as well as the effective data dimensionality $\hat{d}$ have to be estimated from the data. These estimations can be done by standard techniques, e.g. Parzen window estimators, and the algorithm of Grassberger and Procaccia (1983), but in general those are indeed difficult problems that are widely discussed in literature (Scott, 1992; Tong, 1993).

Transfer of Localized Learning to Batch Neural Gas

The formulation of localized learning by the modified update rule (3.8) that only relies on local densities at data locations opens a way to integrate magnification control into Batch Neural Gas as follows.

For a given finite dataset $V = \{v_1, v_2, \ldots, v_m\}$ the cost function (3.10) becomes

$$\hat{E}_\lambda(W, c) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot \|w_i - v_j\|^2 \cdot p(v_j)^c. \tag{3.11}$$

As beforehand in (3.2), we substitute the terms $k_i(W, v_j)$ by hidden variables k_{ij}, where $\{k_{ij} : i \in \{1, 2, \ldots, n\}\}$ is a permutation of $\{0, 1, \ldots, n-1\}$ for every $j \in \{1, 2, \ldots, m\}$. This yields

$$\hat{E}_\lambda(W, c, k_{ij}) = \frac{1}{2C(\lambda)} \sum_{i=1}^{n} \sum_{j=1}^{m} h_\lambda(k_{ij}) \cdot \|w_i - v_j\|^2 \cdot p(v_j)^c \,. \tag{3.12}$$

Batch optimization in turn determines optimum k_{ij}, given prototype locations, and optimum prototype locations, given values k_{ij}. With the same arguments as used before (cf. page 20), the optimal assignments k_{ij} are given by the ranks for fixed prototype locations W.

It can be seen further by setting the partial derivatives of Eq. (3.12) to zero,

$$\frac{\partial \hat{E}_\lambda(W, c, k_{ij})}{\partial w_i} = \frac{1}{C(\lambda)} \sum_{j=1}^{m} h_\lambda(k_{ij})(w_i - v_j) \cdot p(v_j)^c \stackrel{!}{=} 0 \,,$$

that the optimum prototypes w_i for fixed assignments k_{ij} are given by the average of the points weighted by the ranks and local data densities,

$$w_i = \frac{\sum_j h_\lambda(k_{ij}) \cdot p(v_j)^c \cdot v_j}{\sum_j h_\lambda(k_{ij}) \cdot p(v_j)^c}. \tag{3.13}$$

The optimality follows in analogy to the discussion in the context of Batch Neural Gas (cf. page 22).

Hence, we obtain a Batch Neural Gas algorithm with local learning featuring a magnification exponent of $(c+1) \cdot \hat{d}/(\hat{d}+2)$ that provides an arbitrary control of the magnification via control parameter c. As discussed beforehand, the intrinsic data dimensionality $\hat{d}$ and the local data density $p(v_j)$ at the positions of the data points

Algorithm 3.2: Batch Neural Gas with Localized Learning

Input

Data $V = \{v_1, v_2, \ldots, v_m\} \subset \mathbb{R}^d$

Data density $p(v) \in [0, 1]$ at all points of V

Control parameter c

Begin

(* Initialize prototypes and neighbourhood range *)

Init $w_i \in \mathbb{R}^d$ randomly for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Compute Euclidean distances...

$$d(w_i, v_j) = \|w_i - v_j\|$$

Determine ranks (break ties deterministically)...

$$k_i(W, v_j) = |\{\, l \in \{1, \ldots, n\} \,:\, d(w_l, v_j) < d(w_i, v_j)\}|$$

Update prototype locations...

$$w_i = \frac{\sum_j h_\lambda(k_{ij}) \cdot p(v_j)^c \cdot v_j}{\sum_j h_\lambda(k_{ij}) \cdot p(v_j)^c}$$

Decrease neighbourhood range...

$$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$$

endfor;

(* Return representative prototypes *)

Return w_i

End.

have to be estimated. In Algorithm 3.2 an outline of the algorithm is presented.

It can be shown in analogy to Theorem 3.1.3 that the Batch Neural Gas algorithm with local learning (Algorithm 3.2) converges to a local minimum of the cost function (3.11) (see also Hammer et al., 2006c).

By now, the magnification control of Neural Gas has proven beneficial not only in artificial experiments but also in several tasks in the field of robotics and satellite image processing (cf. Hammer et al., 2007b; Villmann et al., 2003; Villmann and Heinze, 2000; Merényi and Jain, 2004). Explicit magnification control of Batch Neural Gas has shown its benefits in applications in the field of collision detection of geometric data structures as demonstrated by Weller (2008).

3.3 Incorporating Additional Information

As it is often the case in data mining, there is additional information available about class structures within the data that should be included into processing, otherwise it can be difficult or even impossible to get meaningful results from unsupervised learning. In particular the minimization of the quantization error for high dimensional or noisy data often does not yield meaningful clusters, additional information

such as cluster labels or correlation can help to improve the quality. This issue is also named 'garbage in - garbage out' problem of unsupervised learning and is discussed e.g. by Kaski et al. (2005, 2003).

For an integration of additional information into prototype-based methods, we presume that the additional information is available in form of a class label for each data point (cf. Villmann et al., 2006). Without loss of generality, we assume that these data labels are embedded in $\mathbb{R}^c$ by an unary encoding of the c classes. Let the label that is attached to a data point $v_j \in V$ be denoted by $v_j^L \in \mathbb{R}^c$, i.e. v_j^L is a vector with component $(v_j^L)^{(k)} = 1$ iff v_j^L represents class k, and $(v_j^L)^{(k)} = 0$ else.

Additionally, each prototype w_i features a label $w_i^L \in [0,1]^c$ which shall be adapted during learning depending on the label distribution of the data points in its neighbourhood. By the above construction, we are able to interpret the prototype labels w_i^L as a collection $W^L = \left(w_i^L\right)_{i\in\{1,\ldots,n\}}$ of prototypes in the label space $L = [0,1]^c$.

To integrate supervision into a prototype-based method, we now substitute the original Euclidean metric in the cost function $E_\lambda(W)$ by a combination of the metric in data space and the metric in label space, yielding a new cost function

$$E_\lambda^*(W, W^L, \beta) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} h_\lambda(k_i(W, v_j))\cdot\left((1-\beta)\cdot\|w_i - v_j\|^2 + \beta\cdot\left\|w_i^L - v_j^L\right\|^2\right),$$

where the weighting of the metric in data space and the metric in label space is controlled by parameter β.

In this section, we have chosen Batch Neural Gas as a representative example. Nevertheless the concept can easily be migrated to other prototype-based batch methods like Batch SOM (Hammer et al., 2006b).

Alternating optimization as introduced beforehand leads to new update rules for the prototypes W in data space and the prototypes W^L in label space. It can easily be seen that the cost function separates such that the update rule for the prototypes in data space stays the same as before Eq. (3.5), and the prototype labels are updated in label space by the same rule, i.e. we have

$$w_i^L = \frac{\sum_j h_\lambda(k_{ij})\cdot v_j^L}{\sum_j h_\lambda(k_{ij})} \qquad \text{for all } i \in \{1, 2, \ldots, n\}. \tag{3.14}$$

The optimal assignments k_{ij} are determined subject to a modified rank function that is based on the combined metric. That way, the additional label information exerts influence on the prototype updates in data space turning the unsupervised methods into a supervised one.

It follows directly from the update rule that the components of each prototype label sum up to 1 and are therefore interpretable as probabilities or fuzzy labels. Crisp labels can be obtained by applying e.g. the majority vote principle.

There is another way to interpret the chosen approach of supervision in the Euclidean setting. It can easily be seen that if the data space is Euclidean, data space and label space can be embedded into $\mathbb{R}^{d+c}$ by vectors $(\sqrt{1-\beta}\cdot x, \sqrt{\beta}\cdot x^L)$. The squared Euclidean norm then results in $\|(\sqrt{1-\beta}\cdot x, \sqrt{\beta}\cdot x^L)\|^2 = (1-\beta)\cdot\|x\|^2 + \beta\cdot\left\|v^L\right\|^2$, which yields the same separable cost function as above. Thus, applying the standard batch methods on the embedded Euclidean vector set is equivalent to the supervised batch methods introduced above.

All the batch variants of Neural Gas, SOM, and kMeans can be modified like presented above to incorporate additional label information. The convergence of the supervised variants is guaranteed for all the methods as shown in (Hammer et al., 2006b). In Algorithm 3.3 a sketch of the *Supervised Batch Neural Gas* algorithm (Hammer, Hasenfuss, Schleif, and Villmann, 2006b) is shown as a representative example.

Algorithm 3.3: Supervised Batch Neural Gas

Input

Data $V = \{v_1, v_2, \ldots, v_m\} \subset \mathbb{R}^d$

Labels $\left\{v_1^L, v_2^L, \ldots, v_m^L\right\} \subset \mathbb{R}^c$

Begin

(* Initialize prototypes and neighbourhood range *)

Init $w_i \in \mathbb{R}^d$ randomly for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Compute distances...

$$d_\beta^*(w_i, v_j) = (1-\beta) \cdot \|w_i - v_j\|^2 \; + \; \beta \left\|w_i^L - v_j^L\right\|^2$$

Determine ranks (break ties deterministically)...

$$k_i(W, v_j) = \left|\left\{ l \in \{1, \ldots, n\} \; : \; d_\beta^*(w_l, v_j) < d_\beta^*(w_i, v_j) \right\}\right|$$

Update prototype locations...

$$w_i = \textstyle\sum_j h_\lambda(k_{ij}) \cdot v_j / \sum_j h_\lambda(k_{ij})$$

Update prototype labels...

$$w_i^L = \textstyle\sum_j h_\lambda(k_{ij}) \cdot v_j^L / \sum_j h_\lambda(k_{ij})$$

Decrease neighbourhood range...

$$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$$

endfor;

(* Return representative prototypes and prototype labels *)

Return w_i and w_i^L

End.

The introduced modification now allows to incorporate additional information into the learning process which turns the original unsupervised methods into supervised ones. This supervision helps to improve the quality of the results because the receptive fields are influenced by the underlying class information. Especially, when the prototypes are used as a classifier, this leads to a better generalization ability as it can be seen from the experiments in Section 3.5.

Up to now, we have been discussing methods that are capable of dealing with ten thousands of datapoints from Euclidean spaces. But nowadays there are many situations where the practitioner is facing datasets with millions of points. While handling those huge datasets, the computational limits of the standard methods,

even those of the fast batch approaches, quickly arise. In the following, we will present a technique to handle even those very large datasets by prototype-based methods. The introduced technique is based on patches of smaller size that are consecutively cut from the dataset. While processing the patches, statistical information gained from the processing of former patches is taken into account.

3.4 Processing of Very Large Datasets

Twenty month from now, the amount of electronic data stored worldwide will be doubled—and the rate is accelerating. In many fields, e.g. computational biology, image processing, or particle physics, vast amounts of data are produced from day to day in experiments. As a consequence, nowadays almost every scientific discipline is facing the problem to handle huge data repositories. Here, automatic data mining constitutes an indispensable tool bridging the gap between available data and desired knowledge which would otherwise be inaccessible. Some of the big challenges of real-world data mining have been identified in a panel discussion at the 2007 SIAM International Conference on Data Mining as follows (Hirsh, 2008): Mining massive data which go beyond the capacities of standard algorithms, mining streaming data which is generated in a continuous process and which requires immediate feedback, mining heterogeneous data which stem from different sources, applicability of methods and interpretability of the results by researchers outside the data mining community, among others. These facts pose particular requirements towards standard data mining tools concerning their efficiency, flexibility, and interpretability.

Prototype-based techniques, like the above introduced SOM and Neural Gas, possess several striking features because of which they are widely used in applications: training is intuitive and simple, the final classifier represents classes by geometrically meaningful prototypes, and the algorithms are quite powerful compared to more complex alternative clustering algorithms.

The original online variants that sample point for point from the dataset can be applied to very large datasets without restrictions but due to their convergence properties (Martinetz et al., 1993; Kohonen, 2001) the computation time needed might be beyond practical limits.

In the fast batch variants, all data is used for every iteration over and over again. But for a reasonable performance it is necessary to hold all data at once in random-access memory during computation. Unfortunately, those very large datasets cannot be held at once within memory for processing due to the sheer amount of data.

To fill the gap, a special computation scheme is introduced, relying on a single pass technique of fixed sized patches. In between processing of the patches a sufficient statistic is passed over describing the data distribution in the former patches. The patch size can then be chosen to match the given memory constraints. The proposed patch versions reduce the computation and space complexity with a small loss of accuracy. As beforehand, we develop the idea on the basis of Neural Gas to give a concrete and clear representation, how to migrate the idea to SOM or other prototype-based methods is obvious besides technicalities.

A variant of Neural Gas for very large datasets was introduced by Alex et al. (2009). It is based on a patch processing scheme that splits the dataset into a number of disjunct patches and applies Batch Neural Gas consecutively on each patch. Additionally, each patch is enhanced by the prototypes of the processing of the former patch weighted by the sizes of their receptive fields. That way, a sufficient statistic of former data is transferred in between the patches. Batch Neural

Gas in conjunction with this patch scheme shows a very good performance in the experiments. Moreover, it can easily be parallelized with an almost linear speedup as shown in (Alex and Hammer, 2008). Open questions by now concern with the existence of bounds on the quantization error and the sufficiency of the statistics from a theoretical point of view.

In the following, the *Patch Neural Gas* algorithm is presented in more detail, laying the foundations to build on in later chapters about patch schemes for prototype-based methods in non-Euclidean spaces.

Patch Neural Gas

Assume a finite dataset $V = \{v_1, v_2, \ldots, v_m\} \subset \mathbb{R}^d$ is given. During processing of *Patch Neural Gas*, n_p disjoint patches of fixed size $p = \lfloor m/n_p \rfloor$ are taken from the dataset V consecutively[1], i.e. every patch P_i is a subset of V and it holds $\bigcup_i P_i = V$.

The idea of the patch scheme is to add the set of final prototypes W_{t-1} from the processing of the former patch P_{t-1} as additional weighted datapoints to the current patch P_t, forming an extended patch $P_t^* = P_t \cup W_{t-1}$ to work on further.

Every original datapoint $v_j \in V$ is equipped with a multiplicity $m(v_j) = 1$. The additional datapoints – as former prototypes – are weighted according to the size of their receptive fields taking into account possible multiplicities, i.e. how many datapoints they have been representing in the former patch counted with multiplicities. The multiplicity $m(w_i)$ of a prototype $w_i \in W_t$ is then given by

$$m(w_i) = \sum_{v \in P_t^* \cap V_i(W_t)} m(v).$$

Now, the original Batch Neural Gas method must be modified to take into account the multiplicities. Incorporating the multiplicities into the cost function (3.2) yields

$$\hat{E}_\lambda(W, k_{ij}) = \frac{1}{2} \sum_i \sum_j h_\lambda(k_{ij}) \cdot m(v_j) \cdot \|w_i - v_j\|^2 ,$$

since every datapoint v_j is weighted with multiplicity m_j. The new update rule for the prototypes derived from the modified cost function in analogy to (3.5) is given by

$$w_i = \frac{\sum_j h_\lambda(k_{ij}) \cdot m(v_j) \cdot v_j}{\sum_j h_\lambda(k_{ij}) \cdot m(v_j)} \qquad \text{for all } i \in \{1, 2, \ldots, n\}.$$

For the processing of each patch, all properties concerning optimal assignments and convergence follow the same way as in Section 3.1, since the multiplicities can be interpreted as multiples of a single point.

Picking up the pieces, we obtain the *Patch Neural Gas* algorithm (Alex, Hasenfuss, and Hammer, 2009) as sketched in Algorithm 3.4.

Using the above introduced technique of patch processing, it is now possible to handle very large datasets with millions of data points. The technique has shown its strength in several experiments as it can be seen in Section 3.5 and also later on in Section 4.6.

In the next section, we will present experimental results for all of the above introduced methods and techniques.

[1] The remainder is no further considered here for simplicity. In the practical implementation the remaining datapoints are simply distributed over the first $(M - p \cdot n_p)$ patches.

Algorithm 3.4: Patch Neural Gas

Begin

Cut the first Patch P_1

Apply Batch Neural Gas on $P_1 \longrightarrow$ Prototypes W_1

Update Multiplicities $m(W_1)$

Repeat for $i = 2, \ldots, n_p$

Cut patch P_i

Construct Extended Patch $P_i^* = P_i \cup W_{i-1}$

Apply modified Batch Neural Gas with Multiplicities

$\longrightarrow$ Prototypes W_i

Update Multiplicities $m(W_i)$

Return final Prototypes W_{n_p}

End.

3.5 Experimental Results and Applications

In the following, experimental results and applications of the above introduced methods are presented. The data in the assessment was chosen to cover different aspects that are common in real-life applications, or to demonstrate the behaviour of the methods in situations which are of theoretical importance. Also, most of the times, data that has already been used in literature was chosen to provide a comparison to the state-of-the-art. There emerged already some interesting applications of the methods, most notably Batch Neural Gas has proved its amenities in the efficient compression of sequences of 3D models in Computer Graphics by determining key-frames in a very fast way (Winkler et al., 2008).

Experimental Results on Batch Methods

We demonstrate the behavior of the algorithms in different scenarios which cover characteristic situations. All algorithms have been implemented based on the SOM Toolbox for Matlab (Vesanto et al., 2000). We used k-means, SOM, and Online Neural Gas with default parameters as provided in the toolbox. Batch NG and Batch SOM have been implemented according to the above formulas. Note, that the training sets are normalized prior to training using the z-transformation, i.e. data is normalized in a way that the expectation value in each dimension is 0 and the standard deviation is 1. Initialization of prototypes takes place using small random values. The initial neighborhood rate for neural gas is set to $\lambda = n/2$, n being the number of neurons, and it is multiplicatively decreased during training. For SOM, we used square lattices of $n = \sqrt{n} \times \sqrt{n}$ neurons and a rectangular neighborhood structure, whereby $\sqrt{n}$ is rounded to the next integer. Here the initial neighborhood rate is $\sqrt{n}/2$. To measure the quality of the mapping the empirical quantization error according to eq. (2.2) was determined.

Synthetic data

The first data set is the two-dimensional synthetic data set from (Ripley, 1996) consisting of 250 data points and 1000 training points. Clustering has been done

using $n = 2, \ldots, 25$ prototypes, resp. the closest number of prototypes implemented by a rectangular lattice for SOM. Training takes place for $5n$ epochs.

The mean quantization error on the test set is shown in Fig. 3.1. For k-means, idle prototypes can be observed for large n. For Batch SOM and standard SOM, the quantization error is worse (ranging from 1.7 for two neurons up to 0.3 for 24 neurons, not depicted in the diagram), which can be attributed to the fact that the map does not fully unfold upon the data set and edge effects remain, which could be addressed to a small but non-vanishing neighborhood in the convergent phase in standard implementations of SOM which is necessary to preserve topological order. Thus, Batch NG allows to achieve results competitive to NG in this case, however, using less effort. The median variants whose results are also displayed in the figure will be discussed later on.

Segmentation data

The segmentation data set from the UCI repository consists of 210 (training set) resp. 2100 (test set) 19 dimensional data points which are obtained as pixels from outdoor images preprocessed by standard filters such as averaging, saturation, intensity, etc. The problem is interesting since it contains high dimensional and only sparsely covered data. The quantization error obtained for the test set is depicted in Fig. 3.1. As beforehand, SOM suffers from the restriction of the topology. Neural gas yields very robust behavior, whereas for k-means, idle prototypes can be observed.

Checkerboard

This data set is taken from (Hammer et al., 2005). Two-dimensional data are arranged on a checkerboard (cf. Figure 3.12), resulting in 10 times 10 clusters, each consisting of 15 to 20 points. For each algorithm, we train 5 times 100 epochs for 100 prototypes. Obviously, the problem is highly multimodal and usually the algorithms do not find all clusters. The number of missed clusters can easily be judged in the following way: the clusters are labeled consecutively using labels 1 and 2 according to the color black resp. white of the data on the corresponding field of the checkerboard. We can assign labels to prototypes a posteriori based on a majority vote on the training set. The number of errors which arise from this classification on an independent test set count the number of missed clusters, since 1% error roughly corresponds to one missed cluster.

	Online NG	Batch NG	Online SOM	Batch SOM	Batch k-Means
quantization error					
train	0.0043	0.0028	0.0127	0.0126	0.0043
test	0.0051	**0.0033**	0.0125	0.0124	0.0050
classification error					
train	0.1032	0.0330	0.2744	0.2770	0.1136
test	0.1207	**0.0426**	0.2944	0.2926	0.1376

Table 3.1: Quantization error and classification error on Checkerboard Data for posterior labeling for training and test set (both are of size about 1800). The mean over 5 runs is reported. The best results on the test set is depicted in boldface.

The results are collected in Tab. 3.1. The smallest quantization and classification error is obtained by Batch NG. As beforehand, SOM and Batch SOM do not fully

unfold the map in the data. In the same way online NG does not achieve a small error because of a restricted number of epochs and a large data set which prevents online NG from fully exploring the data space. K-means also shows a quite high error (it misses more than 10 clusters) which can be explained by the existence of multiple local optima in this setting, i.e. the sensitivity of k-means with respect to initialization of prototypes. In contrast, Batch NG finds all but 3 to 4 clusters. Hence, batch versions show much better behavior than their online correspondents, due to a faster convergence of the algorithms. Here, SOM suffers from border effects, whereby the topology mirrors precisely the data topology. Batch NG yields quite good classification results which are even competitive to supervised prototype-based classification results as reported in (Hammer et al., 2005).

Experimental Results on Supervised Methods

Artificial data

The main difference of Batch NG (BNG) with posterior labeling and Supervised Batch NG (SBNG) consists in the fact that the rank assignments also take into account whether the labels fit. This has the effect that the prototypes of SBNG better account for cluster borders of labeled data points, whereas BNG only follows the overall statistics. The parameter β controls the strength of the label contribution, $\beta = 1$ corresponds to the original BNG. This effect can be clearly observed in the following example. We consider two Gaussian clusters labeled by 0 resp. 1, whereby points with x-component at least 0 are dropped for class 0, and points with x-component at most 0 are dropped for class 1. Hence, the classes are well separated, whereby a couple of data points lies close to the decision boundary. Figure 3.2 shows a typical result of the receptive fields of the prototypes obtained by BNG and SBNG with mixing parameter $\beta = 0.1$, respectively.

Thereby, prototypes of BNG are labeled by a majority vote within the receptive field. Fuzzy labels for SBNG arise automatically during training, these are turned into crisp classes based on the largest component of the label vector. Obviously, SBNG well approximates the decision border, whereas BNG yields a couple of errors at this region. This corresponds to the classification accuracy of 99.1% for SBNG and 97.8% for BNG.

Iris data

We train batch BNG and SBNG using 9 prototypes on the well-known iris dataset, which consists of 150 points characterized by 4 real-valued attributes coming from three classes of equal size. Class 1 is well separated from class 2 and 3, but classes 2 and 3 slightly overlap. The task here is to train a classifier on a subset of the data and assess the trained classifier with the remaining part. The accuracy, i.e. the fraction of correctly classified data points, is measured. For each run, the set is randomly divided into a training and test set of equal size, and averages of the accuracy over 50 runs are reported. The neighborhood range λ is multiplicatively annealed starting from 4.5 over 100 training epochs. Different values of the mixing parameter β are reported.

The classification accuracy on the training and test set for SBNG and, in comparison, for BNG with posterior labeling can be observed in Fig.3.3. Obviously, the classification accuracy becomes better for smaller β, i.e. more emphasis of the given data labels. Thereby β must not become 0 which corresponds to a pure label adaptation without adaptation of the prototypes. $\beta = 1$ corresponds to the original BNG. The classification accuracy of the original BNG is inferior compared to the supervised version due to the overlap of classes 2 and 3 which is not accounted for

by the overall statistics of the input vectors. The algorithm reported in (Villmann et al., 2006) (Gaussian approximation) achieves (in a single run with parameter $\beta = 0.5$) a training set accuracy of 0.85 and test set accuracy of 0.91 using 9 prototypes, thus it is better than post labeled NG, but worse compared to Supervised Batch NG for this parameter choice.

Wisconsin breast cancer

The well-known Wisconsin breast cancer data (WDBC) consists of 569 data points described by 30 real-valued input features. It originates from medical research where fine needle aspiration biopsy was applied to suspicious breast masses and microscope images of the tissues were analyzed. Ten real-valued features describing characteristics of cell nuclei like radius, perimeter, fractal dimension, symmetry, concavity, etc. were gathered. For every of the ten features the mean, standard deviation, and the largest value over all observed cells were calculated leading to 30-dimensional data points. The task is to determine whether the sample is benign or malignant. There are 357 benign and 212 malignant samples.

For training 20 prototypes were used and the same parameters as beforehand, starting with an initial neighborhood range 10. The results are presented in Fig.3.4. As before, a larger emphasis on the correctness of the labels yields a better classification accuracy which is superior to BNG. Interestingly, the approach presented in (Villmann et al., 2006) which relies on a different supervised extension of NG (Gaussian approximation of the rank) achieves an accuracy of 0.92 for the training set and 0.91 on the test set for a mixing parameter 0.5, which is in this case worse than the result obtained by BNG with posterior labeling.

Experimental Results on Magnification Control

For all experiments the initial neighborhood range λ_0 is chosen as $n/2$ with n the number of neurons used. The neighborhood range $\lambda(t)$ is decreasing exponentially with the number of adaptation steps t according to $\lambda(t) = \lambda_0 \cdot (0.01/\lambda_0)^{t/t_{\max}}$ (cf. Martinetz et al., 1993). The value $t_{\max}$ is given by the number of epochs of a training run.

Control experiment

In a first control experiment we use the setting as proposed e.g. in (Villmann and Claussen, 2006). We use the distribution $(x_1, \ldots, x_d, \prod_{i=1}^{d} \sin(\pi \cdot x_i))$ for $d \in \{1, 2, 3\}$ and uniformly distributed x_i in $[0, 1]$. Thus, the intrinsic data dimensionality in these examples is $\hat{d} = d$. The number of stimuli is 2500 for $d = 1$, 5000 for $d = 2$, and 10000 for $d = 3$. These numbers account for the fact that the necessary number of data points to sufficiently sample a d-dimensional data space grows exponentially with d. Due to computational complexity a $k \cdot 2^d$ - scheme with a large constant $k = 2500$ was chosen instead of k^d.

Optimum information transfer for NG with magnification control can be expected for values which yield $\alpha' = (c + 1) \cdot \hat{d}/(\hat{d} + 2) \stackrel{!}{=} 1$, where c is the control parameter and $\hat{d}$ the intrinsic dimensionality of data. Hence, it is $c = 2$ ($\hat{d} = 1$), $c = 1$ ($\hat{d} = 2$), and $c = 2/3$ ($\hat{d} = 3$). We train an NG network with magnification control for control values $c \in [-1.5, 3.5]$ (step size 0.25) such that the overall behavior of the local learning rule for different c can be observed. An NG network with 50 neurons, initial neighborhood range 25 and 200 epochs per training run has been used. The reported results have been averaged over 20 runs. The data density $p(v)$ has been estimated by a Parzen window estimator with bandwidth given by the average training point distances divided by 3.

The information theoretic quality of the information transfer of the map can be judged by counting the balance of patterns in the receptive fields. For equal values, optimum information transfer is achieved. We count the number of data points of the training set in the receptive of a given prototype averaged over the number of data points and report the entropy thereof. The resulting values are reported in Fig. 3.5. The entropy should be maximum for optimum information transfer, i.e. we expect the optimum for $c = 2$ ($\hat{d} = 1$), $c = 1$ ($\hat{d} = 2$), and $c = 2/3$ ($\hat{d} = 3$), respectively. As indicated by the arrows, the experimental optimum of the curves is very close to the expected theoretical values in all cases, thus confirming the theory presented in this thesis.

Remote sensing image analysis

In geophysics, geology, astronomy, and many environmental applications airborne and satellite-borne remote sensing spectral imaging has become one of the most advanced tools for collecting vital information about the surface of the Earth and other planets. Thereby, automatic classification of intricate spectral signatures has turned out far from trivial: discrimination among many surface cover classes and discovery of spatially small, interesting species proved to be an insurmountable challenge to many traditional methods. Usually, spectral images consist of millions of data points such that clustering often becomes a necessary prerequisite for further processing and inspection. Thereby, it is often crucial to preserve characteristics from spatially small interesting regions which are easily oppressed by standard vector quantization. Here we show that magnification control can help to preserve the characteristics of rare classes.

We consider a LANDSAT TM image from the Colorado area, U.S.A., for which a manually generated label map of surface covers is available. The original image covers an area of about 50x50 kilometers yielding approximately 2 million data points which can be divided into 14 classes with different surface covers. Data are 6-dimensional, the thermal band with low resolution was left away. An initial Grassberger-Procaccia analysis yields the intrinsic dimensionality $\hat{d} = 3.1414$ (Villmann et al., 2003). For the experiments, we randomly selected 17004 data points with a representative class distribution. Tab. 3.2 shows the different surface cover types and their respective percentage. As depicted in Tab. 3.3 dimensions 1, 2, 3, 5, and 6 are correlated.[2]

Therefore, we depict the projection onto dimensions 1 and 4. Data are approximately unimodal with several small characteristics at the data borders, as can be seen in Fig. 3.9. The second largest class (1-scotch pine) lies very close to the center of gravity, the largest one (13-dry meadow) a bit off center. Small extremal classes at the data borders are given by 9-water and 14-alpine vegetation.

We train a neural map with 60 neurons for 100 epochs. The local data density is estimated by Parzen windows as before. Thereby, we use different control values. The final location of neurons is depicted in Figs. 3.6-3.8 for $\alpha = 0.01$, $\alpha = 0.61$ (i.e. standard neural gas), and $\alpha = 2$. Thereby, the star denotes the center of gravity of the data points. Obviously, the neurons focus on the common effects and move towards the center of gravity for large values of α, whereas they sample rare effects at the borders (e.g. class 14 at the upper right corner) for small values of α.

This behavior can be quantified by counting the number of neurons responsible for a given class. Thereby, neurons are labeled according to a majority vote on their receptive field. The number of hits for every class averaged over 10 runs is depicted in Fig. 3.10. $c = 0$ corresponds to standard NG, $c = 0.64$ is close to the

[2]This observation is also stressed when training a SOM using the data: the gradients of the component planes of dimensions 1, 2, 3, 5, and 6 look very similar, whereas the gradient of the component plane for dimension 4 is perpendicular.

class number	percentage	surface cover type
1	17.30%	scotch pine
2	10.46%	Douglas fir
3	5.32%	pine/fir
4	7.93%	mixed pine forest
5	4.26%	supple/prickle pine
6	6.13%	aspen/mixed pine forest
7	5.05%	without vegetation
8	8.12%	aspen
9	0.48%	water
10	2.83%	moist meadow
11	3.78%	bush land
12	7.78%	grass/pastureland
13	19.77%	dry meadow
14	0.79%	alpine vegetation

Table 3.2: Surface cover classes and their respective percentage

dim	**1**	**2**	**3**	**4**	**5**	**6**
1	1	**0.9564**	**0.9493**	0.4691	**0.8162**	**0.8544**
2	**0.9564**	1	**0.9678**	0.5858	**0.8448**	**0.8682**
3	**0.9493**	**0.9678**	1	0.4867	**0.8339**	**0.8897**
4	0.4691	0.5858	0.4867	1	0.5783	0.4417
5	**0.8162**	**0.8448**	**0.8339**	0.5783	1	**0.9574**
6	**0.8544**	**0.8682**	**0.8897**	0.4417	**0.9574**	1

Table 3.3: Correlation coefficients of the data dimensions: Correlated dimensions 1, 2, 3, 5, and 6 are shown in bold face.

information theoretic optimum. $c \to 0$ focuses on rare events, whereas large values c account for final prototype locations in typical regions, the center of gravity as the limit. The depicted values are $c \in \{-0.9836, -0.91817, -0.59, 0, 0.64, 2.27, 3.9\}$ which corresponds to a magnification factor $\alpha \in \{0.01, 0.05, 0.25, 0.61, 1.0, 2.0, 3.0\}$ assuming $\hat{d} \approx 3.1414$. One can see that small classes at the borders of the data set (classes 9-water and 14-alpine vegetation) are represented by neurons only for small values c which emphasize rare events. In the limit of large c, class 1 (scotch pine), which is the second largest class and located near the center of gravity, accumulates most neurons. Thus, magnification control allows, depending on the control values, to detect rare cover types or, conversely, to focus on the most representative surface cover type in inspections.

Experimental Results on Patch Processing

We test patch and batch versions of k-means and NG for three different data sets, a very simple four-mode clustering problem, a highly-multimodal benchmark dataset from (Cottrell et al., 2006), and a large data set from the 1998 KDD cup data mining contest which was also tested in (Farnstrom et al., 2000). For all experiments, cluster centers are initialized at random positions close to the origin which, due to data normalization, constitutes a reasonable center point for all data sets. It can be observed that the exact position of the initial prototypes has hardly an influence on the overall results and virtually no influence if averaged over several runs, as done

	Batch KM	Patch KM	Batch NG	Patch NG
Mean quantization error (Four Clouds)				
	1.25	1.28	1.25	1.26
Variance $\cdot 10^3$				
	0.07	0.37	0.07	0.1
Mean classification accuracy in % (Checkerboard)				
patch size 200				
	87.32	86.38	93.35	90.18
patch size 400				
		85.43		90.76
patch size 600				
		84.47		91.80
Mean quantization error (KDD)				
patch size 1000				
	0.468	0.464	0.460	0.462
variance $\cdot 10^3$				
	0.265	0.265	0.03	0.012
patch size 10000				
		0.468		0.461
variance $\cdot 10^3$				
		0.114		0.011

Table 3.4: Quantization error or classification accuracy, respectively, as obtained by the different clustering algorithms for the Four Clouds data set, the Checkerboard data set, and the KDD data set for batch and patch clustering using different patch sizes

in the experiments.

Four Clouds

Data stem from a mixture of four Gaussians with unit variance in two dimensions as depicted in Fig. 3.11. The set consists of 40000 data points. Training is done by ten-fold crossvalidation using four clusters and a random (i.i.d.) order of the points. Thereby, the results on the training and test sets differ only slightly. We use a crossvalidation to evaluate the robustness and sensitivity of the algorithm with respect to noise. The patch size is 100, and each clustering of a patch includes 20 epochs, thereby annealing the neighbourhood of NG from $n/2 = 2$ to 0. The mean value of the quantization error is given in Tab. 3.4. Obviously, the results are almost identical for all runs due to the simplicity of the data set. Very slight differences of the quantization error are due to the fact that the points at the (overlapping) cluster borders are assigned differently. For all methods, the four cluster centers have been found in every run, showing no differences between batch- and patch-clustering and different patch sizes, respectively, in the principled location of the cluster centers.

Checkerboard

Data constitute a multimodal distribution with 100 clusters in two dimensions as depicted in Fig. 3.12 separated into a training and test set. The overall number of data points in both, training and test set is about 2000. Training is done using 100 neurons. The patch size is chosen as 200, 400, and 600, respectively. The initial neighbourhood size of NG, 10, is annealed to 0 during training. The number of epochs is 20 for each patch clustering. It is easily possible to evaluate the number of missed clusters in this task by referring to the classification error of the underlying checkerboard: we assign the label 0 and 1 to the data points such that a

checkerboard-pattern with 100 fields arises. We label the cluster centers based on the training set and evaluate the classification error of this clustering on the test set. Each percentage of misclassification corresponds to one missed cluster (of 100 total clusters). The mean correct classification (in percentage) obtained in ten runs is reported in Tab. 3.4. In this case, the quantization error does not allow to infer the quality of the clustering due to the large number of comparably small clusters: it is around 0.055 for batch and patch variants of k-means and NG, hardly showing any difference between batch and patch clustering.

The clustering results clearly show the following: the task is quite hard and all methods miss a few cluster centers (ranging from 15 for k-means to 7 for NG). Note that each of the 100 clusters is only represented, on average, by 20 data points, and the number of neurons is chosen exactly as 100, i.e. every neuron must represent exactly one center for optimum classification accuracy. In these cases, a clear difference of patch and batch clustering can be observed: overall, patch clustering finds about 3-4 clusters less compared to batch clustering. This effect depends slightly on the size of the patches, as can be observed in particular for NG. However, for reasonable patch size the loss in accuracy is only minor and it could easily be accounted for by using a slightly larger number of cluster centers than necessary. In this scenario the dependency of k-means on initialization pops out for both, batch and patch clustering.

Due to the intuitive evaluation of the clustering result by means of the classification error, a comparison to online-neural gas, which can directly be applied to large data sets since it adapts the prototypes directly after every pattern, is easily possible in this scenario: After only one pass through the data, no convergence can be observed in the sense that the neurons are not located in the cluster centers at all. After about 5 epochs, convergence can be observed. On average, 15 clusters are missed after 5 epochs, about 10 clusters are missed after 10 epochs, about 8 clusters are missed after 20 epochs (this setting is comparable to the setting tested for batch NG, whereby batch NG obtains, on average, slightly better performance), about 5 clusters after 50 epochs, and about 3 after 100 epochs. Thus, several passes over the entire data set are necessary for online NG to show competitive results to patch NG.

KDD Cup Data Mining Contest

This data set stems from the 1998 KDD cup data mining contest, and we use the same setting as proposed in (Farnstrom et al., 2000). Data contains 95412 records with 481 statistical fields which describe statistical information about people who made charitable donations in response to direct mailing requests. For our experiments, 56 features from these fields have been selected, including numerical features such as donation amount, income, age; date values, such as donation date, date of birth; and binary values such as income category. Data are preprocessed such that only numerical values with zero mean and unit variance result. The number of clusters was set to 10, as proposed in (Farnstrom et al., 2000), and the number of epochs is 20. The mean quantization error averaged in a ten-fold crossvalidation is reported in Tab. 3.4 for two different patch sizes corresponding to roughly 1% and 10% of the data, respectively. Since the preprocessing in (Farnstrom et al., 2000) has been described only qualitatively, we cannot compare to the results reported there, but we test k-means in our setting. Obviously, a slight improvement of NG compared to k-means can be observed, whereby patch clustering using 10% of the data only slightly reduces the achieved results. Interestingly, the variance of the results can be reduced by patch clustering compared to batch clustering, and – as expected – NG compared to k-means. However, it is clearly demonstrated that for all variants, patch optimization only slightly decreases the overall result whereby

	Batch	Patch	Ratio
Checkerboard			
	k-means		
Accuracy	0.8707	0.8675	1.0037
Time (ms)	1053.4	452.5	2.3280
	NG		
Accuracy	0.9353	0.9074	1.0307
Time (ms)	102781.9	37074	2.7723
11 Clouds			
	k-means		
Accuracy	0.9928	0.0.995	1
Time (ms)	76376.5	64465.6	1.18
	NG		
Accuracy	0.9984	0.9984	1
Time (ms)	94703.9	79134.5	1.2

Table 3.5: Classification results and time difference of batch and patch clustering with 20 (for batch) and 5 (for patch) epochs and patch size 200 for the Checkerboard data, and for batch and patch clustering with 8 (for batch) and 5 (for patch) epochs for the simpler 11 clouds data set.

reducing the required buffer size to a fixed size and reducing the training to a single run over the overall data set (combined with a small number of epochs for each patch).

Comparison of the Clustering Times

Due to the smaller size of the data sets, patch clustering requires less iterations until convergence compared to batch clustering. This effect can be measured in experiments as follows:

For the Checkerboard data, we perform the same experiment as beforehand, thereby using 20 iterations for batch clustering and 5 iterations for patch clustering for a patch size 200. These numbers represent the necessary number of iterations until convergence for the respective scenario. As can be seen from Tab. 3.5, the quotient of the performance measured by means of the classification error is close to one, whereby the gain of the efficiency accounts for a factor larger than 2.

For a simpler data set consisting only of 11 clouds and 44000 data points, the effect is a bit less pronounced: for batch and patch clustering, the classification accuracy is the same while obtaining an efficiency gain of about 1.2 for patch compared to batch clustering. This gain is due to the reduced number of necessary epochs until convergence, which are 5 for patch clustering and 8 for batch clustering, see Tab. 3.5.

Nonstationary Distributions

So far, experiments were conducted using i.i.d. data. It can be expected that streaming data usually displays a trend because of different times or modes of data acquisition. Thus, it is crucial to test the behavior of patch clustering for nonstationary distributions. When dealing with nonstationary distributions, two different objectives can be specified: on the one hand, it can be desirable to only take the most recent data into account, incorporating a 'forgetting factor' into the methods for life-long learning. This can be achieved by means of a leaky weighting of data points

or prototypes, respectively, for patch clustering. However, we are not interested in the scenario of life-long learning in this contribution. Rather, we are interested in the question whether patch clustering can achieve results competitive to full batch clustering also if the consecutive patches are non i.i.d. but display a trend. Thus the question is whether the representation of already seen data points by means of prototype centers is sufficient even if the prototypes are not yet remotely representative for the full data set.

We test this question by means of the 11 clouds distribution as beforehand, see Fig. 3.13. We use a data set consisting of 11000 points. The patch size is 1000, i.e. it corresponds to one mode of the data. We present the data set in three different ways to patch clustering, for comparison: randomly permuted i.i.d. data, data which are strictly sorted according to the modes (from left to right and bottom to top), and data which are mixed in such a way that every patch comprises 56% of a specified mode and 21% resp. 1% of the immediate two preceding and succeeding modes. We train patch clustering using 100 epochs per patch. The exact data presentation for every patch as well as the results of patch clustering after every patch are displayed in Figs. 3.14,3.15,3.16. As can be seen, the behavior is very robust with respect to the final results: in all runs, the final clustering is almost identical with respect to the given data distribution. Interestingly, the third case, where modes are presented consecutively subject to a diffusion process, seems the most difficult setting in this case. The clustering is affected by this distribution in the form of a small shift of the ideal center of cluster 7 to the right in the final result as can be seen in Fig.3.16 in the last three patches. However, the receptive fields in the data set are not affected by this small shift.

As it can be seen from the experiments, the batch variants of the prototype-based methods generate results of the same quality as the standard online approaches but much faster. Also the extensions to supervision, magnification control, and patch processing have shown their benefits in the experiments.

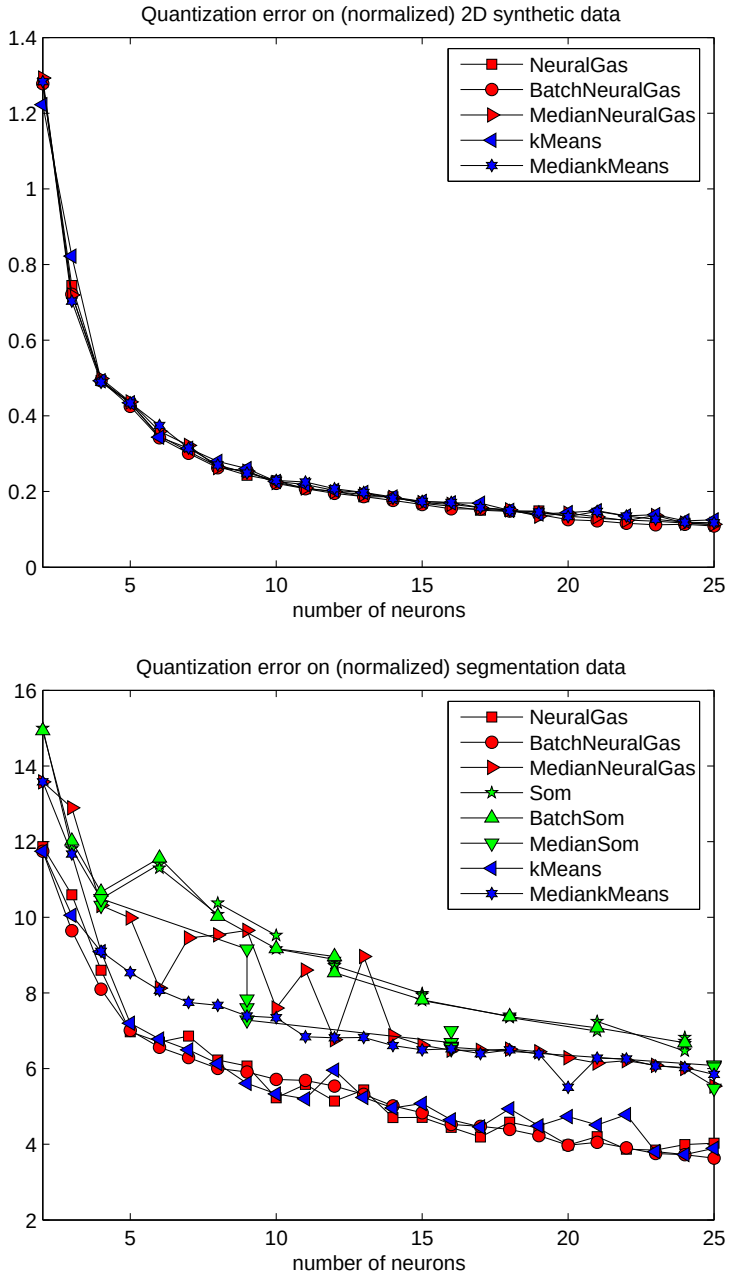


Figure 3.1: Mean quantization error of the methods for the synthetic data set (top) and the segmentation data set (bottom)

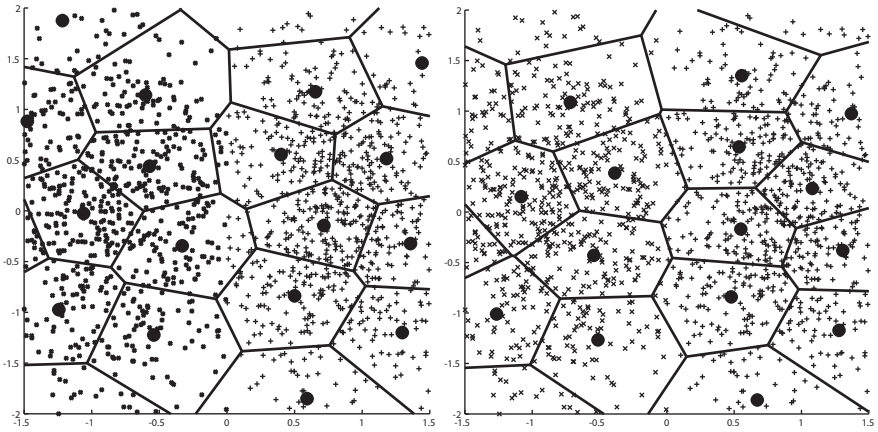


Figure 3.2: Receptive fields obtained by BNG (left) and SBNG (right) on an artificial two-dimensional data set. Obviously, the incorporation of the label information for SBNG yields a better separation of the two classes; the prototype locations follow the classification boundary.

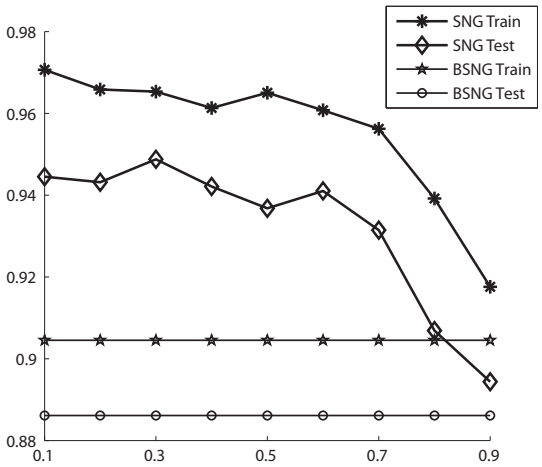


Figure 3.3: Accuracy on the training and test set achieved by Supervised Batch NG (SBNG) and Batch NG (BNG) on the iris dataset for different mixing parameters β.

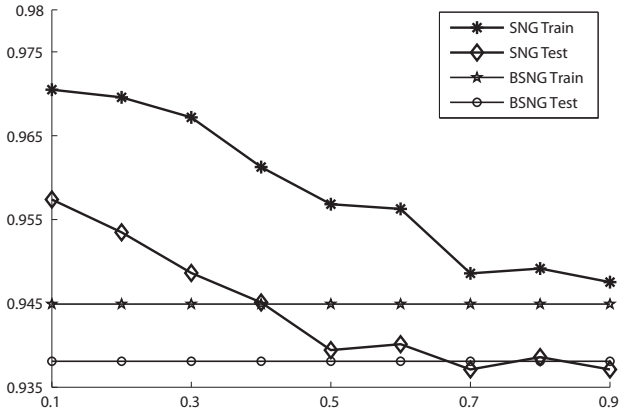


Figure 3.4: Accuracy on the training and test set achieved by Supervised Batch NG (SBNG) and Batch NG (BNG) on the Wisconsin breast cancer dataset for different mixing parameters β.

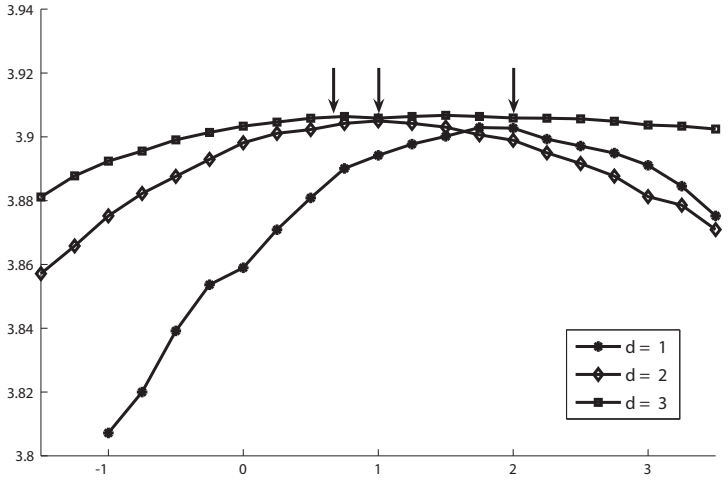


Figure 3.5: Entropy of map formation for different values c of magnification control and training sets of intrinsic dimensionality $\hat{d} \in \{1, 2, 3\}$. The arrows indicate the expected optima of the entropy according to the underlying theory.

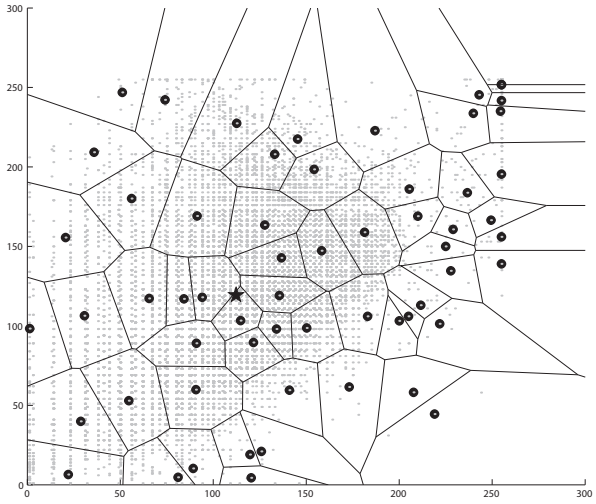


Figure 3.6: Final prototype location for $\alpha = 0.01$

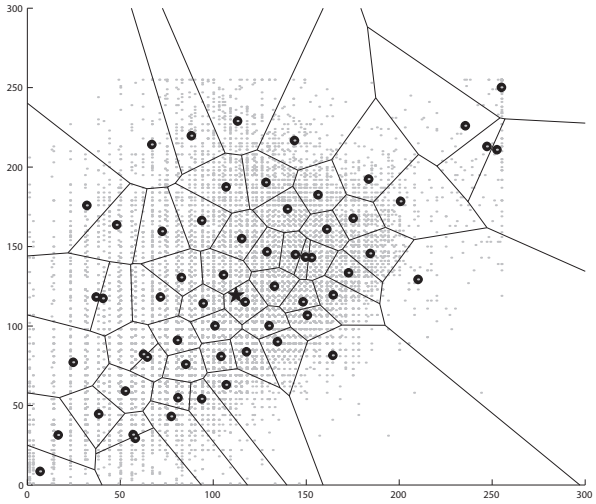


Figure 3.7: Final prototype location for $\alpha = 0.61$

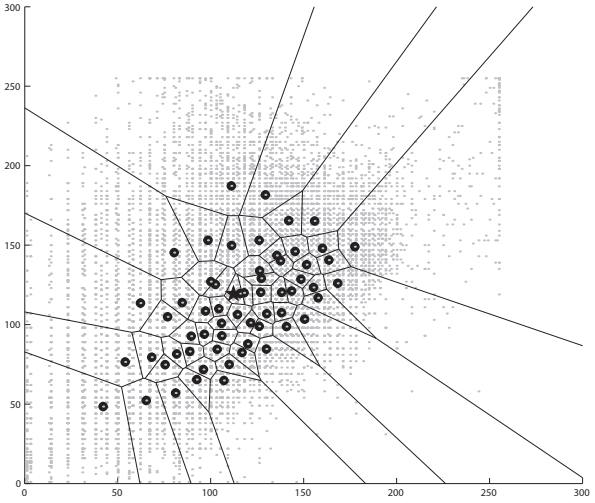


Figure 3.8: Final prototype location for $\alpha = 2.0$.

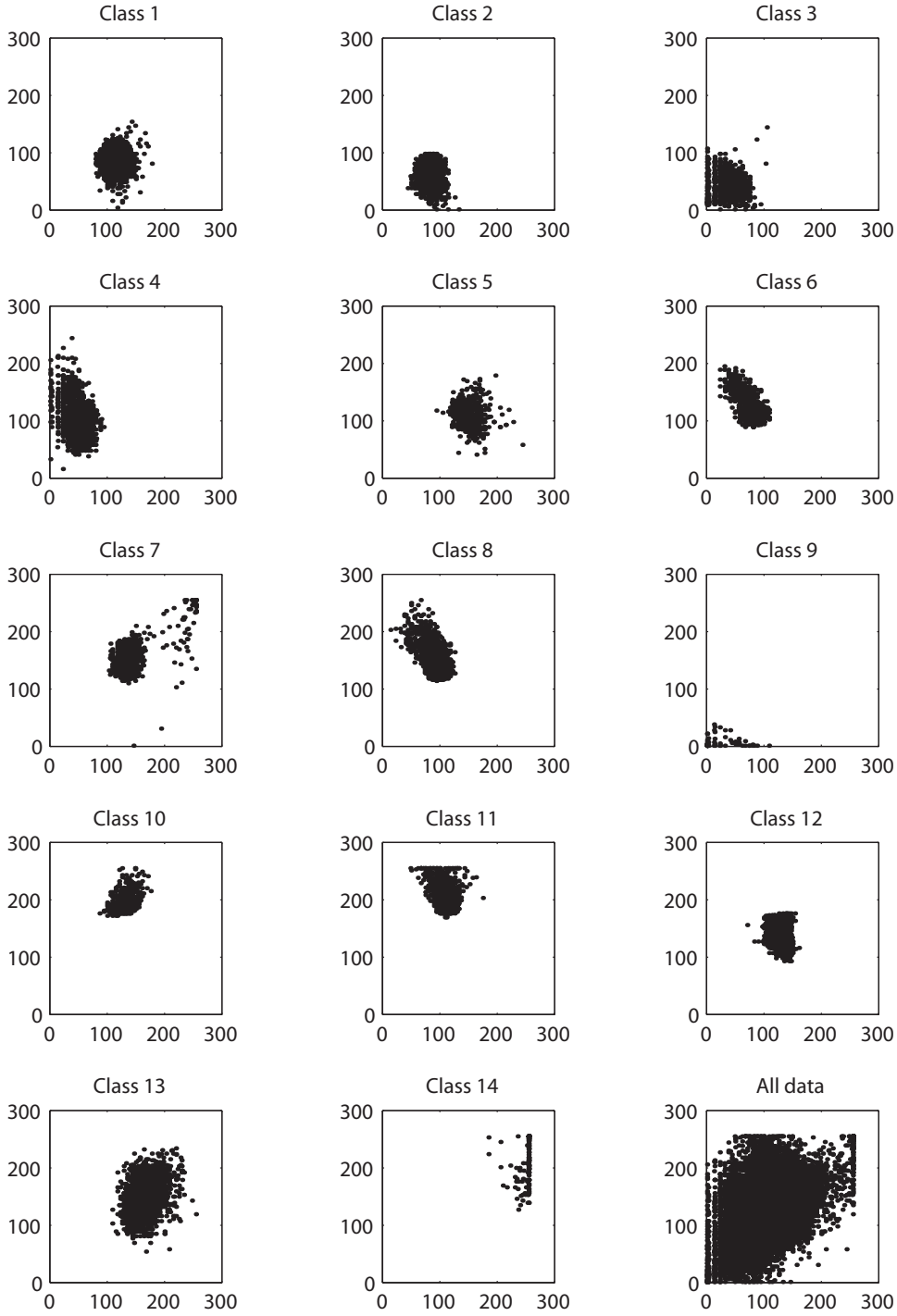


Figure 3.9: Location of the classes on the map: a large overlap of the classes can be observed since many classes are centered around different kinds of forest. Classes 9 (water) and 14 (alpine vegetation) , respectively, constitute two extremal classes at the borders.

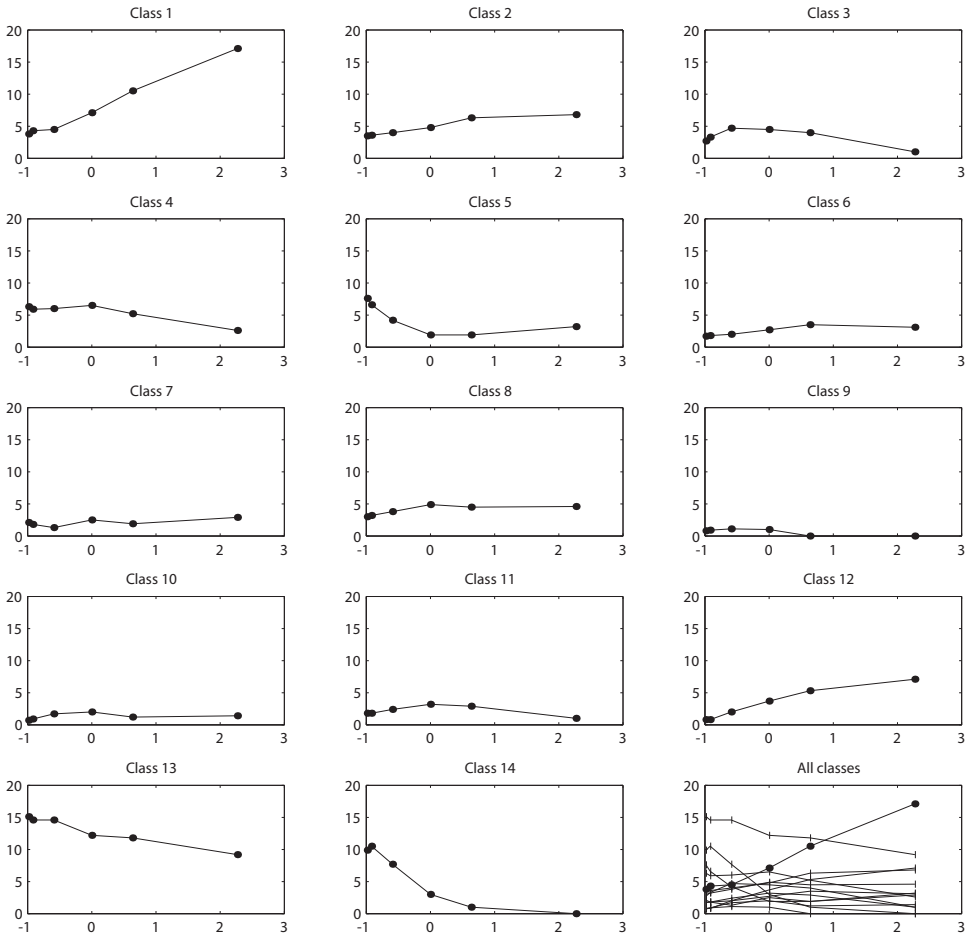


Figure 3.10: Number of neurons per class in dependence of the control parameter c.

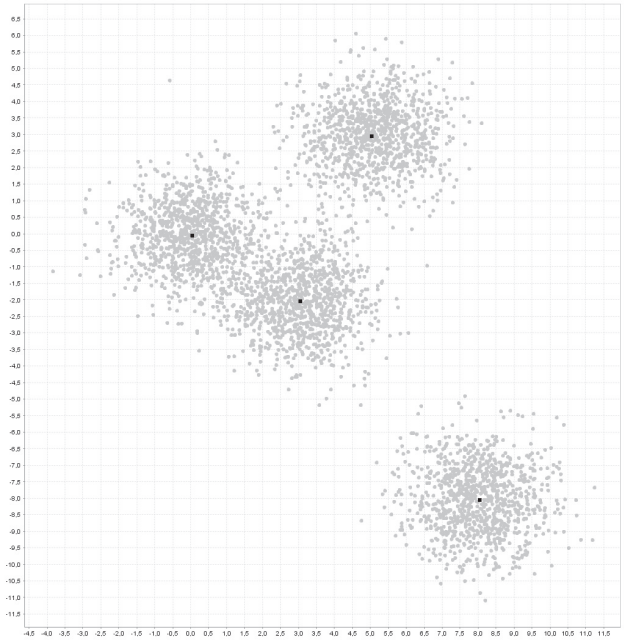


Figure 3.11: Mixture of four Gaussian clusters

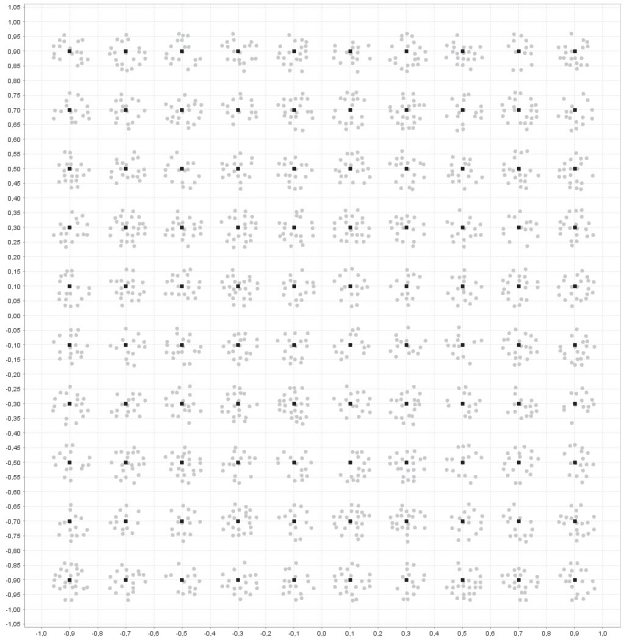


Figure 3.12: Checkerboard data

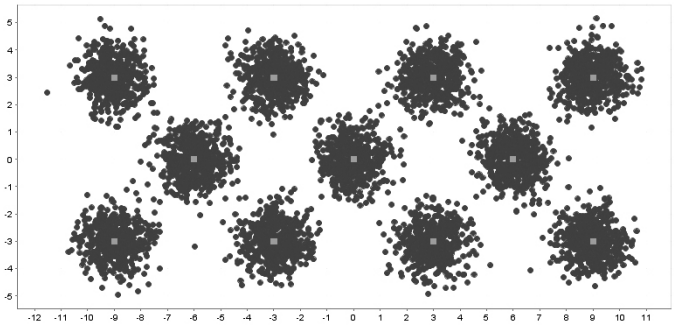


Figure 3.13: 11 clouds data set given by a mixture of Gaussians.

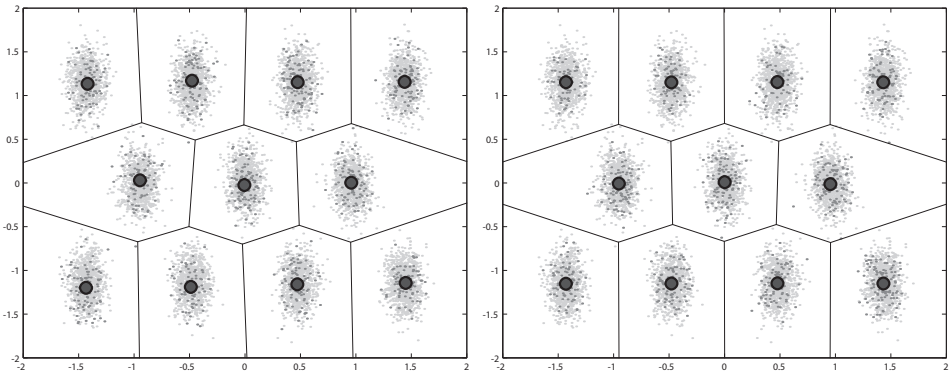
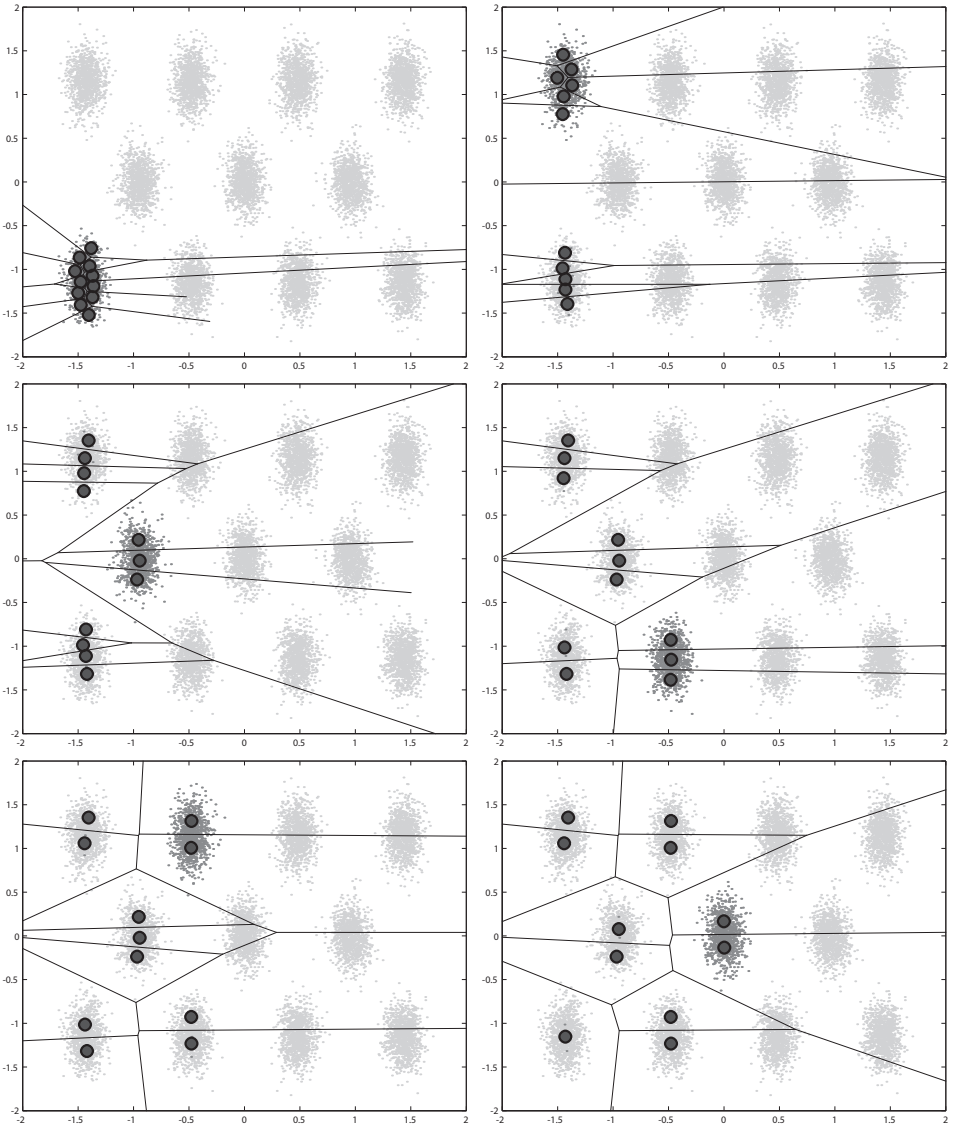


Figure 3.14: Results of patch clustering for i.i.d. data representations (randomly sampled from all clusters) after the first (left) and last (right) patch. The data presented in the patch are highlighted. Obviously, the cluster centers are almost fixed after the first patch due to the nature of the problem.



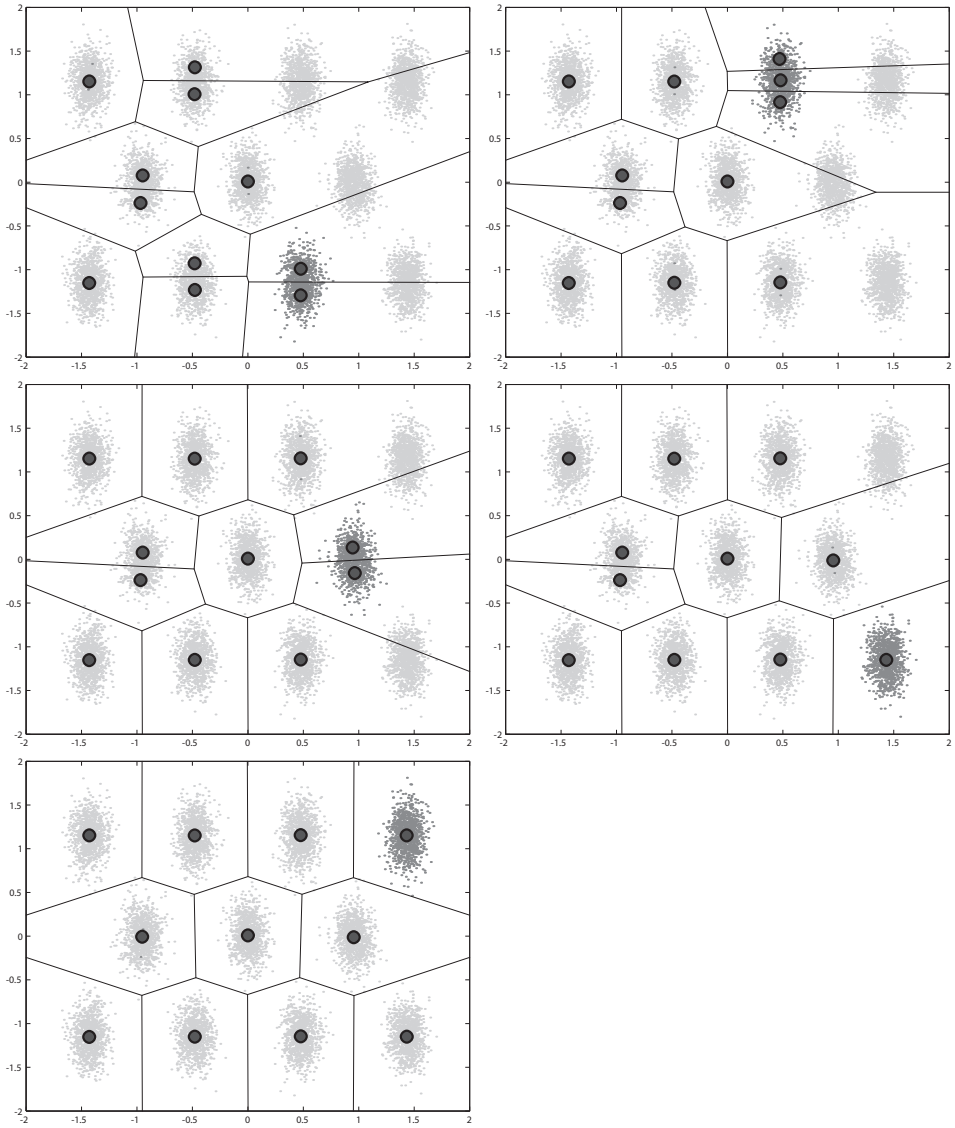
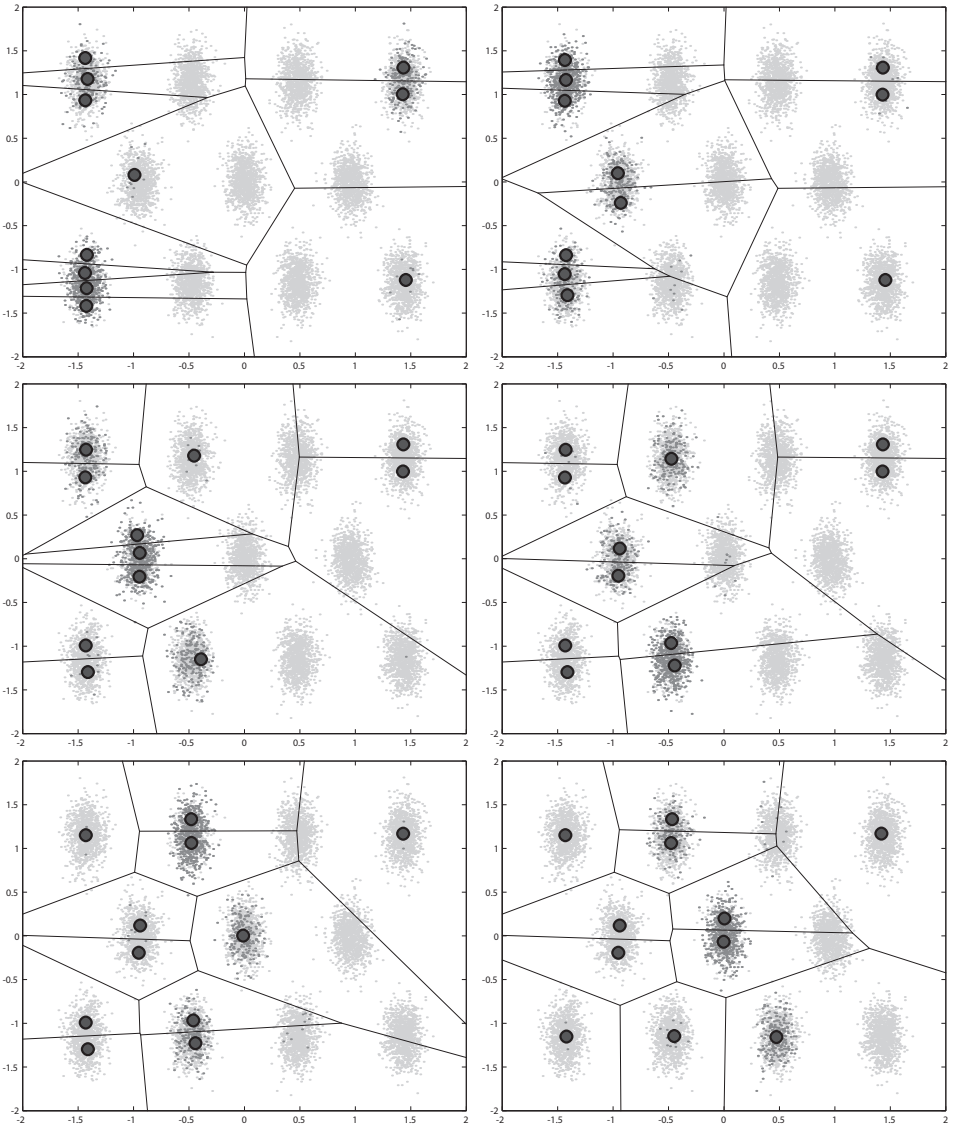


Figure 3.15: Results of patch clustering for data presented consecutively mode-wise according to its location in the data set (from left to right and top to bottom) after every patch. The data presented in each patch is highlighted. As can be clearly observed, the cluster centers vary in every patch, however, they are located at almost ideal position with respect to all already processed data, resulting in an almost perfect clustering after the final patch.



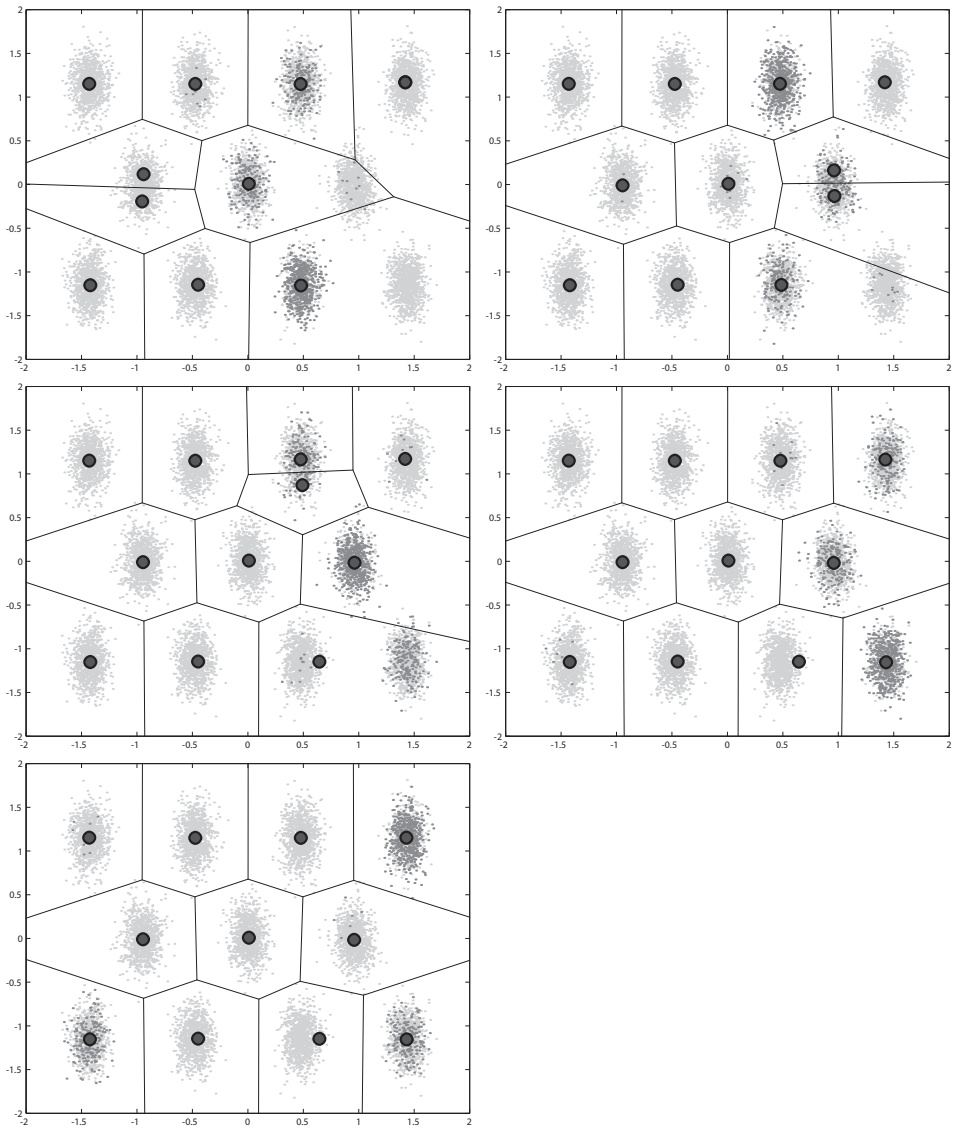


Figure 3.16: Results of patch clustering for data presented according to its location in the data set subject to a diffusion process (from left to right and top to bottom, mode-wise presentation with parts of the current, the two preceding, and the two subsequent clusters) after every patch. The data presented in each patch is highlighted. As before, the cluster centers vary in every patch. After processing the full data set, the cluster centers are located at almost ideal position with respect to all already processed data. Apart from a small shift of the center of cluster 7 which does not effect its receptive field in the given data set, an almost perfect clustering after the final patch is achieved.

Chapter 4

Topographic Mapping of Dissimilarity Data

In this chapter we are concerned with the processing of data that is in general not of Euclidean nature, that means an explicit embedding into Euclidean spaces is not available. The classical prototype-based topographic mapping methods introduced above cannot handle these data since their original formulations are based on vector operations in Euclidean spaces. In what follows, we will extend the principles of the above introduced prototype-based methods to handle those non-vectorial datasets. For the sake of clarity, the techniques are once again demonstrated exemplarily for Neural Gas, but a transfer of the ideas to Self-Organizing Maps is obvious and shall be sketched where necessary.

4.1 Introduction to Dissimilarity Data

As the name suggests, *dissimilarity* describes a certain degree of difference between objects that might be measured by arbitrary values. Dissimilarities are not required to be distances in a sense that they are metric. We only expect them to be symmetric and nonnegative, what is certainly a natural demand.[1] Also an object should be self-similar, so we require the dissimilarity to itself to be zero. We formalize dissimilarities by the following definition of dissimilarity measures.

Dissimilarity Measures Given a set of data points V from an arbitrary space. A function $d : V \times V \to \mathbb{R}$ is a *dissimilarity measure* on V, if it is non-negative, reflexive, and symmetric. That is, for all $v, v' \in V$ it holds

- $d(v, v') \geq 0$ (non-negativity),
- $d(v, v') = 0$ (reflexivity), and
- $d(v, v') = d(v', v)$ (symmetry).

For convenience, we denote the dissimilarity measure $d(v_i, v_j)$ defined on a finite dataset $V = \{v_1, v_2, \ldots, v_m\}$ also as d_{ij} in the following when it can be done without ambiguity. □

So by definition, the Euclidean metric is obviously a dissimilarity measures. Further examples of dissimilarity measures used in practical applications are, for instance,

[1] Although in the context of psychological experiments there are different opinions whether the demand of symmetry might be too restrictive.

- *alignment distances* from bioinformatics,
- *Levenshtein metric* from string processing,
- *Hamming distance* from information theory,
- *geodesic distance* from geometry,
- *Jaccard index* and *Tanimoto coefficient* from statistics, and
- *normalized compression distance* from algorithmic information theory.

Note that for a finite set $V = \{v_1, v_2, \ldots, v_m\}$ of data points the dissimilarity measure can be written as a *dissimilarity matrix* $D = (d_{ij})_{ij}$. Typically, we will use this representation in the following. The set of data points and the structure of the underlying mathematical space is in general unknown, in almost all cases only a dissimilarity matrix is given. For historical reasons, dissimilarity datasets are also called *relational data* (cf. Hathaway and Bezdek, 1994), because relations between objects are described.

It is to be understood here, that dissimilarity data does not necessarily originate from a Euclidean vector space. Even if given data stems from a metric space, like data gained from Levenshtein metric or alignment distances for instance, it is in general not isometrically embeddable into any Euclidean space (Matoušek, 2002; Indyk and Matoušek, 2004; Pekalska and Duin, 2005). So almost all of the above presented dissimilarity measures are in general non-Euclidean. It is still an active field of mathematical research which structures are isometrically embeddable into a Euclidean space and which are not, and if they are not, the question arises what are the bounds on the distortion that has to be accepted for any embedding (cf. Matoušek, 2007).

Gram Matrices Given a finite dataset $X \subset \mathbb{R}^{m \times d}$ from a Euclidean space, the corresponding Gram matrix $G = XX^T = (\langle x_i | x_j \rangle)_{ij}$ containing the pairwise standard inner products is always positive definite.

In the following, Gram matrices of indefinite inner products for non-Euclidean spaces are constructed and their eigenspectra are analyzed. It will turn out that dissimilarity datasets are characterized by the eigenspectra of their corresponding Gram matrices, namely the respective number e_+ and e_- of positive and negative eigenvalues indicate the ratio between Euclidean and non-Euclidean character of the data.

Let I_m denote the identity matrix of order m, $\mathbf{1}_m = (1, 1, \ldots, 1)^T$ the vector of order m, and let $J = I - \frac{1}{m} \cdot \mathbf{1}_m \mathbf{1}_m^T$. Furthermore, we denote the element-wise square operation on a matrix A by $A^{\star 2} = \left(a_{ij}^2\right)_{ij}$.

The corresponding *Gram matrix* G of a dissimilarity matrix D is then defined to be

$$G = -\frac{1}{2} J D^{\star 2} J. \tag{4.1}$$

The *inertia* of Gram matrix G is defined to be the triple (e_+, e_-, e_0) in which e_+, e_-, and e_0 are the respective number of positive, negative, and zero eigenvalues, counting algebraic multiplicities. The inertia is invariant under congruence transformations and it can be shown that if G has inertia (p, q, z) then there exists a

basis transformation with basis S such that $SGS^T = H$, written $G \simeq H$, where

$$H = \begin{pmatrix} I_p & & \\ & -I_q & \\ & & \mathbf{0}_z \end{pmatrix} \quad \text{(cf. Meyer, 2000).} \tag{4.2}$$

It should be mentioned here, that the corresponding Gram matrix G with inertia (p, q, z) of a given dissimilarity dataset, as an indefinite inner product matrix, can always be represented by a set of points X in a suitable Euclidean space $\mathbb{R}^{p+q}$ such that $G = ((x_i|Hx_j))_{ij} = XHX^T$, where $H \simeq G$ is defined as above, and $(.|.)$ denotes the standard inner product (cf. Gohberg et al., 2005). This representation leads to the theory of pseudo-Euclidean spaces (Goldfarb, 1984), that shall not be considered further within this work.

Later on in the experimental section, Table 4.2 shows the inertia of several non-Euclidean datasets that are used in the experiments. Moreover, the sorted eigenspectra of the Gram matrices can be visualized as it is done on page 82 to gain deeper insights into the structure of the non-Euclidean datasets.

Obviously, the prototype-based topographic mapping methods introduced in the previous chapter cannot handle dissimilarity data since they are based on vectorial updates. At first feel, a solution would be the prior embedding of given dissimilarity data into a Euclidean space accepting some distortion. But this approach is most of the times impracticable because embeddings of good quality are often computationally hard to get (cf. Matoušek, 2007).

Spread Transform An alternative approach proposed by Hathaway and Bezdek (1994) for their NERF c-means algorithm relies on a spread transformation of the dissimilarity matrices that converts them to Euclidean distance matrices by

$$D_\gamma = D + \gamma \cdot (\mathbf{1}_m \mathbf{1}_m^T - I_m)$$

for a suitable parameter $\gamma > 0$. Their method then operates directly on the generated Euclidean distances. However, as discussed later on, the spreading transformation alters the data space significantly in an unprofitable way.

It can be shown, see (Gower and Legendre, 1986), that the minimum spreading value to make a given dissimilarity matrix D Euclidean is given by the largest eigenvalue of the matrix

$$\begin{pmatrix} \mathbf{0}_m & -JD^{\star 2}J \\ -I_m & -2JDJ \end{pmatrix},$$

where I_m denotes the identity matrix of order m, and $J = I - \frac{1}{m} \cdot \mathbf{1}_m \mathbf{1}_m^T$.

This technique is also used by Roth et al. (2003) to transform dissimilarities into squared Euclidean distance matrices. Let $e_{\min}(.)$ denote the minimum eigenvalue of a given matrix. To make D squared Euclidean, the minimum spreading value is $2 \cdot e_{\min}(\frac{1}{2}JDJ)$.

In the experimental section, Table 4.3 shows the minimum spreading values for the used datasets to become squared Euclidean or Euclidean, respectively.

It should be noted here, that the spreading transformation distorts the original dissimilarities in a way that the structure of the data space can change considerably.

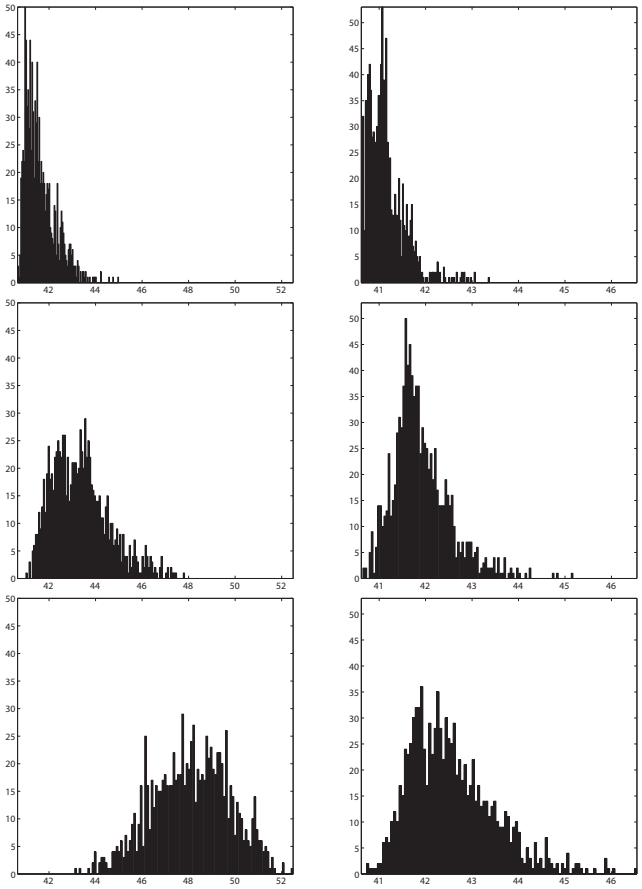


Figure 4.1: Implications of the Spread-Transformation: Local optima of cost function (4.3) reached by repeated runs of Relational Neural Gas (left column) and Deterministic Annealing (right column) on Cat Cortex dataset. First row is on untouched data, second row on squared-Euclidean spreaded data, and third row on Euclidean spreaded data. The parameters were 5 clusters, 1000 runs, 100 epochs RNG and 300 epochs DA

To empirically demonstrate this effect, we consider the intra-cluster distance, a well-known clustering measure, given by the cost function

$$E(\delta, d) = \sum_i \frac{1}{\sum_j \delta_{ij}} \cdot \sum_j \sum_k \delta_{ij}\delta_{ik}d_{jk} \tag{4.3}$$

with crisp cluster assignments $\delta_{ij} \in \{0, 1\}$. It was shown by Roth et al. (2003) that the spreading transformation does not alter the local optima of the cost function (4.3), only a constant is added. The global optimization of the cost function is NP-hard (Brucker, 1978), the methods used in the simulation are trying to find good local optima.

As it can be seen from the simulations (cf. Figure 4.1 and Figure 4.2), the algorithms cannot explore the space efficiently any longer. The distribution of local optima found by the algorithms over time is altered towards worse local optima. The algorithms even might fail to find the best local optima that they have found before the transformation.

Here, the need for methods arises that are able to process dissimilarity data

directly. In the subsequent sections, we will focus on topographic mapping techniques for dissimilarity datasets that are based on prototypes and perform exactly as desired.

Before going into details, we consider briefly a related stochastic approach dealing with dissimilarity data that is very popular. A stochastic formulation of Self-Organizing Maps for dissimilarity data was introduced by Graepel and Obermayer (1999). It is based on a cost function which was derived in analogy to the one that has been proposed by Luttrell (1994) in a general stochastic framework utilizing a folded Markov chain approach and comprising in particular Kohonen's Self-Organizing Maps. The optimization of the cost function is done by an Expectation Maximization algorithm (Jaynes, 1957a,b) in combination with a deterministic annealing scheme (Rose et al., 1990, 1992; Rose, 1998). A mathematical framework for deterministic annealing and mean-field approximations in the context of dissimilarity data was presented by Hofmann and Buhmann (1999). Successful applications of the stochastic approach have been made by Saalbach et al. (2005) to medical data measured by the Earth Mover's Distance utilizing a hyperbolic lattice structure.

4.2 Prototype-based Methods in Non-Euclidean Spaces

Up to this point, the introduced methods were based exclusively on Euclidean spaces. But unlike data originating from vector spaces, the important class of relational data is characterized only by pairwise dissimilarities $d(v_i, v_j)$ given for some underlying (and in general unknown) data points $v_i, v_j \in V$. Recall from section 4.1 that the only demands made on dissimilarity measures are non-negativity $d_{ij} \geq 0$, reflexivity $d_{ii} = 0$, and symmetry $d_{ij} = d_{ji}$. They are not necessarily metric by nature, and even if given data stems from a metric space, the space might not be isometrically embeddable into any Euclidean space at all (cf. Matoušek, 2002).

Obviously, all the classical processing methods as introduced above cannot handle relational data since their original formulation is based on prototype updates in a vector space. Now, we are restricted to a discrete space whose structure is characterized only by pairwise dissimilarities between its elements. In general, there are no linear combinations possible and potential prototypes cannot be settled in between the data points.

A possible way to circumvent these issues would be accepting a certain degree of distortion and embed the relational data into a Euclidean space anyway. Among other techniques, this non-isometrical embedding can be achieved by transforming the data using a spreading technique (cf. Hathaway and Bezdek, 1994). That way, we end up with Euclidean data points but as discussed above, the structure of the data space is changing and optimization is getting harder.

In this section, we will follow another way by keeping the given dissimilarities as they are and find better suited methods to work directly on them. It will turn out that, for this first approach, we also have to sacrifice a certain amount of accuracy. But later on, we will overcome these restrictions by introducing continuous prototype updates.

The key idea of the so-called *median* approach (Cottrell et al., 2006) is the restriction of prototype locations to given data points, such that prototype locations are well defined in the given space.

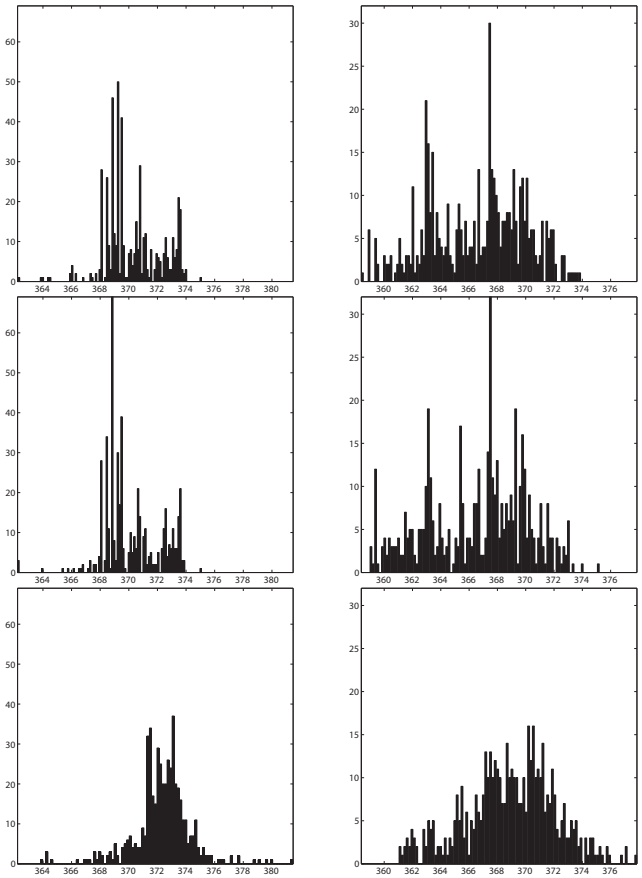


Figure 4.2: Implications of the Spread-Transformation: Local optima of cost function (4.3) reached by repeated runs of Relational Neural Gas (left column) and Deterministic Annealing (right column) on Protein dataset. First row is on untouched data, second row on squared-Euclidean spreaded data, and third row on Euclidean spreaded data. The parameters were 10 clusters, 500 runs, 100 epochs RNG and 250 epochs DA. Note that the dataset is almost squared Euclidean, so the distribution of local optima on untouched and squared Euclidean spreaded data is similar.

Median Neural Gas

Given a dataset V characterized solely by a dissimilarity matrix D, we introduce a finite collection of *median prototypes* $W = (w_i)_{i\in\{1,\dots,n\}}$, where for all $w_i \in W$ it holds $w_i = v_{k_i}$ with $v_{k_i} \in V$. In literature, techniques that restrict the location of prototypes to data positions are also called *exemplar-based* approaches (cf. e.g. Frey and Dueck, 2007).

In analogy to the original Neural Gas, we can define ranks on these median prototypes by setting

$$k_i(W, v) = |\{w_k \,:\, d(v, w_k) < d(v, w_i)\}|$$

with the same properties required in Definition 2.4.1. The cost function of *Median*

Neural Gas is then given by

$$E_\lambda(W) = \sum_{i=1}^{n} \sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot d(w_i, v_j)^2. \tag{4.4}$$

For the same reason as given for Batch Neural Gas, we now introduce hidden variables k_{ij} to replace the rank function and apply alternating optimization. As beforehand, the optimal assignments are given by the ranks $k_{ij} = k_i(W, v)$. The optimal prototype locations, however, are now determined by the so-called *generalized median* (cf. Kohonen and Somervuo, 2002)

$$w_i^* = \underset{v_k}{\operatorname{argmin}} \sum_j h_\lambda(k_{ij}) \cdot d(v_j, v_k)^2 \tag{4.5}$$

and can be found by exhaustive search. A sketch of the algorithm is given in table Algorithm 4.2. It has been shown in (Cottrell et al., 2006) that Median Neural Gas converges to a (local) optimum of the cost function.

Algorithm 4.1: Median Neural Gas

Input

Dissimilarity matrix $D \in \mathbb{R}^{m \times m}$

Begin

(* Initialize prototypes *)

Init w_i randomly for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Determine ranks (break ties deterministically)...

$k_i(W, v_j) = |\{ l \in \{1, \ldots, n\} \ : \ d(w_l, v_j) < d(w_i, v_j)\}|$

Update prototype locations by generalized median...

$w_i = \operatorname{argmin}_{v_k} \sum_j h_\lambda(k_{ij}) \cdot d(v_j, v_k)^2$

Decrease neighbourhood range...

$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$

endfor;

(* Return representative prototypes *)

Return w_i

End.

Despite our focus on Median Neural Gas, it should be noted here, that the median approach can easily be transferred to Self-Organizing Maps (El Golli et al., 2004; Cottrell et al., 2005).

Obviously, the median approach features the inherent drawback that only discrete adaptation steps can be performed which can dramatically reduce the representation quality of the mapping. Furthermore, the complexity of the median batch approaches is quadratic in the number of data points, what sets up a great obstacle when dealing with very large datasets. We will see later on in section 4.2 that this

obstacle can be overcome by an elegant patch processing method. For Batch SOM there are also faster approaches than exhaustive search (Conan-Guez et al., 2006), exploiting characteristics of the grid structure for pre-calculations, reusing earlier results, and applying early stopping among other intelligent techniques to speed up processing.

Another drawback of the median approach is its incapacity to separate prototypes that have been collapsing onto a single location in data space. A quick solution to the problem consists in adding some noise term to the update rule provoking a separation by chance. More sophisticated strategies to tackle the issue of collapsing prototypes in median approaches are addressed in (Rossi, 2007) utilizing a branch and bound approach.

Also the integration of additional information can be achieved the same way as done before. This is realized more precisely in the next section:

Supervised Median Neural Gas

Following Section 3.3, we can also incorporate additional information in form of labels given for each data point into the processing of Median Neural Gas (cf. Hammer et al., 2006a). The concept can also easily be migrated to other prototype-based median methods like Median SOM (cf. Hammer et al., 2006a).

Let a dataset $V = \{v_1, v_2, \ldots, v_m\}$ be given indirectly by a dissimilarity measure d on V. In addition, there shall be a collection of labels $\left(v_i^L\right)_{i\in\{1,\ldots,m\}}$ from $\mathbb{R}^c$ corresponding to an unary encoding of the classes the data points belong to.

Now, we define a new dissimilarity measure

$$d_\beta^*(v_i, v_j)^2 \;=\; (1-\beta)\cdot d(v_i, v_j)^2 \;+\; \beta \left\| v_i^L - v_j^L \right\|^2$$

based on d incorporating also the label space L. Replacing the original dissimilarity measure d in the original cost function (4.4) by d^* yields an extended cost function, namely

$$\begin{aligned} E_\lambda^*(W, W^L) &= \sum_{i=1}^{n}\sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot d_\beta^*(w_i, v_j)^2 \\ &= (1-\beta)\cdot \sum_{i=1}^{n}\sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot d(w_i, v_j)^2 \\ &\quad + \beta \cdot \sum_{i=1}^{n}\sum_{j=1}^{m} h_\lambda(k_i(W, v_j)) \cdot \left\| w_i^L - v_j^L \right\|^2 . \end{aligned} \tag{4.6}$$

This extended cost function can be optimized by alternating optimization treating the different parts separately as shown in Section 3.3.

Note that it can be shown in the same way as presented in (Hammer et al., 2006a) that the *Supervised Median Neural Gas* algorithm converges.

The introduced modification now allows to incorporate additional information into the learning process which turns the original unsupervised methods into supervised ones. This supervision helps to improve the quality of the results because the receptive fields are influenced by the underlying class information. Especially, when the prototypes are used as a classifier, this leads to a better generalization ability.

As observed beforehand, an important drawback of the median approaches is their quadratic time complexity in the size of input data. For the processing of

very large datasets, arising in numerous fields of applications today, this fact poses a severe limitation of applicability or even the uselessness of the median approaches.

In what follows, we are concerned with an alternative processing scheme that circumvents the quadratic time complexity of the median approaches.

Processing of Very Large Dissimilarity Datasets

One of the greatest challenges in data mining today (Yang and Wu, 2006), arising especially in the fields of web mining and bioinformatics, is the topographic mapping of very large dissimilarity datasets to explore their structure and gain information that otherwise would remain concealed, buried due to the sheer amount of data. These large non-Euclidean datasets cannot be held at once within random-access memory during computation, so the batch variants presented above like Median Self-Organizing Maps or Median Neural Gas cannot be applied.

Here, we present fast approximate (semi-)supervised versions of SOM and NG for non-Euclidean data that are able to handle large text datasets by a single pass technique. The introduced methods are based on patches that can be chosen in accordance to the size of the available random-access memory. These algorithms are running in constant space and linear time and provide different visualizations of the data space. Moreover, the patch technique opens the way for long-term learning and scalability, what is especially suited for information systems where the database is updated perpetually, e.g. in visual search engines etc.

For an illustration of one of the several issues to deal with, we show the storage needed for dissimilarity matrices due to their quadratic growth in the number of data points. The following table shall demonstrate the obvious demands that are posed by some real world large datasets.

Size of Dissimilarity Matrix (Double Precision)

n	*Size*	*Name of Dataset*
5000	190MB	Copenhagen Chromosome Dataset
10,000	763MB	
20,000	3.0GB	20 Newsgroup Dataset
50,000	18.6GB	
200,000	300.0GB	Large 13 Newsgroup Dataset

Unfortunately, also the time complexity of the Median methods is quadratic. To cope with very large datasets, it is therefore necessary to accelerate the methods and reduce their space usage. Fast variants of Median approaches that are able to handle very large datasets by a single pass technique based on patches chosen in accordance to the size of the available random-access memory were first introduced by Hasenfuss and Hammer (2008). In the upcoming section, we will introduce these variants on the basis of Median Neural Gas, whereby an extension to Median SOM is obvious.

Assume data are given as a dissimilarity matrix $D = (d_{ij})_{i,j=1,\ldots,m}$ with entries $d_{ij} = d(v_i, v_j)$ representing the dissimilarity of the datapoints v_i and v_j.

During processing of Patch Median NG, n_p disjoint patches of fixed size $p = \lfloor m/n_p \rfloor$ are taken from the dissimilarity matrix D consecutively,[2] where every patch

$$P_i = (d_{st})_{s,t=(i-1)\cdot p+1,\ldots,i\cdot p} \in \mathbb{R}^{p\times p}$$

[2]The remainder is no further considered here for simplicity. In the practical implementation the remaining datapoints are simply distributed over the first $(M - p \cdot n_p)$ patches.

is a submatrix of D.

The idea of the patch scheme is to add the prototypes from the processing of the former patch P_{i-1} as additional datapoints to the current patch P_i, forming an extended patch P_i^* to work on further.

The additional datapoints – the former prototypes – are weighted according to the size of their receptive fields, i.e. how many datapoints they have been representing in the former patch. Therefore, every datapoint v_j, as a potential prototype, is equipped with a multiplicity m_j, that is at first initialized with $m_j = 1$.

But unlike the situation of Patch NG in Euclidean space (Alex et al., 2009), where inter-patch distances can always be recalculated with help of the Euclidean metric, we are now dealing with an unknown mathematical space. We have to construct the extended patch from given dissimilarity data.

Let n be the fixed number of prototypes and let N_k denote the index set of prototypes of step k pointing onto elements of the dissimilarity matrix D. The extended patch P_i^* is then defined as

$$P_i^* = \left(\begin{array}{c|c} d(N_{i-1}) & d(N_{i-1}, P_i) \\ \hline d(N_{i-1}, P_i)^T & P_i \end{array} \right)$$

where

$$d(N_{i-1}) = (d_{st})_{s,t \in N_{i-1}} \in \mathbb{R}^{n \times n}$$
$$d(N_{i-1}, P_i) = (d_{st})_{s \in N_{i-1}, t=(i-1)\cdot p+1,\ldots,i\cdot p} \in \mathbb{R}^{n \times p}$$

denote the inter-distances of former prototypes and the distances between former prototypes and current patch points, respectively.

Now, the original Median Batch Neural Gas method must be modified to handle the weighted datapoints. Incorporating the multiplicities into the cost function yields

$$E(W) = \sum_{i=1}^{n} \sum_{j=1}^{m} h_\lambda(k_{ij}) \cdot m_j \cdot d(w_i, v_j)^2,$$

since every datapoint v_j is weighted with multiplicity m_j. The multiplicity $m(w_i)$ of a prototype $w_i \in W_t$ is given by

$$m(w_i) = \sum_{v \in P_t^* \cap V_i(W_t)} m(v),$$

the sum of multiplicities of all data points in the receptive field. The new update rule for the prototypes derived from the modified cost function is

$$w_i = \underset{v_k}{\operatorname{argmin}} \sum_{j=1}^{p} h_\lambda(k_{ij}) \cdot m_j \cdot d(v_j, v_k)^2.$$

Picking up the pieces, we yield Algorithm 4.2.

Algorithm 4.2: Patch Median Neural Gas (Hasenfuss and Hammer, 2008)

Begin

Cut the first Patch P_1

Apply Median Neural Gas on $P_1 \longrightarrow$ Prototypes N_1

Update Multiplicities m_j

Repeat for $i = 2, \ldots, n_p$

Cut patch P_i

Construct Extended Patch P_i^* using P_i and N_{i-1}

Apply modified Median Neural Gas with Multiplicities

$\longrightarrow$ Prototypes N_i

Update Multiplicities m_j

Return final prototypes N_{n_p}

End.

Analysis of Complexity

For an analysis of the complexity, we assume at first a fixed patch size p independent of the number of datapoints, as it would be the case when the patch size is chosen according to memory limitations. The algorithm then works only on $\mathcal{O}(\frac{m}{p} \cdot p^2) = \mathcal{O}(m \cdot p) = \mathcal{O}(m)$ entries of the dissimilarity matrix, compared to $\mathcal{O}(m^2)$ in the original Median NG method. Moreover, the algorithm uses at most $\mathcal{O}(p^2) = const$ entries at a specific point in time.

In case of a fixed patch size, also the time complexity is linear, because the Median NG step is $\mathcal{O}(p^2)$ what results in $\mathcal{O}(p^2 \cdot \frac{m}{p}) = \mathcal{O}(p \cdot m) = \mathcal{O}(m)$, an advantage compared to the $\mathcal{O}(m^2)$ time complexity of the original Median NG.

For a fixed dataset the total number of dissimilarity matrix elements touched during processing, compared to an imaginary application of Median NG, is decreasing in a reciprocal manner for an increasing number of patches. For sake of simplicity, we assume here w.l.o.g. that m is completely divisible by p. In detail, for a patch size p chosen independent of the size of the dataset m, the algorithm performs $\frac{m}{p} \cdot p \cdot n \log n + p^2 = m(n \log n + p) \in \mathcal{O}(m)$ steps. The original Median NG in contrast would require $m \cdot (n \log n + m) \in \mathcal{O}(m^2)$ processing steps. So the ratio

$$\frac{m(n \log n + p)}{m \cdot (n \log n + m)} = \frac{n \log n + \frac{m}{p_n}}{n \log n + m} = \text{const} + \text{const}' \cdot \frac{1}{p_n}$$

is a reciprocal function in the number of patches p_n.

Therefore the method does not only overcome the problem of limited memory, it also accelerates the processing of datasets, what might be useful in time critical applications.

These advantages in space and time complexity are obviously paid back by a loss of accuracy, what is also affirmed in the experiments later on, but the loss turns out to be small on realistic datasets. Theoretical bounds are not easily obtainable, this is still subject of ongoing research.

Obviously, the above argumentation does not hold if the patch size depends on the number of datapoints.

4.3 Continuous Prototype Updates for Dissimilarity Data

The restriction of prototypes to data point locations as used for the Median techniques may induce additional distortion particularly when the data space is sparse. This effect can easily be observed in comparative experiments versus the original Batch Neural Gas on Euclidean datasets. This drawback of the Median approaches motivates further research on the possibilities of continuous updates in non-vector spaces.

First steps towards a continuous update of prototypes were made by Hasenfuss et al. (2007) with the intention to represent prototypes as combinations of data points. Later on, the same authors introduced Relational Neural Gas (RNG) (Hammer and Hasenfuss, 2007) that overcomes the problem of discrete adaptation steps by using convex combinations of Euclidean embedded data points as prototypes. That way, distance calculations between prototypes and data can be done without any knowledge about the embedding. The idea of relational prototypes was extended to Relational SOM by Hasenfuss and Hammer (2007).

In what follows next, we are concerned with an overview about the relational approach. Later on, in the succeeding sections we shall discuss the concepts in more detail.

For that purpose, we assume that there exists a set of (in general unknown and presumably high dimensional) Euclidean points V such that $d_{ij} = \|v_i - v_j\|$ for all $v_i, v_j \in V$ holds, i.e. we assume there exists an (unknown) isometric embedding into a Euclidean space. The key observation is based on the fact that, under the assumptions made, the squared distances $\|w_i - v_j\|^2$ between (unknown) embedded data points and optimum prototypes can be expressed just in terms of known distances d_{ij}.

The prototypes in the relational approach are expressed as $w_i = \sum_j \alpha_{ij} v_j$ with $\sum_j \alpha_{ij} = 1$. Given a coefficient matrix $(\alpha_{ij}) \in \mathbb{R}^{n \times m}$ and a matrix $D^{\star 2} = \left(d_{ij}^2\right) \in \mathbb{R}^{m \times m}$ of squared distances, it then holds

$$\|w_i - v_j\|^2 = (\alpha_{i*} \cdot D^{\star 2})_j - \frac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T \tag{4.7}$$

what is verified later on in Section 4.3. It should be emphasized here, that thanks to (4.7) the squared distances can be calculated without explicit knowledge of points from V. Because of this fact, we are able to substitute all terms $\|w_i - v_j\|^2$ in Batch NG and Batch SOM by (4.7) and derive new update rules working only on distances. This allows to reformulate the batch optimization schemes in terms of relational data.

Note that, if an isometric embedding into Euclidean space exists, these schemes are exactly equivalent to the batch schemes and will yield identical results. Otherwise, the consecutive optimization scheme can still be applied on non-Euclidean data that is not isometrically embeddable into any Euclidean space.

By now, relational methods have been shown beneficial in the field of bioinformatics (cf. Hammer et al., 2007a), text processing (cf. Hasenfuss et al., 2008a), and visualization of discrete musical data (cf. Mokbel et al., 2009).

In the upcoming sections, we will introduce the relational approach in more details.

Representation of Distances

In what follows, a representation of distances between relational prototypes and data points is derived that does not include any knowledge about the data but the distance matrix. The section is based on the work of Hathaway and Bezdek (1994) who relied on earlier works of Torgerson (1958) and Young and Householder (1938).

Theorem 4.3.1 (Distances between Convex Combinations) Let $\mathcal{V}$ be a real inner-product space and $V = \{v_1, v_2, \ldots, v_m\}$ a finite subset of $\mathcal{V}$. Let $\|\star\| = \sqrt{\langle\star|\star\rangle}$ denote the canonical norm on $\mathcal{V}$. Furthermore, we denote the matrix of squared pairwise distances of V by

$$D^{\star 2} = \left(\|v_i - v_j\|^2\right)_{ij} \in \mathbb{R}^{m\times m}.$$

(a) For the squared pairwise distances between convex combinations over V, i.e. for elements

$$w_i = \sum_{j=1}^{m} \alpha_{ij} v_j \qquad \text{(where } \alpha_{ij} \geq 0 \text{ for all } i,j \text{ and } \sum_{j=1}^{m} \alpha_{ij} = 1 \text{ for all } i\text{)},$$

the following identity holds

$$\|w_i - w_j\|^2 \;=\; \alpha_{i*} D^{\star 2} \alpha_{j*}^T - \frac{1}{2}\alpha_{i*} D^{\star 2} \alpha_{i*}^T - \frac{1}{2}\alpha_{j*} D^{\star 2} \alpha_{j*}^T. \tag{4.8}$$

(b) For an element $v \in \mathcal{V}$, the distance to a convex combination $w_i = \sum_{j=1}^{m} \alpha_{ij} v_j$ over V is given by

$$\|v - w_i\|^2 \;=\; \sum_j \alpha_{ij} \|v - v_j\|^2 - \frac{1}{2}\alpha_{i*} D^{\star 2} \alpha_{i*}^T. \tag{4.9}$$

Remark Note that the squared distances to convex combinations over V can be expressed without using the elements from set V directly, only the squared distances and the coefficients are considered. Thus, the theorem provides a way to handle distance calculations for a set of unknown underlying points, only the squared distances have to be known. Since this situation occurs most of the times in real applications dealing with non-vectorial datasets, Theorem 4.3.1 turns out to be the key to sophisticated methods on dissimilarity data.

Proof. (a) In the first step, $\|w_i - w_j\|^2$ will be expressed by inner-products. Then, these inner-products will be replaced by distances of elements from V applying the technique of completing the square and using the identity $\|x - y\|^2 = \langle x - y|x - y\rangle = \langle x|x\rangle - 2 \cdot \langle x|y\rangle + \langle y|y\rangle$. Converting the terms into matrix notation using $D_{ij}^{\star 2} = \|v_i - v_j\|^2$ will complete the proof.

It directly follows

$$
\begin{aligned}
&\|w_i - w_j\|^2 = \|\sum_k \alpha_{ik} v_k - \sum_k \alpha_{jk} v_k\|^2 \\
&= \left\langle \sum_k \alpha_{ik} v_k \middle| \sum_k \alpha_{ik} v_k \right\rangle - 2 \cdot \left\langle \sum_k \alpha_{ik} v_k \middle| \sum_k \alpha_{jk} v_k \right\rangle + \left\langle \sum_k \alpha_{jk} v_k \middle| \sum_k \alpha_{jk} v_k \right\rangle \\
&= \underbrace{\sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_l \rangle}_{=: A(i,j)} - 2 \cdot \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_l \rangle + \underbrace{\sum_k \sum_l \alpha_{jk}\alpha_{jl} \langle v_k | v_l \rangle}_{=: B(i,j)} \\
&= A(i,j) + B(i,j) - 2 \cdot \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_l \rangle + \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle \\
&+ \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_l | v_l \rangle \underbrace{- \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle}_{=: C(i,j)} \underbrace{- \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_l | v_l \rangle}_{=: D(i,j)} \\
&= \sum_k \sum_l \alpha_{ik}\alpha_{jl} \|v_k - v_l\|^2 \; + \; (A(i,j) + C(i,j)) \; + \; (B(i,j) + D(i,j)) \\
&= \sum_k \sum_l \alpha_{ik}\alpha_{jl} \|v_k - v_l\|^2 - \frac{1}{2} \sum_k \sum_l \alpha_{ik}\alpha_{il} \|v_k - v_l\|^2 - \frac{1}{2} \sum_k \sum_l \alpha_{jk}\alpha_{jl} \|v_k - v_l\|^2 \\
&= \alpha_{i*} D^{\star 2} \alpha_{j*}^T - \frac{1}{2} \alpha_{i*} D^{\star 2} \alpha_{i*}^T - \frac{1}{2} \alpha_{j*} D^{\star 2} \alpha_{j*}^T.
\end{aligned}
$$

Because, it holds

$$
\begin{aligned}
&-(A(i,j) + C(i,j)) = \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle - \sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_l \rangle \\
&= \frac{1}{2} \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle - \sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_l \rangle + \frac{1}{2} \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle \\
&\overset{(*)}{=} \frac{1}{2} \cdot \left[\sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle - 2 \cdot \sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_l \rangle + \sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_l | v_l \rangle \right] \\
&= \frac{1}{2} \sum_k \sum_l \alpha_{ik}\alpha_{il} \left(\langle v_k | v_k \rangle - 2 \langle v_k | v_l \rangle + \langle v_l | v_l \rangle \right) \\
&= \frac{1}{2} \sum_k \sum_l \alpha_{ik}\alpha_{il} \|v_k - v_l\|^2 \\
&= \frac{1}{2} \alpha_{i*} D^{\star 2} \alpha_{i*}^T
\end{aligned}
$$

and similar $-(B(i,j) + D(i,j)) = \frac{1}{2} \alpha_{j*} D^{\star 2} \alpha_{j*}^T$.

In step $(*)$, the following identity was used,

$$
C(i,j) = \sum_k \sum_l \alpha_{ik}\alpha_{jl} \langle v_k | v_k \rangle = \sum_k \alpha_{ik} \langle v_k | v_k \rangle = \sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_k \rangle ,
$$

that holds because it is $\sum_t \alpha_{st} = 1$ for all $s \in \{1, 2, \ldots, n\}$ and therefore it can be left away or substituted arbitrarily in the above situation.

(b) Applying the same technique as in part (a), it is

$$
\begin{aligned}
&\|v - w_i\|^2 = \|v - \sum_j \alpha_{ij} v_j\|^2 = \sum_j \alpha_{ij} \left[\langle v | v \rangle - 2 \cdot \langle v | v_j \rangle + \langle v_j | v_j \rangle \right] \\
&+ \left(\sum_k \sum_l \alpha_{ik}\alpha_{il} \langle v_k | v_l \rangle - \sum_j \alpha_{ij} \langle v_j | v_j \rangle \right) = \sum_j \alpha_{ij} \|v - v_j\|^2 + (A(i,j) + C(i,j)) \\
&= \sum_j \alpha_{ij} \|v - v_j\|^2 - \frac{1}{2} \alpha_{i*} D^{\star 2} \alpha_{i*}^T.
\end{aligned}
$$

□

Corollary 4.3.2 (Simplified Representation) For the squared distance of an element v_k of a set $V = \{v_1, v_2, \ldots, v_m\}$ to any convex combination $w_i = \sum_{j=1}^{m} \alpha_{ij} v_j$ over V, the following simplified identity holds:

$$\|w_i - v_k\|^2 \;=\; (\alpha_{i*} \cdot D^{\star 2})_k - \frac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T \tag{4.10}$$

Proof. A point v_k of the set $V = \{v_1, v_2, \ldots, v_m\}$ can be seen as a convex combination $v_k = \sum_{j=1}^{m} \delta_{kj} v_j$, where δ_{ij} denotes the Kronecker delta.

Let $\delta_{k*} = (0, \ldots, 0, 1, 0, \ldots, 0)$ denote the corresponding coefficient vector. It then holds $\|w_i - v_k\|^2 = \|\sum_{j=1}^{m} \alpha_{ij} v_j - \sum_{j=1}^{m} \delta_{kj} v_j\| \overset{(4.8)}{=} \alpha_{i*} D^{\star 2} \delta_{k*}^T - \frac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T - \frac{1}{2} \cdot \delta_{k*} D^{\star 2} \delta_{k*}^T = (\alpha_{i*} \cdot D^{\star 2})_k - \frac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T$, since $(\delta_{k*} \cdot D^{\star 2}) \cdot \delta_{k*}^T = D_{k*}^{\star 2} \cdot \delta_{k*}^T = 0$ and also $D_{kk}^{\star 2} = 0$. □

Having proved the above theorems for real inner product space, naturally the question arises whether there are interesting real inner product spaces to deal with in Machine Learning. Since every normed vector space can be completed to a Banach space, every real inner product space (as a normed space) can be completed to a real Hilbert space. In turn, every finite dimensional real Hilbert space is norm-isomorphic to a Euclidean space of the same dimension. That means, every real inner product space is isometrically embeddable into a Euclidean space. For that reason, we will focus our further discussion on Euclidean spaces and non-Euclidean spaces, leaving away the subtleties of norm-isomorphic abstract spaces in between.

Relational Duals of Prototype-based Methods

In the following, we will establish a dual formulation of Batch Neural Gas for dissimilarity data. If given data is isometrically embeddable into a Euclidean space, the dual formulation will generate exactly the same results as Batch NG would do when applied to the corresponding points in Euclidean space.

Furthermore, if no Euclidean embedding exists, the algorithm can still be applied successfully.

Relational Neural Gas (Hammer and Hasenfuss, 2007)

Let there be unknown Euclidean data points $V = \{v_1, v_2, \ldots, v_m\}$ described only by squared distances $D^{\star 2} = \left(\|v_i - v_j\|_2^2\right)_{ij}$. For a dual formulation of Batch Neural Gas given only distances, we assume that *relational prototypes* are expressed as convex combinations of data points. Without loss of generality, we define

$$w_i = \sum_{j=1}^{m} \alpha_{ij} v_j \tag{4.11}$$

with coefficients $\alpha_{ij} \geq 0$ for all i,j and $\sum_{j=1}^{m} \alpha_{ij} = 1$ for all $i \in \{1, \ldots, n\}$.

Due to the nature of the update rule (cf. Eq. (3.5)), prototypes generated by any processing step of Batch Neural Gas are always situated in the convex hull of data, so the definition includes all potential prototype locations in the embedding space.

Also note, that relational prototypes are always defined on a set of points which we will refer to as the *spanning set*, because all possible relational prototype locations are situated in the convex hull spanned by those points.

The update rule of the dual formulation is now given by

$$\alpha_{ij} = \frac{h_\lambda(k_{ij})}{\sum_j h_\lambda(k_{ij})} \tag{4.12}$$

for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, because inserting Eq. (4.12) into the relational prototype definition Eq. (4.11) yields the same outcome as the original update rule Eq. (3.5), it is

$$w_i = \sum_{k=1}^{m} \alpha_{ik} v_k \quad \overset{(4.12)}{\Longrightarrow} \quad w_i = \sum_{k=1}^{m} \left(\frac{h_\lambda(k_{ik})}{\sum_j h_\lambda(k_{ij})} \right) v_k \;\hat{=}\; \text{Eq. (3.5)}.$$

Relational Neural Gas, the dual formulation of Batch NG, handles data given only as distances by performing an alternating optimization using the identity from equation (4.10) to determine the ranks and the dual update rule (4.12) to update the relational prototypes, see table Algorithm 4.3 for a sketch of the algorithm.

Algorithm 4.3: Relational Neural Gas

Input

Matrix $D^{\star 2} \in \mathbb{R}^{m \times m}$ of squared dissimilarities for (unknown) data points $v_1, \ldots, v_m$

Begin

(* Initialize relational prototypes $w_i = \sum_j \alpha_{ij} v_j$ *)

Init $\alpha_{ij} \in \mathbb{R}^{n \times m}$ randomly,

where $\sum_j \alpha_{ij} = 1$ for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Compute squared dissimilarities...

$d(w_i, v_j) = (\alpha_{i*} \cdot D^{\star 2})_j - \frac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T$

Determine ranks (break ties deterministically)...

$k_i(W, v_j) = |\{ l \in \{1, \ldots, n\} \; : \; d(w_l, v_j) < d(w_i, v_j) \}|$

Update prototype locations...

$\alpha_{ij} = h_\lambda(k_i(W, v_j)) / \sum_l h_\lambda(k_i(W, v_l))$

Decrease neighbourhood range...

$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$

endfor;

(* Return representative prototypes *)

Return α_{ij}

End.

Convergence Properties of Relational Neural Gas

In the Euclidean case the convergence properties transfer from the standard batch variants to the relational approach. That is given data points $V = \{v_1, v_2, \ldots, v_m\}$, a dissimilarity measure d, and let $D^{\star 2} = \left(d(v_i, v_j)^2\right)_{ij}$. If (V, d) is isometrically embeddable into a Euclidean space, Relational NG on $D^{\star 2}$ will generate exactly the same sequence $\{W_t\}_t$ of prototypes while processing as Batch NG does on V as discussed above.

Even if we replace the metric $\|x - y\|$ by a general dissimilarity measure $d(x, y)$ in all preceding formulas, i.e. a general dissimilarity dataset is given, the relational methods will still generate meaningful results as can be seen best from a dual formulation of the cost function (4.14) introduced in the following section. But it must be stated here that the relational methods might converge to a saddle point of the cost function, and not always end up in a local optimum like for Euclidean embeddable data. A theoretical framework of convergence and optimality in the non-Euclidean case is still subject of ongoing research and will not be considered here further.

Complexity Issues

Like the median techniques, the serious drawback of the relational approach is its quadratic time complexity. One solution utilizing a sparse prototype representation was presented by Rossi et al. (2007). It reduces time complexity by approximating relational prototypes by fewer spanning points and therefore slightly raises the quantization error. Another technique that also accelerates the relational methods is discussed in a subsequent section. It is based on patches of fixed size and is capable of processing very large datasets that do not fit into memory at once.

The space needed to store the prototypes of the relational methods is $\mathcal{O}(n \cdot m)$, what is higher compared to the standard approach.

Dual Cost Functions

As introduced in section 3.1, Batch Neural Gas is based on an alternating optimization scheme, alternately optimizing rank assignments and prototype locations. In the following section, we will demonstrate that the original cost function of BNG can be related to a dual formulation that can be interpreted as a dual constrained optimization problem of the intra-cluster distance. For converged prototypes of Batch Neural Gas the original cost function and the dual one with corresponding assignments have the same outcome. Moreover, the dual formulation also relates Neural Gas to Deterministic Annealing (Hofmann and Buhmann, 1999) which optimizes the dual formulation. We derive the dual formulation as follows:

For fixed ranks $k_i(W, v_j)$ the optimal prototypes w_i^* are given by

$$w_i^* = \frac{\sum_j h_\lambda(k_i(W, v_j)) v_j}{\sum_j h_\lambda(k_i(W, v_j))}. \tag{4.13}$$

For convenience, we abbreviate $k_i(W, v_j)$ in the following part by k_{ij}. Later on in our argumentation, k_{ij} will be considered as free parameters of the objective function.

Replacing w_i in the cost function (3.1) of BNG by the corresponding optimal

prototypes (4.13) of given k_{ij} directly yields

$$
\begin{aligned}
E(W^*, k_{ij}) &= \frac{1}{2}\sum_i\sum_j h_\lambda(k_{ij})\|w_i^* - v_j\|^2 \overset{(4.13)}{=} \frac{1}{2}\sum_i\sum_j h_\lambda(k_{ij})\left\|\frac{\sum_l h_\lambda(k_{il})v_l}{\sum_l h_\lambda(k_{il})} - v_j\right\|^2 \\
&= \frac{1}{2}\sum_i\sum_j \frac{h_\lambda(k_{ij})}{\left(\sum_l h_\lambda(k_{il})\right)^2}\left\|\sum_l h_\lambda(k_{il})v_l - \sum_l h_\lambda(k_{il})v_j\right\|^2 \\
&= \frac{1}{2}\sum_i\sum_j \frac{h_\lambda(k_{ij})}{\left(\sum_l h_\lambda(k_{il})\right)^2}\cdot \\
&\quad \left[\left\|\sum_l h_\lambda(k_{il})v_l\right\|^2 - 2\cdot\left\langle\sum_l h_\lambda(k_{il})v_l\middle|\sum_l h_\lambda(k_{il})v_j\right\rangle + \left\|\sum_l h_\lambda(k_{il})v_j\right\|^2\right] \\
&= \frac{1}{2}\sum_i \frac{1}{\left(\sum_l h_\lambda(k_{il})\right)}\cdot\left[\sum_s\sum_t h_\lambda(k_{is})h_\lambda(k_{it})\cdot\left(\langle v_s|v_t\rangle - 2\cdot\langle v_s|v_t\rangle + \langle v_t|v_t\rangle\right)\right] \\
&= \frac{1}{2}\sum_i \frac{1}{\left(\sum_l h_\lambda(k_{il})\right)}\cdot\left[\sum_s\sum_t h_\lambda(k_{is})h_\lambda(k_{it})\cdot\left(\frac{1}{2}\langle v_s|v_s\rangle - \langle v_s|v_t\rangle + \frac{1}{2}\langle v_t|v_t\rangle\right)\right] \\
&= \frac{1}{4}\sum_i \frac{1}{\left(\sum_l h_\lambda(k_{il})\right)}\cdot\left[\sum_s\sum_t h_\lambda(k_{is})h_\lambda(k_{it})\cdot\|v_s - v_t\|^2\right].
\end{aligned}
$$

Thus, the dual of the cost function (3.1) of Batch NG is given by

$$
\frac{1}{4}\sum_i \frac{1}{\left(\sum_l h_\lambda(k_{il})\right)}\cdot\left[\sum_s\sum_t h_\lambda(k_{is})h_\lambda(k_{it})\cdot\|v_s - v_t\|^2\right]. \tag{4.14}
$$

From the construction it directly follows that for given prototype locations w_i and their corresponding optimal assignments k_{ij} the dual cost function (4.14) has the same value as the original one (3.1). Note, that the dual formulation is based solely on the assignments k_{ij} and no longer on the distances. The assignments k_{ij} can now be seen as free parameters of a new objective function with the constraint that $\{k_{1j}, k_{2j}, \ldots, k_{nj}\}$ is a permutation of $\{0, 1, \ldots, n-1\}$ for every $j \in \{1, 2, \ldots, m\}$.

As a side effect, the dual cost function constitutes also a descriptive error measure for dissimilarity data to replace the quantization error in non-vectorial spaces. This is an advantage, since only the assignments are needed that could be derived directly from the relational coefficients α_{ij} without utilizing the prototype locations w_i.

Incorporating Additional Information

Following Section 3.3, we can also incorporate additional information in form of labels given for each data point into the relational variants (Hammer and Hasenfuss, 2007). As a showcase, we will concentrate on Relational Neural Gas but incorporating additional information into Relational SOM is similar.

Let a dataset $V = \{v_1, v_2, \ldots, v_m\}$ be given indirectly by a dissimilarity measure d on V. In addition, there shall be a label $v_j^L \in \mathbb{R}^c$ attached to each datapoint v_j corresponding to an unary encoding of the classes the data points belong to.

In the context of relational methods as explained above, we assume that the dataset V is isometrically embeddable into a Euclidean space $X = \mathbb{R}^k$ for some k. We define a new dissimilarity measure

$$
d_\beta^*(x_i, x_j)^2 \;=\; (1-\beta)\cdot d(x_i, x_j)^2 \;+\; \beta\left\|x_i^L - x_j^L\right\|^2
$$

on the underlying space X based on d that also incorporates the label space L. It is based on the original dissimilarity measure d and the Euclidean metric in label space L, and that way connects data space X and label space L. Moreover, the new dissimilarity measure d^*_β depends on a weighting parameter $\beta \in [0, 1]$, which shall allow us later on to control the influence of the label information on the learning process.

In the relational approach, distances between the relational prototypes, between relational prototypes and data points, and also between data points regarded as convex combinations are according to (4.8) given by

$$d(x_i, x_j)^2 \;=\; \alpha_{i*} D^{\star 2} \alpha_{j*}^T - \frac{1}{2}\alpha_{i*} D^{\star 2} \alpha_{i*}^T - \frac{1}{2}\alpha_{j*} D^{\star 2} \alpha_{j*}^T,$$

where $D^{\star 2}$ is the matrix of squared dissimilarities.
Using this expression to calculate the dissimilarity measure d^* leads to Algorithm 4.3, introduced by Hammer and Hasenfuss (2007).

Algorithm 4.4: Supervised Relational Neural Gas

Input

Matrix $D^{\star 2} \in \mathbb{R}^{m \times m}$ of squared dissimilarities for (unknown) data points $v_1, \ldots, v_m$

Labels $\left\{v_1^L, v_2^L, \ldots, v_m^L\right\} \subset \mathbb{R}^c$

Begin

(* Initialize relational prototypes $w_i = \sum_j \alpha_{ij} v_j$ *)

Init $\alpha_{ij} \in \mathbb{R}^{n \times m}$ randomly,

where $\sum_j \alpha_{ij} = 1$ for all $i \in \{1, \ldots, n\}$ and $\lambda_0 = n/2$, $\lambda = \lambda_0$

(* Repeat for a given number of epochs... *)

for $t := 1$ to *epochs* **do**

Compute squared dissimilarities...

$$d^*_\beta(w_i, v_j) = (1-\beta) \cdot \left((\alpha_{i*} \cdot D^{\star 2})_j - \tfrac{1}{2} \cdot \alpha_{i*} D^{\star 2} \alpha_{i*}^T\right) \;+\; \beta \left\| w_i^L - v_j^L \right\|^2$$

Determine ranks (break ties deterministically)...

$$k_i(W, v_j) = \left| \left\{ l \in \{1, \ldots, n\} \;:\; d^*_\beta(w_l, v_j) < d^*_\beta(w_i, v_j) \right\} \right|$$

Update prototype locations...

$$\alpha_{ij} = h_\lambda(k_i(W, v_j)) / \textstyle\sum_l h_\lambda(k_i(W, v_l))$$

Update prototype labels...

$$w_i^L = \textstyle\sum_j h_\lambda(k_{ij}) \cdot v_j^L / \sum_j h_\lambda(k_{ij})$$

Decrease neighbourhood range...

$$\lambda = \lambda_0 \cdot (0.01/\lambda_0)^{t/epochs}$$

endfor;

(* Return representative prototypes and prototype labels *)

Return α_{ij} and w_i^L

End.

With the introduced modification it is now possible to incorporate additional information into the learning process which turns the original unsupervised meth-

ods into supervised ones. This supervision helps to improve the quality of the results because the receptive fields are influenced by the underlying class information. Especially, when the prototypes are used as a classifier, this leads to a better generalization ability.

In what follows, another interesting modification of the relational approach is introduced that is utilizing the popular *kernel* technique. Particularly complex structured data can be treated by this technique.

Kernelizing the Relational Methods

Prototype based methods in Euclidean spaces are somewhat restricted by the linear boundaries between the Voronoi regions generated by the prototypes (Aurenhammer, 1991; Graf and Luschgy, 2000). Although non-linear structures in the data can be approximated by using more prototypes, this might not be satisfying. Here the kernel approach comes into play, a very popular approach that gained a lot of attention in the last decade. By linearizing non-linearities in a kernel-defined feature space, kernelized methods can handle various complex structured data. The reader has probably heard of kernels in the context of Support Vector Machines where they are mainly utilized, but there are a variety of other kernelized methods, e.g. Kernel PCA (Schölkopf et al., 1998), Kernelized NERF c-Means (Hathaway et al., 2005), Online Kernel SOM (MacDonald and Fyfe, 2000), Batch Kernel SOM (Villa and Rossi, 2007), Online Kernel NG (Qinand and Suganthan, 2004), and the like. For a recent overview see the survey of Filippone et al. (2008). In what follows we are concerned with kernelizing the above introduced relational methods.

Let us first introduce briefly the principles and basic terminology used in the kernel approach. For an detailed introduction to kernel methods see e.g. the excellent book of Shawe-Taylor and Cristianini (2004).

The key idea of the kernel approach is a mapping $\phi : \mathcal{V} \supseteq V \ni v \longmapsto \phi(v) \in \mathcal{F}$ of input data V from input space $\mathcal{V}$ into an (high-dimensional) inner product space $\mathcal{F}$ which is called the *feature space*. The motivation for such an embedding comes with the hope that non-linear input data is linearized in feature space making it more accessible for algorithmic approaches. Kernelized methods process the embedded data points in feature space utilizing only the pairwise inner products $\langle \cdot | \cdot \rangle_{\mathcal{F}}$. The clou of the kernel approach, also known as the *kernel trick*, is that there is no need to determine the embedding ϕ and the inner product $\langle \cdot | \cdot \rangle_{\mathcal{F}}$ explicitly, what would often be a costly operation for complex data structures. The efficient computation of the inner product in feature space can be done indirectly by applying a *kernel function* $k : V \times V \to \mathcal{F}$, satisfying $k(v, v') = \langle \phi(v) | \phi(v') \rangle$ for all $v, v' \in V$, on the original data points from input space.

Hence, to kernelize a method, it has to be reformulated in a way that it is dealing only with pairwise inner products instead of any norm or metric. For instance, this can be done by utilizing the canonical representations $\|v\| = \sqrt{\langle v | v \rangle}$ or $d(v, v') = \|v - v'\| = \sqrt{\langle v - v' | v - v' \rangle}$, respectively (cf. Meyer, 2000). That way, the method is able to operate directly on any inner product space, whereby finite datasets would be given by Gram matrices. In a last step, the inner product can be substituted by an arbitrary kernel function enabling the method to work efficiently on a (possibly high-dimensional and hopefully linearized) feature space. Here, a finite dataset can be given as a *kernel matrix* $(k(v_i, v_j))_{ij}$ generated by pairwise applications of a kernel function $k(\cdot, \cdot)$ on the input data V, what is nothing else than pairwise inner products $\langle \phi(v_i) | \phi(v_j) \rangle_{\mathcal{F}}$ from a corresponding feature space $\mathcal{F}$, and hence a Gram matrix.

As it can be seen, kernelizing a method transforms it in a sense into an universal method able to process arbitrary complex structured data via kernel functions. Kernel functions proposed for complex structured data are, for instance, Edit distance-based kernel functions for strings and graphs (Neuhaus and Bunke, 2006), Fisher Kernel (Saunders et al., 2003) and String Kernel (Lodhi et al., 2002) for text processing, Graph Kernel (Kashima et al., 2004; Bach, 2008), Kernel on pointsets (Parsana et al., 2008), and Alignment Kernel from bioinformatics (Qiu et al., 2007). General approaches dealing with kernelizing of dissimilarity measures were proposed by Pekalska et al. (2001), and Haasdonk and Bahlmann (2004).

Now kernelizing of the relational methods can be accomplished as discussed above by replacing the norm in the distance calculations by the corresponding inner-product, enabling the method to operate in a feature space. Then the inner product can be substituted by a kernel function circumventing the need to calculate mappings into feature space. In the following we will derive a representation of squared distances that just depends on entries of a given kernel matrix. These squared distances between relational prototypes and data points are part of all introduced relational algorithms. Hence, the substitution provides kernelized relational variants, namely Kernelized Relational k-Means, KRSOM, and KRNG (Hammer and Hasenfuss, 2007).

Kernelized Distance Representation Assume that there is a finite dataset $V = \{v_1, v_2, \ldots, v_m\} \subseteq \mathcal{V}$ given. Let $\mathcal{F}$ be a real inner-product space with inner product $\langle \cdot | \cdot \rangle_{\mathcal{F}}$ and $\phi : \mathcal{V} \to \mathcal{F}$ a mapping as defined above. We denote the corresponding Gram matrix of V by $K = (k_{ij})_{ij}$ where $k_{ij} = \langle \phi(v_i) | \phi(v_j) \rangle_{\mathcal{F}}$. Assume further that prototypes are convex combinations of projected points in feature space. Then the identity

$$\|w_i - \phi(v_j)\|^2 \;=\; k_{jj} - 2\alpha_{i*}k_{j*}^T + \alpha_{i*}K\alpha_{i*}^T \tag{4.15}$$

holds for any convex combination $w_i = \sum_{j=1}^m \alpha_{ij}\phi(v_j)$ over $\mathcal{F}$.

Proof. It is easy to verify, that

$$\begin{aligned}
&\|w_i - \phi(v_j)\|_{\mathcal{F}}^2 = \\
&\langle \phi(v_j) | \phi(v_j) \rangle - 2\textstyle\sum_s \alpha_{is} \langle \phi(v_j) | \phi(v_s) \rangle + \sum_s \sum_t \alpha_{is}\alpha_{it} \langle \phi(v_s) | \phi(v_t) \rangle \\
&= k_{jj} - 2\textstyle\sum_s \alpha_{is}k_{js} + \sum_s \sum_t \alpha_{is}\alpha_{it}k_{st} = k_{jj} - 2\alpha_{i*}k_{j*}^T + \alpha_{i*}K\alpha_{i*}^T.
\end{aligned}$$

□

Hence in this section we have sketched how to kernelize Relational Neural Gas and Relational SOM. First successful applications of Kernelized Relational SOM have been made by Boulet et al. (2008) to social network analysis. In the following section we are concerned with magnification control for relational methods to pave the way for outlier suppression, information-optimal transfer, rare event emphasis, and the like also in the relational approach.

4.4 Magnification Control for Relational Methods

In this section we will explore the possibility of magnification control for the relational methods and show how to transfer the ideas from Section 3.2 in the context of Batch Neural Gas. Especially the behavior of magnification control on non-Euclidean datasets is of great interest, because the established theory does not cover this case. It shall be demonstrated empirically that magnification control is

also possible in the non-Euclidean case. The theoretical analysis seems to be difficult or even impossible by adapting the established mathematical techniques and proofs.

As demonstrated by Hasenfuss et al. (2008b), the localized learning technique can easily be transferred to RNG by integrating the local density – in analogy to Section 3.2 – into the prototype updates rule (4.12) as follows

$$\alpha_{ij} = (h_\lambda(k_i(v_j)) \cdot p(v_j)^m) \,/\, \sum_j (h_\lambda(k_i(v_j)) \cdot p(v_j)^m)\,. \tag{4.16}$$

If a Euclidean embedding of data points exists, this learning rule is equivalent to localized learning for Batch Neural Gas (3.13) as it can be seen by inserting the rule (4.16) into the prototype representation $w_i = \sum_j \alpha_{ij} v_j$. Thus, the theoretical guarantees as derived in (Hammer et al., 2007b) hold for this case.

For the non-Euclidean case, the theoretical effect of the localized learning rule is not clear a priori, this issue is still in the focus of ongoing work. We are nevertheless able to see from the experiments that magnification control in this case is surely possible.

Note that, for optimum α_{ij}, the cost function is equivalent to the extended relational dual cost function

$$\frac{1}{2} \cdot \sum_i \sum_{l,l'} h_\lambda(k_{il}) h_\lambda(k_{il'}) p(v_l)^m p(v_{l'})^m d^2_{ll'} \Big/ \sum_{l''} h_\lambda(k_{il''}) p(v_{l''})^m, \tag{4.17}$$

which measures the dissimilarities of data points assigned to the same clusters, weighted according to $p(v_l)^m$. The denominator accounts for the fact that the size of clusters per se is not important.

Obviously, the control parameter m allows to control the relevance of the value $d^2_{ll'}$ of data points v_l in certain regions of the data space. Assume $p(v_l)$ measures the relative number of similar points (or local data density, if defined). Then a control parameter $m > 1$ emphasizes regions which contain a large number of pairwise similar data points, whereas $m < 1$ emphasizes regions with only few pairwise similar points.

Hence, the presented modification of the relational methods paves the way to better control of their behaviour, what comes in handy for the problem of outlier suppression and emphasis of rare events, and what could also be utilized in a framework allowing for user interactions to magnify local areas of the data space in real-time data inspection.

As discussed above, the major drawback of the relational methods is their quadratic time and space complexity that prevents the usage of the methods in many real-world applications today. In what follows, we present a technique to process also very large non-Euclidean datasets, meaning hundred of thousands or even millions of objects, by relational methods.

4.5 Processing of Very Large Dissimilarity Datasets

As discussed before in Section 4.2, very large dissimilarity datasets require special processing schemes for Median methods due to their demands on space and processing time, what is also true for Relational approaches.

In the following, we will discuss a special processing scheme for Relational Neural Gas that was introduced by Hasenfuss et al. (2008c). The presented scheme can easily be migrated to Relational SOM as done by Hasenfuss et al. (2008a).

Assume as before that data are given as a dissimilarity matrix $D = (d_{ij})_{i,j=1,\ldots,m}$ with entries $d_{ij} = d(v_i, v_j)$ representing the dissimilarity of the datapoints v_i and v_j. During processing of Patch Relational NG, n_p patches of fixed size $p = \lfloor m/n_p \rfloor$ are cut consecutively from the dissimilarity matrix D,[3] where every patch

$$P_i = (d_{st})_{s,t=(i-1)\cdot p+1,\ldots,i\cdot p} \in \mathbb{R}^{p\times p}$$

is a submatrix of D centered around the matrix diagonal.

The idea of the original patch scheme is to add the prototypes from the processing of the former patch P_{i-1} as additional datapoints to the current patch P_i, forming an extended patch P_i^* which includes the previous points in the form of a compressed statistics. The additional datapoints – the former prototypes – are weighted according to the size of their receptive fields, i.e. how many datapoints do they represent in the former patch. To implement this fact, every datapoint v_j is equipped with a multiplicity m_j, which is initialized with $m_j = 1$ for data points from the training set and it is set to the size of the receptive fields for data points stemming from prototypes. This way, all data are processed without loss of previous information which is represented by the sufficient statistics. Moreover, in contrast to dynamic approaches such as (Prudent and Ennaji, 2005) the number of prototypes can be fixed a priori.

Unlike the situation of original Patch NG (Alex et al., 2009), where prototypes can simply be converted to datapoints and the inter-patch distances can always be recalculated using the Euclidean metric, the situation becomes more difficult for relational clustering. In Relational NG prototypes are expressed as convex combinations of unknown Euclidean datapoints, only the distances can be calculated. Moreover, the relational prototypes gained from processing of a patch cannot be simply converted to datapoints for the next patch. They are defined only on the datapoints of the former patch. To calculate the necessary distances between these prototypes and the datapoints of the next patch, the distances between former and next patch must be taken into account, as shown in Section 4.3. But that means touching all elements of the upper half of the distance matrix at least once during processing of all patches, what foils the idea of the patch scheme to reduce computation and memory-access costs.

In this contribution, another way is proposed. In between patches not the relational prototypes itselves but representative datapoints obtained from a so called k-approximation are used to extend the next patch. As for standard patch clustering, the points are equipped with multiplicities. On each extended patch a modified Relational NG is applied taking into account the multiplicities.

k-Approximation

Assume there are given n relational prototypes by their coefficient matrix $(\alpha_{ij}) \in \mathbb{R}^{n\times m}$ defined on Euclidean datapoints V. These prototypes are taken after convergence of the Relational NG method, i.e. these prototypes are situated at optimal locations.

As can be seen from the update rule (4.12), after convergence in the limit $\lambda \to 0$ it holds

$$\alpha_{ij} \longrightarrow \left\{ \begin{array}{ccc} 1/|R_i| & : & v_j \in R_i \\ 0 & : & v_j \notin R_i \end{array} \right. , \text{ because } \left\{ \begin{array}{l} h_\lambda(k_{ij}) = 1 \text{ for } v_j \in R_i \\ h_\lambda(k_{ij}) \to 0 \text{ for } v_j \notin R_i \end{array} \right\},$$

where $R_i = \{v_j \in V : \|w_i - v_j\| \leq \|w_k - v_j\| \text{ for all } k\}$ denotes the receptive field of prototype w_i. That means, in the limit only datapoints from the receptive fields

[3] The remainder is no further considered here for simplicity. In the practical implementation the remaining datapoints are simply distributed over the first $(M - p \cdot n_p)$ patches.

have positive coefficients and equally contribute to the winning prototype that is located in the center of gravity of its receptive field.

A *k-approximation* of an optimal relational prototype w_i is a subset $R' \subseteq R_i$ with $|R'| = \min\{k, |R_i|\}$ such that $\sum_{r' \in R'} \|w_i - r'\|^2$ is minimized. That means, we choose the k nearest points from the receptive field of a prototype as representatives. If there are less than k points in the receptive field, the whole field is taken. This computation can be done in time $\mathcal{O}(|R_i| \cdot k)$. For a set W of relational prototypes, we refer to the set containing a k-approximation for each relational prototype $w_i \in W$ a *k-approximation* of W.

These k-approximations in combination with their corresponding coefficients can be interpreted as a convex-combined point in the relational model, defined just over the points of the k-approximation. Therefore, if merged into the next patch, the number of the prototype coefficients remains limited, and the distances of these approximated prototypes to points of the next patch can be calculated using the original equations. This way, only a fraction of the inter-patch distances needs to be considered.

Construction of Extended Patches

Let W_t be a set of optimal relational prototypes gained in a step t. Assume N_t denotes the index set of all points included in the union of a k-approximation of W_t pointing onto elements of the dissimilarity matrix D. The extended patch P_t^* is then characterized by the distance matrix

$$P_t^* = \left(\begin{array}{c|c} d(N_{t-1}) & d(N_{t-1}, P_t) \\ \hline d(N_{t-1}, P_t)^T & P_t \end{array} \right)$$

where

$$d(N_{t-1}) = (d_{uv})_{u,v \in N_{t-1}} \in \mathbb{R}^{n_t \times n_t}$$
$$d(N_{t-1}, P_t) = (d_{uv})_{u \in N_{t-1}, v=(t-1)\cdot p+1,\ldots,t\cdot p} \in \mathbb{R}^{n_t \times p}$$

denote the inter-distances of points from the k-approximation and the distances between points from the k-approximation and current patch points, respectively. The size n_t is bounded by $|W_t| \cdot k$.

Integrating Multiplicities

The original Relational Neural Gas method has to be modified to handle data-points v_j equipped with multiplicities m_j. If these data points originate from a k-approximation of prototypes of the previous patch, the multiplicities are given by

$$m(w_i) = \frac{1}{k} \cdot \sum_{v \in P_t^* \cap V_i(W_t)} m(v),$$

the sum of multiplicities of all data points in the receptive field divided by k.

Incorporating multiplicities into the cost function yields the update rule

$$\bar{\alpha}_{ij} = \frac{m_j \cdot h_\lambda(k_i(v_j))}{\sum_t m_t \cdot h_\lambda(k_i(v_t))}$$

for prototype coefficients. The computation of distances is not changed.

Patch Relational Neural Gas

Assembling the pieces, we obtain Algorithm 4.5.

Algorithm 4.5: Patch Relational Neural Gas

Begin

Cut the first Patch P_1

Apply Relational NG on $P_1 \longrightarrow$ Relational prototypes W_1

Use k-Approximation on $W_1 \longrightarrow$ Index set N_1

Update Multiplicities m_j according to the receptive fields

Repeat for $t = 2, \ldots, n_p$

Cut patch P_t

Construct Extended Patch P_t^* using P_t and index set N_{t-1}

Apply modified RNG with Multiplicities $\longrightarrow$ Relational prototypes W_t

Use k-Approximation on $W_t \longrightarrow$ Index set N_t

Update Multiplicities m_j according to the receptive fields

Return k-approximation of final prototypes N_{n_p}

End.

Complexity

Obviously, the size of extended patches is bounded by the size of the new patch read from the distance matrix and the distances of the at most $k \cdot n$ points representing the n prototypes of the last run by their k approximation. Assume a bounded extended patch size p independent of the number of datapoints, as it would be the case when the patch size is chosen according to memory limitations. The algorithm then works only on $\mathcal{O}(\frac{m}{p} \cdot p^2) = \mathcal{O}(m \cdot p) = \mathcal{O}(m)$ entries of the dissimilarity matrix, compared to $\mathcal{O}(m^2)$ in the original RNG method. Moreover, the algorithm uses at most $\mathcal{O}(p^2) = const$ entries at a specific point in time.

In case of fixed patch size, also the time complexity is linear, because the RNG step is $\mathcal{O}(p^2)$ what results in $\mathcal{O}(p^2 \cdot \frac{m}{p}) = \mathcal{O}(p \cdot m) = \mathcal{O}(m)$, an advantage compared to the $\mathcal{O}(m^2)$ time complexity of the original RNG. Furthermore, the algorithm can be run in a single pass over the data.

These advantages in space and time complexity are obtained by an approximation of the prototypes. As we will see in experiments, this leads only to a small loss in accuracy.
Hence one of the key findings of this section is that the relational methods, despite their original quadratic time complexity, can now be applied successfully to the very large datasets arising in almost all fields of science today.

In the last section, we will focus on experiments and simulations that shall demonstrate the benefits of the above presented methods on non-Euclidean datasets.

4.6 Experimental Results and Applications

In this section, we will demonstrate the benefits of the introduced methods on non-Euclidean datasets.

Class No.	Count	Percentage
HA	72	31.86%
HB	72	31.86%
MY	39	17.26%
GG/GP	30	13.27%
Others	13	5.75%

Table 4.1: Class Statistics of the Protein Dataset

Description of Datasets

In the following, we give a brief description of the datasets that are used throughout the experiments. A small collection of datasets has been carefully chosen, in a way that these datasets cover very different aspects, so the experimental results can be generalized to real-world applications as well.

Cat Cortex The Cat Cortex Data Set originates from anatomic studies of cats' brains. A matrix of connection strengths between 65 cortical areas of cats was compiled from literature (Graepel et al., 1999). There are four classes corresponding to four different regions of the cortex. For our experiments a preprocessed version of the data set from Haasdonk et al. (Haasdonk and Bahlmann, 2004) was used. The matrix is symmetric but the triangle inequality does not hold.

Chicken Pieces Silhouettes The task is to classify 446 silhouettes of chicken pieces into 5 categories (wing, back, drumstick, thigh and back, breast). Data silhouettes are represented as a string of the angles of consecutive tangential line pieces of length 20, including appropriate scaling. The strings are then compared using a (rotation invariant) edit distance, where insertions/deletions cost 60, and the angle difference is taken otherwise.

Copenhagen Chromosomes The Copenhagen chromosomes database is a benchmark from cytogenetics (Lundsteen et al., 1980). A set of 4200 human chromosomes from 22 classes (the autosomal chromosomes) are represented by the gray levels of their images. These images were transferred to strings representing the profile of the chromosome by the thickness of their silhouettes. The strings were then compared using edit distance with substitution costs given by the signed difference of the entries and insertion/deletion costs given by 4.5 (Neuhaus and Bunke, 2006). The edit distance is a typical distance measure for two strings of different length, as described in (Juan and Vidal, 2000).

Protein Data The evolutionary distance of 226 globin proteins is determined by alignment as described in (Mevissen and Vingron, 1996). These samples originate from different protein families: hemoglobin-α, hemoglobin-β, myoglobin, etc. Here, we distinguish five classes as proposed in (Haasdonk and Bahlmann, 2004): HA, HB, MY, GG/GP, and others. Unlike the other datasets considered here, the protein dataset has a highly unbalanced class structure. Table 4.1 shows the class distribution of the dataset. Note that the class *Others* combines small classes from the original dataset and represents only a small fraction of the whole dataset.

Macroarray Data An important topic in bioinformatics is the gene expression analysis. For this dataset, the temporal development of barley seeds was analyzed, what is an main issue for the derivation of key metabolic processes during different stages of growth. Extensive gene expression measurements at 14 time points from

day zero after flowering to day 26 in steps of two days were carried out using cDNA macroarray technology. High signal to noise ratios and high reproducibility between two independently taken experimental series led to a selection of 4824 genes out of 11786 genes available on the 12k macroarrays. Thus, a data matrix of 4824 (genes) by 14 (time points) is considered. As common in gene expression analysis, $\log_2$-transformed final expression values are considered. Coexpression analysis of the available 4824 gene expression time series is required to identify groups of commonly regulated genes that may have temporal impact on each other. Such a clustering helps to extract candidate genes responsible for triggering later events like, for example, the influence of cell wall degradation factors for lateral tissue nutrition or subsequent starch accumulation processes.

Wisconsin Diagnostic Breast Cancer The Wisconsin Diagnostic Breast Cancer database (WDBC) is a standard benchmark set from clinical proteomics (Wolberg et al., 1995). This dataset is only Euclidean chosen to serve as benchmark. It consists of 569 Euclidean data points described by 30 real-valued input features: digitized images of a fine needle aspirate of breast mass are described by characteristics such as form and texture of the cell nuclei present in the image. The data are labeled into two classes, benign and malignant.

Analysis of non-Euclidean Datasets

In the following the non-Euclidean datasets are analyzed regarding their Euclideanity. For that reason, characteristical properties like eigenvalues of the corresponding Gram matrices are shown that have been discussed before in the last section.

The signatures of the dissimilarity matrices (inertia of corresponding Gram matrices) for the datasets is shown in Table 4.2. Except for the *Protein* dataset, all dissimilarity matrices feature a large value e_- meaning there are many negative eigenvalues in the spectra of the corresponding Gram matrices what indicates that they might be highly non-Euclidean.

Signature of Non-Euclidean Datasets	
Cat Cortex	(41,23,1)
Chicken Pieces	(240,205,1)
Copenhagen Chromosomes	(1951,2206,43)
Macroarray Data	(2450,2374,0)
Protein Data	(218,4,4)

Table 4.2: Inertia (e_+, e_-, e_0) of corresponding Gram matrices

A more detailed view on the structure of the datasets is given by the magnitude of the eigenvalues. Here, the non-Euclidean datasets used in the experiments are analyzed by means of their sorted eigenspectra of the respective Gram matrices. The eigenvalues were normalized by the average dissimilarity before visualization to make them more comparable.

As it can be seen, all datasets feature a few dominant positive eigenvalues and many eigenvalues around zero, meaning that most structural information is contained only in a few dimensions. The negative eigenvalues are of a lower magnitude for most datasets. It should be noted that the non-Euclidean part cannot be larger than the Euclidean part, because the dissimilarities are constrained to be positive.

Finally, we draw the reader's attention to the Cat Cortex dataset (see Figure 4.3) which has a strong negative part in its eigenspectrum, so its non-Euclideanity is high.

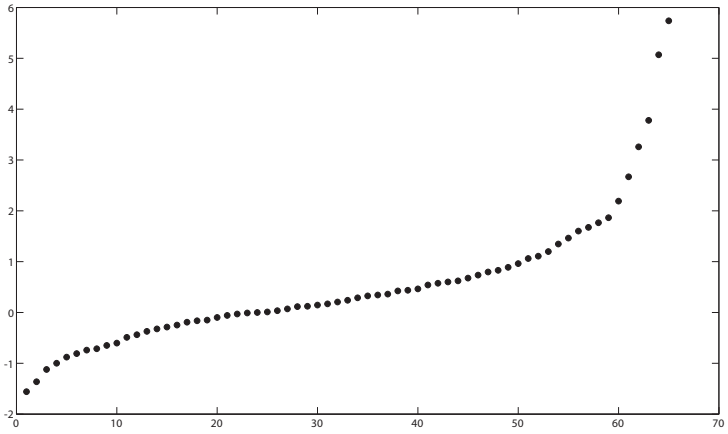


Figure 4.3: Cat Cortex Dataset: Eigenspectrum of Gram Matrix

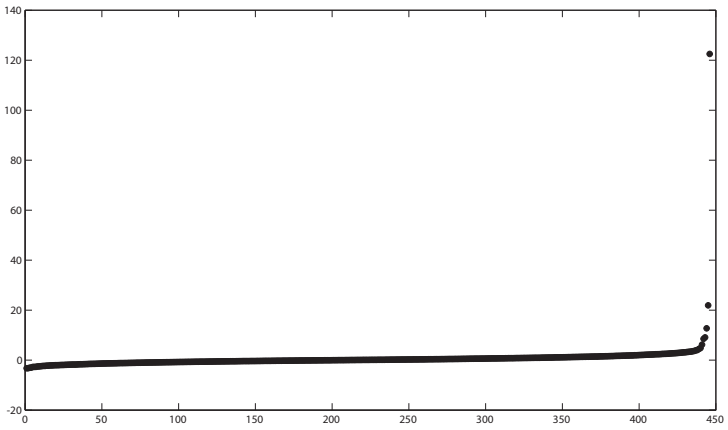


Figure 4.4: Chicken Pieces Dataset: Eigenspectrum of Gram Matrix

Just to give an idea, Table 4.3 shows how large the spreading values have to be chosen to transform the dissimilarity into Euclidean or squared Euclidean distances, respectively.

Experimental Results on Dissimilarity Datasets

We demonstrate the performance of the relational algorithms in different scenarios given by the above described datasets covering a variety of characteristic situations.

Note that, for all median versions, prototypes situated at identical points of the data space do not separate in subsequent runs. Therefore constellations with exactly identical prototypes should be avoided. For the Euclidean and relational versions this problem is negligible, presumed prototypes are initialized at different positions. However, for median versions it is likely that prototypes move to an identical locations due to the limited number of different positions in data space, in particular for small data sets. To cope with this fact in median versions, we

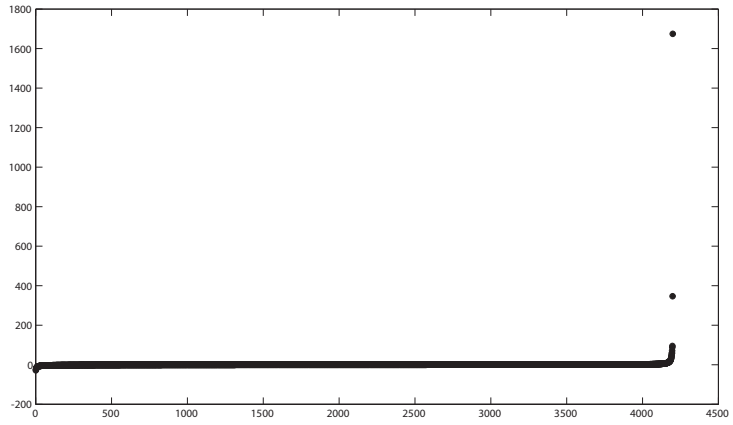


Figure 4.5: Copenhagen Chromosome Dataset: Eigenspectrum of Gram Matrix

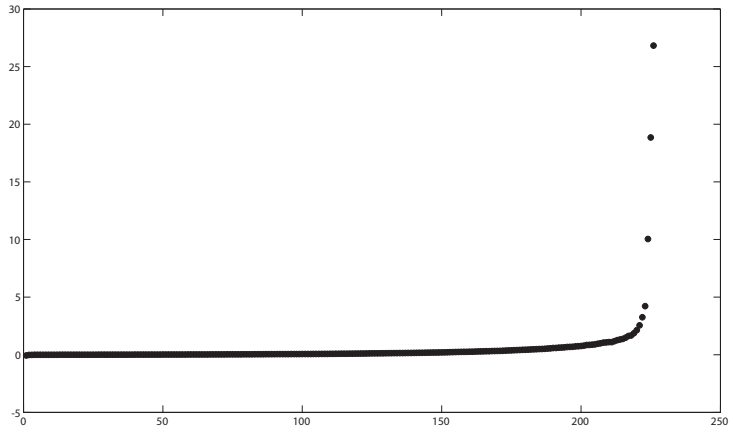


Figure 4.6: Protein Dataset: Eigenspectrum of Gram Matrix

add a small amount of noise to the distances in each epoch in order to separate identical prototypes. In all runs, the methods have been applied directly without any correction of the given dissimilarity matrices.

For all experiments the initial neighborhood range λ_0 is chosen as $n/2$ with n the number of neurons used. The neighborhood range $\lambda(t)$ is decreased exponentially with the number of adaptation steps t according to $\lambda(t) = \lambda_0 \cdot (0.01/\lambda_0)^{t/t_{\max}}$ (cf. (Martinetz et al., 1993)). The value $t_{\max}$ is chosen as the number of epochs.

Wisconsin Diagnostic Breast Cancer For training we used 40 neurons and 150 epochs per run. The dataset was z-transformed beforehand. The results were gained from repeated 2-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 4.4. Moreover, the data set is contained in the Euclidean space therefore we are able to compare the relational versions introduced in this article to the standard Euclidean methods. These results are shown in Table 4.4. The effect of a variation of the mixing parameter is demonstrated in Fig. 4.7. The results are competitive to supervised learning with the state-of-the-art-method GRLVQ as obtained in (Schleif et al., 2007).

As one can see, the results of Euclidean and relational clustering are identical, as

Minimum Spread Transform

Dataset	**sq. Euclidean**	**Euclidean**
Cat Cortex	4.9321 (1.4105)	12.7616 (3.6495)
Chicken Pieces	36.5429 (1.7107)	98.5631 (4.6142)
Copenhagen Chromosomes	851.7972 (6.9389)	1865.9 (15.20)
Macroarray Data	0.0028 (0.0031)	23.8977 (24.1545)
Protein Data	0 (0)	12.6932 (1.2366)

Table 4.3: Minimum spreading value for the datasets to become squared Euclidean and Euclidean. In parentheses, the spreading value normalized by the average dissimilarity is given.

	k-Means	Supervised k-Means	Median k-Means	Relational k-Means	Supervised Relational k-Means
Accuracy					
Mean	93.6	93.0	93.0	93.4	93.5
StdDev	0.8	1.1	1.0	1.2	1.1
	Batch NG	Supervised Batch NG	Median Batch NG	Relational Batch NG	Supervised Relational Batch NG
Accuracy					
Mean	94.1	94.7	93.1	94.0	94.4
StdDev	1.0	0.8	1.0	0.9	1.0

Table 4.4: Classification accuracy on the WDBC database for posterior labeling. The mean accuracy over 100 repeats of 2-fold cross-validation is reported.

expected by the theoretical background of relational clustering. Relational clustering and supervision allow to improve the more restricted and unsupervised median versions by more than 1% classification accuracy.

Cat Cortex The algorithms were tested in 10-fold cross-validation using 12 neurons (three per class) and 150 epochs per run. The results presented show the mean accuracy over 250 repeated 10-fold cross-validations per method. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 4.5. Results for different mixing parameters are shown in Figure 4.8.

Haasdonk and Bahlmann (2004) gained an accumulated error over all classes of at least 10% in leave-one-out experiments with SVMs. Graepel et al. (1999) obtained virtually the same results with the Optimal Hyperplane (OHC) algorithm. In our experiments, the improvement of restricted median clustering by relational

	Median k-Means	Median Batch NG	Relational k-Means	Relational Batch NG	Supervised Median Batch NG	Supervised Relational k-Means	Supervised Relational Batch NG
Accuracy							
Mean	72.8	71.6	89.0	88.7	77.9	89.2	91.3
StdDev	3.9	4.0	3.3	3.0	3.5	3.0	2.8

Table 4.5: Classification accuracy on the Cat Cortex Data Set for posterior labeling. The mean accuracy over 250 repeats of 10-fold cross-validation is reported.

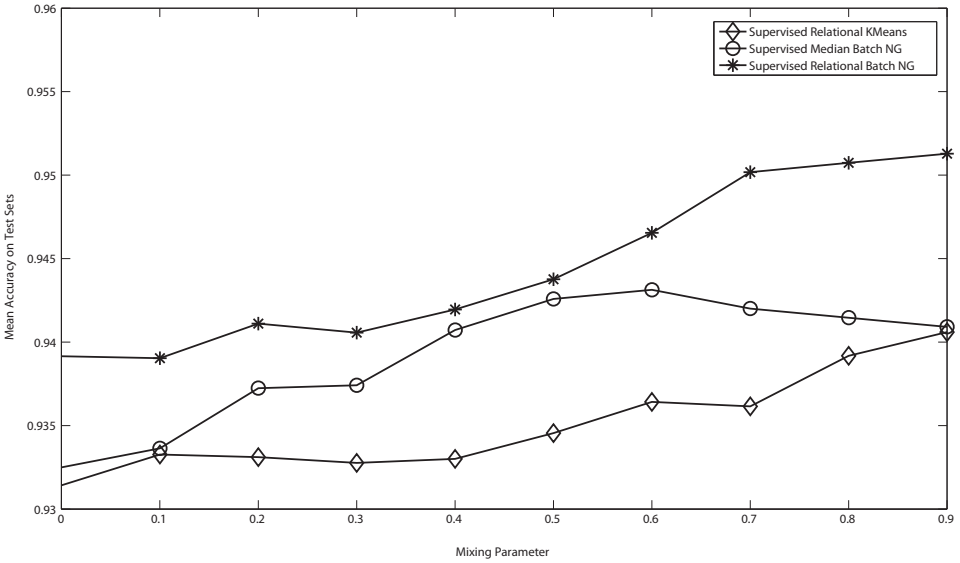


Figure 4.7: Results of the supervised methods for the WDBC data set with different mixing parameters applied.

extensions by more than 10% classification accuracy can clearly be observed. Note that relational clustering works quite well in this case although an interpretation by means of prototypes is not directly possible.

Proteins For training we used 45 neurons and 150 epochs per run. The results were gained from repeated 10-fold cross-validations averaged over 100 runs. The mixing parameter of the supervised methods was set to 0.5 for the simulations reported in Table 4.6.

Unlike the results reported in (Haasdonk and Bahlmann, 2004) for SVM which uses one-versus-rest encoding, the classification in our setting is given by only one clustering model. Depending on the choice of the kernel, (Haasdonk and Bahlmann, 2004) reports errors which approximately add up to 4% for the leave-one-out error. This result, however, is not comparable to our results due to the different error measure. A 1-nearest neighbor classifier yields an accuracy 91.6 for our setting (k-nearest neighbor for larger k is worse; (Haasdonk and Bahlmann, 2004)) which is comparable to our results.

The Self-Organizing Map variants for dissimilarity datasets were tested against the Neural Gas variants. For training we use 29 neurons for Relational Batch NG and Relational Hyperbolic SOM, and a 5x5 grid for standard Relational SOM, respectively. The number of neurons is derived from the hyperbolic grid of depth three (cf. Figure 4.10). The neighborhood range is annealed starting from $N/2$ to

	Median k-Means	Median Batch NG	Relational k-Means	Relational Batch NG	Supervised Median Batch NG	Supervised Relational k-Means	Supervised Relational Batch NG
Accuracy							
Mean	76.1	76.3	88.0	89.9	89.4	88.2	90.0
StdDev	1.3	1.8	1.8	1.3	1.4	1.7	1.0

Table 4.6: Classification accuracy on the Protein Data Set for posterior labeling

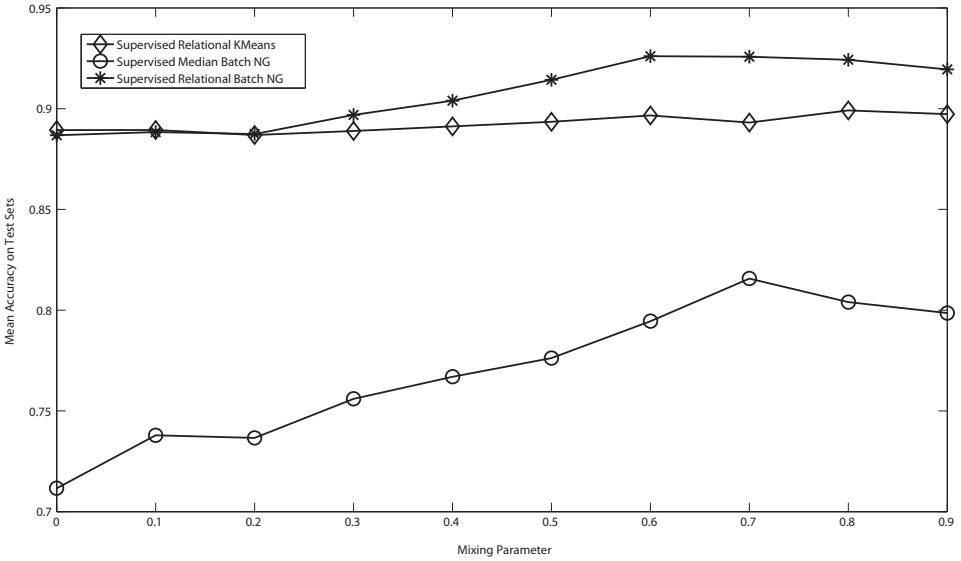


Figure 4.8: Results of the supervised methods for the Cat Cortex Data Set with different mixing parameters applied.

0, N being the number of neurons, in all experiments. The results reported in Table 4.7 are gained from repeated 10-fold stratified cross-validation averaged over 100 repetitions and 150 epochs per run. Supervision is included in the cost function with mixing parameter 0.5.

Note that the different clusters can easily be identified. The Relational SOM provides an improved technique to explore dissimilarity data, revealing the structures of interest.

The projections of a Relational SOM with hyperbolic grid structure and of Relational BNG with non-metric multidimensional scaling using Kruskal's normalized stress1 criterion are shown in figure 4.10 and 4.9. The neurons are depicted according to majority vote. Obviously, the neurons arrange according to the associated class and a very clear two-dimensional representation of the data set is obtained.

Copenhagen Chromosomes The Neural Gas variants for dissimilarity datasets were tested in 2-fold stratified cross-validation using 100 neurons and 100 epochs per run (cf. (Cottrell et al., 2006)). The results presented are the mean accuracy over 10 times 2-fold cross-validation per method. The mixing parameter of the supervised methods was set to 0.9.

As it can be seen from Table 4.8, supervised relational neural gas achieves an accuracy of 0.914 for $\alpha = 0.9$. This is an improvement by 8% compared to the median variants.

	Median Batch NG	Relational Batch NG	Standard Relational SOM	Relational Hyperbolic SOM
Accuracy Protein Data set				
Mean	89.5	92.4	91.5	91.5
StdDev	1.0	0.9	0.8	1.2

Table 4.7: Classification accuracy on the Protein Data Set with posterior labeling

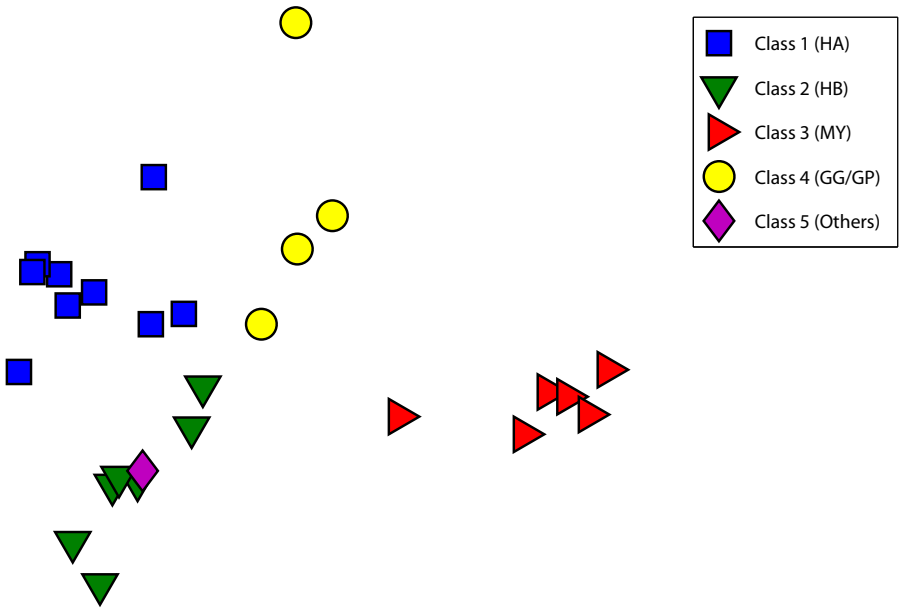


Figure 4.9: Mapping of the non-euclidean Protein dataset by Relational BNG with non-metric multidimensional scaling.

	Median k-Means	Median Batch NG	Relational k-Means	Relational Batch NG	Supervised Median Batch NG	Supervised Relational k-Means	Supervised Relational Batch NG
Accuracy							
Mean	82.3	82.8	90.6	91.3	89.4	90.1	91.4
StdDev	2.2	1.7	0.6	0.2	0.6	0.6	0.6

Table 4.8: Classification accuracy of the Neural Gas variants on the Copenhagen Chromosome Database with posterior labeling

The Self-Organizing Map variants for dissimilarity datasets were tested in a repeated 2-fold stratified cross-validation using 85 neurons for Relational BNG and Relational HSOM (corresponding two three rings), and a 9x9 grid for the standard Relational SOM. The results presented in Table 4.8 are the mean accuracy over 10 repetitions per method and 100 epochs per run. Supervision is incorporated using the mixing parameter 0.9. Again, an improvement of 2% can be observed compared to the median variants.

Macroarray Data In Figure 4.11 a topographic mapping is shown which was obtained by training a HSOM with 85 neurons for 150 epochs. Here, a transformation of Pearson correlation of the expression patterns was used as a dissimilarity measure which better accounts for the overall principled shape as described in (Strickert et al., 2006). Obviously, the map organizes the data according to the evolution of up- and down-regulation of the genes over time. Due to the hyperbolic structure, the map clearly separates opposite shapes and preserves the data topology. It should be noted here that the map nicely captures the relevant classes of up-regulated (upper and left half of the diagram) and down-regulated (lower and right half) as well as

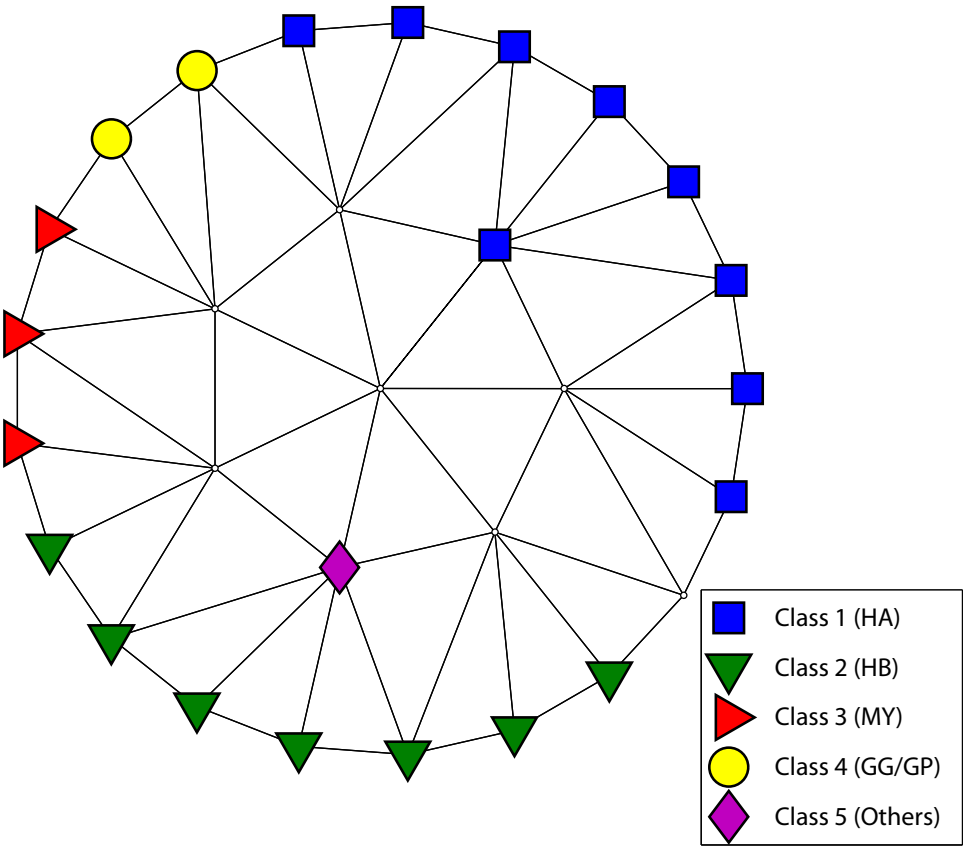


Figure 4.10: Mapping of the non-euclidean Protein dataset by a Relational SOM with hyperbolic grid structure.

shapes in between.

Experimental Results on Magnification Control

We test local learning for a Euclidean benchmark, and four non-Euclidean settings as described in (Neuhaus and Bunke, 2006; Haasdonk and Bahlmann, 2004). In the latter cases, local learning can be applied and it can be expected that 'dense' or 'sparse' regions, respectively, of the data are emphasized depending on m due to the optimized costs (4.17). However, the exact theoretical law of the prototype density and its information theoretic optimum is not known. Note that, if $D^{\star 2}$ stems from a metric, the concept of dimensionality can be defined for the underlying data manifold and the intrinsic data dimensionality can been estimated using a

	Median Batch NG	Relational Batch NG	Standard Relational SOM	Relational Hyperbolic SOM
Accuracy Copenhagen Chromosome Database				
Mean	88.8	90.7	89.9	89.4
StdDev	1.2	0.5	0.6	0.7

Table 4.9: Classification accuracy of the SOM variants on the Copenhagen Chromosome Database with posterior labeling

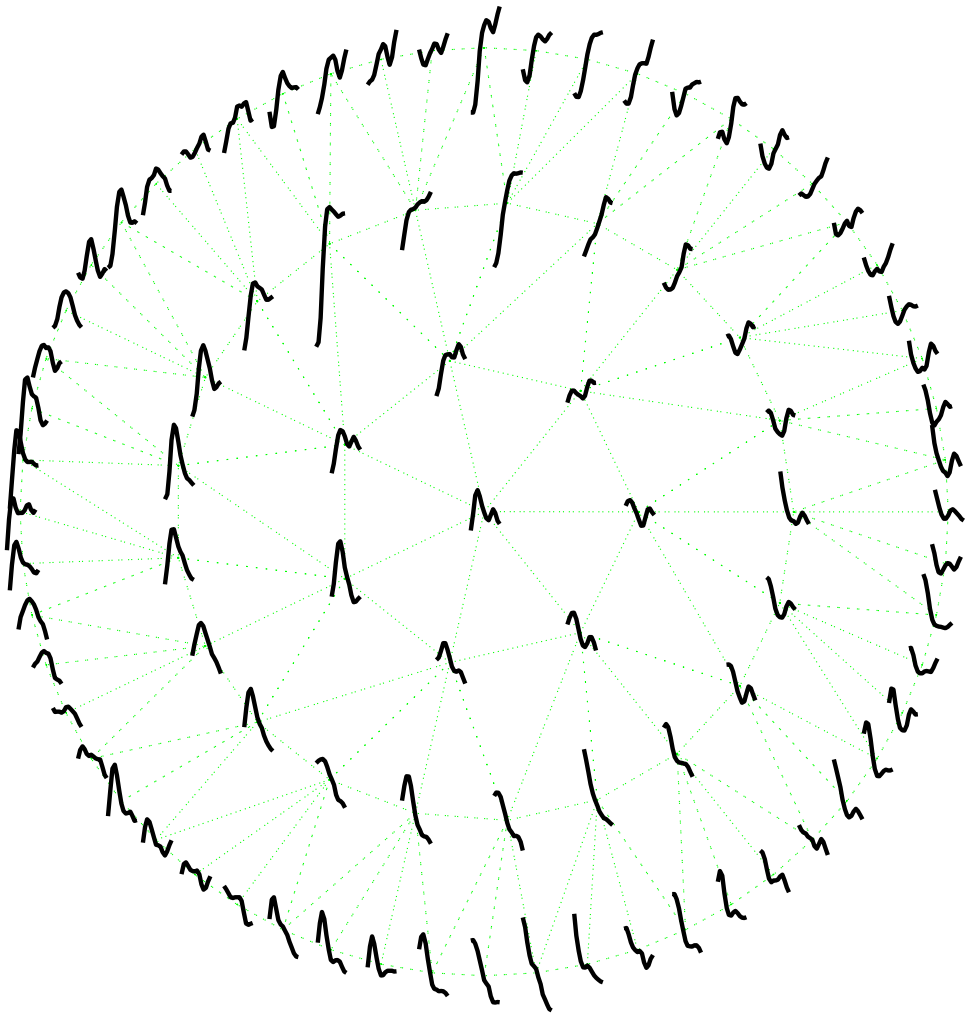


Figure 4.11: Visualization of the macroarray dataset by Relational SOM with hyperbolic grid structure.

Grassberger-Procaccia analysis (Grassberger and Procaccia, 1983). Similarly, density estimation is possible for separable metric spaces and uniformly continuous densities by means of histogram estimators (Geffroy, 1974). For simplicity, we compute $P(v)$ using a simple Parzen window with bandwidth chosen as a third of the average point distance, which gives a rough approximation to the underlying density for Euclidean and metric settings, respectively.

For all experiments the initial neighborhood range λ_0 was chosen as $n/2$, n being the number of neurons. The neighborhood range $\lambda(t)$ at epoch t was decreased according to $\lambda(t) = \lambda_0 \cdot (0.01/\lambda_0)^{t/t_{\max}}$ (cf. (Martinetz et al., 1993)), $t_{\max}$ being the number of epochs. If not indicated otherwise, the number of epochs was 100.

Control experiment At first, the experiment from (Hammer et al., 2007b) for Euclidean data was repeated for RNG as control. Data were sampled from the distribution

$$(v_1, \ldots, v_d, \prod_{j=1}^{d} \sin(\pi \cdot v_j))$$

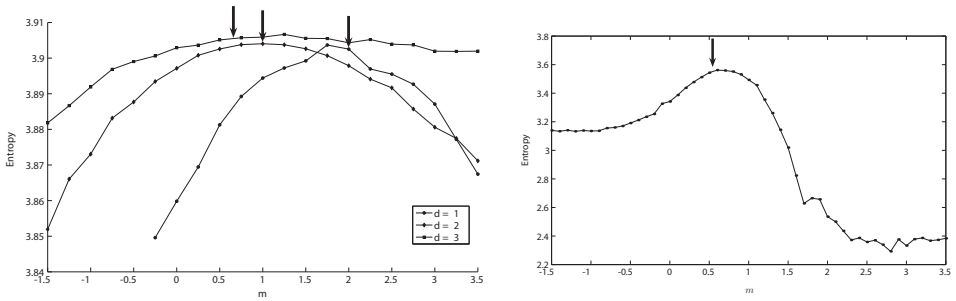


Figure 4.12: *Left:* Euclidean benchmark data – Entropy of map formation for different values c of magnification control and training sets of intrinsic dimensionality $\hat{d} \in \{1, 2, 3\}$ — *Right:* Chicken Pieces Dataset – Entropy for different magnification control parameter values c

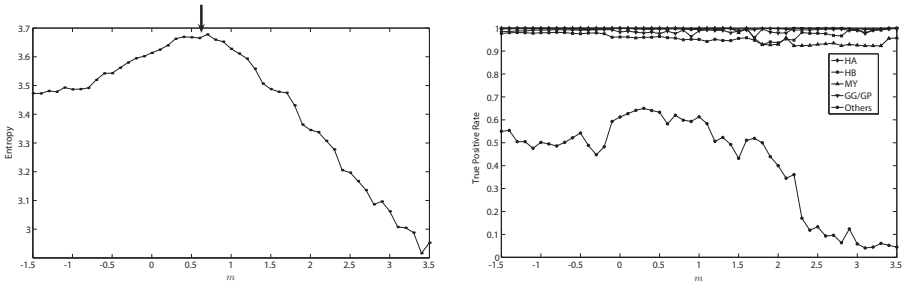


Figure 4.13: Protein Dataset – Left: Entropy for different magnification control parameter values c, Right: true positives rates

for $d \in \{1, 2, 3\}$ and uniform $v_i \in [0, 1]$. The number of data points was chosen as 2500 for $d = 1$, 5000 for $d = 2$, and 10000 for $d = 3$. We trained RNG for control values $m \in [-1.5, 3.5]$ and step size 0.25. A NG network with 50 neurons has been used. The reported results have been averaged over 20 runs.

The information theoretic quality of the map can be judged by computing the balance of patterns in the receptive fields. For equal values, an optimum information transfer is achieved. The values of the map entropy are reported in Fig. 4.12. The entropy should be maximum for optimum information transfer, i.e. for $c = 2$ ($\hat{d} = 1$), $c = 1$ ($\hat{d} = 2$), and $c = 2/3$ ($\hat{d} = 3$). As indicated by the arrows, the experimental optima of the curves are closely situated to the expected theoretical values.

Protein Dataset For the experiment RNG with magnification control parameter $c \in [-1.5, 3.5]$ (step size 0.1) and 50 neurons was trained. The results presented in Fig. 4.13 are the average over 100 runs. The theoretical optimum $c^* \approx 0.63$ for the Euclidean case as indicated by the arrow in Fig. 4.13 was derived from the estimated intrinsic dimension $\hat{d} \approx 3.18$. Note that magnification control by localized learning is obviously possible for this non-Euclidean setting. Interestingly, the information theoretic optimum of the curve is closely situated to the Euclidean one.

To demonstrate the magnification effect on exterior (small) classes, the true positives rate for each class is depicted in Fig. 4.13. Apparently, the classification rate is getting better when focusing on rare events, i.e. for small control parameter c.

Chicken Pieces Silhouettes Dataset We trained a RNG network with magnification control using 50 neurons and control parameter $c \in [-1.5, 3.5]$ (step size

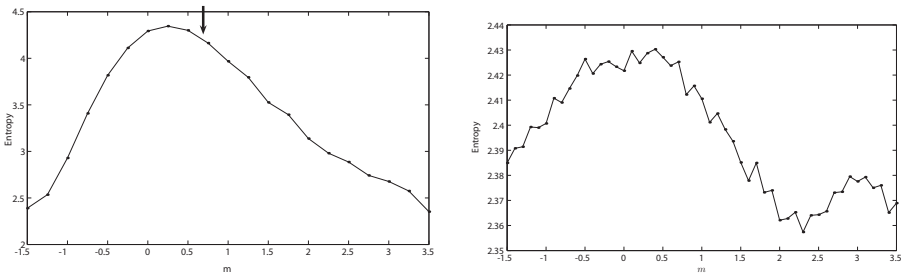


Figure 4.14: Chromosome Dataset (left) and CatCortex Dataset (right) – Entropy for different magnification control parameter values c

0.1). The average over 100 runs for each different value c was taken.

The arrow in Fig. 4.13 indicates the theoretical optimum $c^* \approx 0.54$ for the Euclidean case that was derived from the estimated intrinsic dimension $\hat{d} \approx 3.72$.

Copenhagen Chromosome Database Relational Neural Gas with magnification control has been trained using 80 neurons for control parameter $c \in [-1.5, 3.5]$ (step size 0.25).

The results shown in Fig. 4.14 present the average over 10 runs for each different value c. The figure shows very smooth control of the map entropy by localized learning. The observed optimum for the considered metric differs from the corresponding optimum $c^* \approx 0.68$ ($\hat{d} \approx 2.93$) in the Euclidean case.

Cat Cortex Dataset For the experiment, Relational Neural Gas with magnification control has been trained using 12 neurons for control parameter $c \in [-1.5, 3.5]$ (step size 0.1). The results shown in Fig. 4.14 (right) present the average over 100 runs. A Grassberger-Procaccia analysis cannot be applied due to the data statistics such that the theoretical optimum in the Euclidean case is not known. However, magnification control is clearly possible also in this case.

Experimental Results on Very Large Datasets

To show the overall performance of the proposed patch methods, we have chosen some representative very large dissimilarity datasets. Due to limited computing power and hardware available, the chosen datasets do not represent real-life huge datasets, they should be understood as a proof-of-concept that nevertheless can instantly be transferred to the real problems.

We evaluate the clustering results by means of the classification error for supervised settings, whereby class labels are obtained by posterior labeling of prototypes. Note, however, that the goal of the algorithms is meaningful clustering of data based on a chosen similarity measure and cost function. Hence, the classification error gives only a hint about the quality of the clustering, depending on whether the class labels are compatible to the data clusters and chosen metric or not. We accompany this supervised evaluation be the standard quantization error of the clustering.

Synthetic Dataset To analyze the relation between the number of patches and the quantization error on one hand, and the effect of k-approximation of relational prototypes on the other hand, an artificial dataset from (Cottrell et al., 2006) was taken. It consists of 1250 datapoints in the Euclidean plane gained from three Gaussian clusters.

Effect of k-Approximation For an empirical study of the effect of k-approximation on the quantization error, we trained 50 neurons with the original Relational NG for 100 epochs, i.e. on average every neuron represents 25 datapoints. On the outcoming relational neurons, k-Approximation for $k = 1, \ldots, 20$ were applied. Figure 4.15 shows a comparison of the quantization errors yielded with the different approximations to the quantization error gained by the original relational neurons. For each step the average over 10 runs is reported.

As expected, the quantization error decreases with higher numbers k of datapoints used to approximate each relational neuron. Concerning the patch approach, applying a k-approximation to the relational neurons of each patch clearly results in a loss of accuracy depending on the choice of parameter k. But as can be seen later on, even with k-approximation the quality of the results is still convincing.

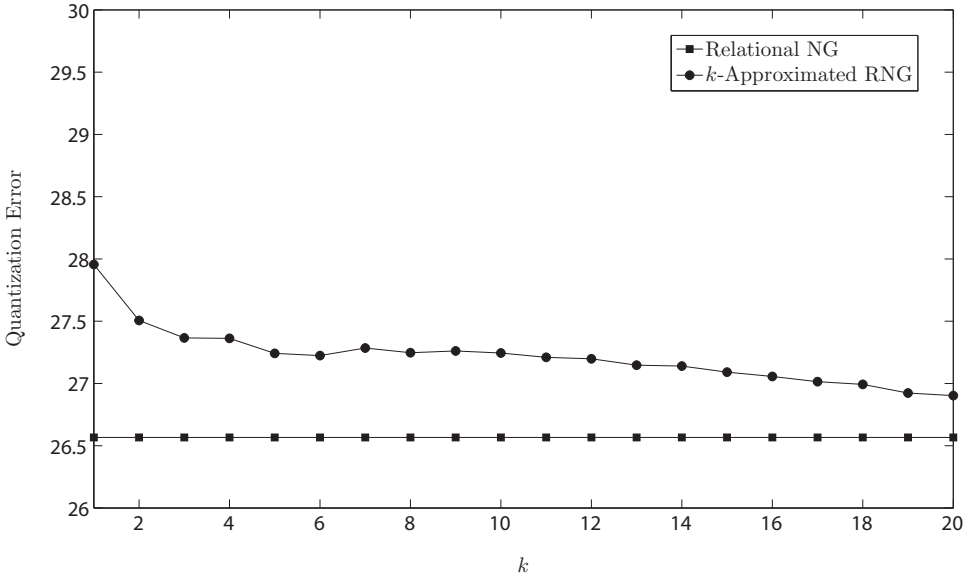


Figure 4.15: Quantization error (i.e. $E(W)$ for $\lambda \to \infty$) of original relational neurons compared to different k-approximations on a synthetic dataset

Effect of Patch Sizes Analyzing the relation between the number of patches chosen and the quantization error, we trained median and relational NG with 20 neurons for 50 epochs. The results presented in figure 4.16 show the quantization error averaged over 10 runs for each number of patches. As expected, the quantization error increases with the number of patches used. But compared to the Median Patch NG approach the presented Patch Relational NG performs very well with only a small loss even for a larger number of patches used.

Chicken Pieces Silhouettes For training we used 30 neurons. For Patch Median NG the dataset was divided into 4 patches, i.e. a patch size of around 111 datapoints. The results reported in Table 4.10 are gained from a repeated 10-fold stratified crossvalidation averaged over 100 repetitions and 100 epochs per run. The k-approximation for Patch Relational NG was done with $k = 3$.

Protein Classification The Protein Dataset as described on page 80 is processed by the patch variants of Neural Gas. For training we used 20 neurons. The dataset was divided into 4 patches, i.e. a patch size of around 57 datapoints. The results

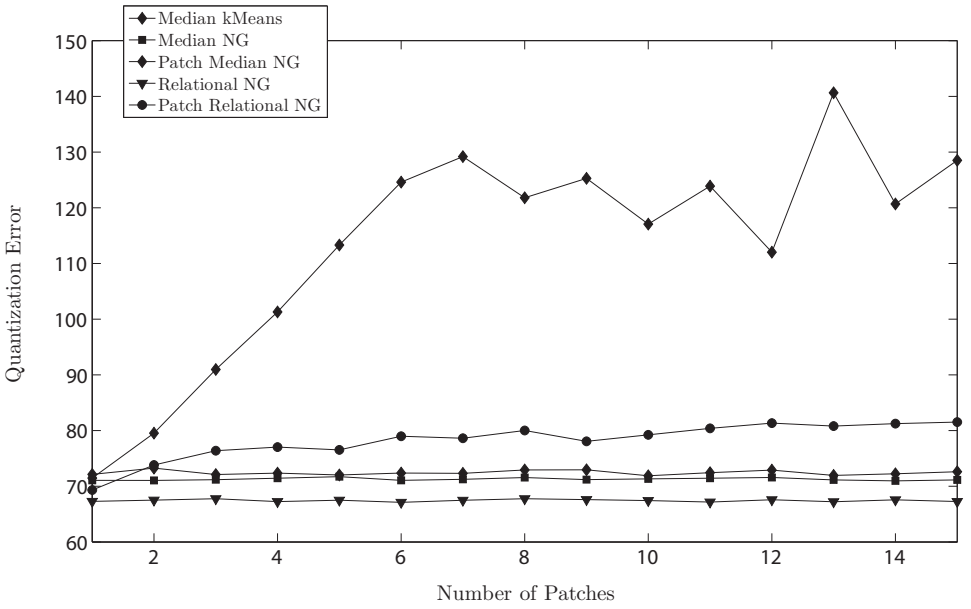


Figure 4.16: Quantization error for different patch sizes on a synthetic dataset

reported in Table 4.11 are gained from a repeated 10-fold stratified crossvalidation averaged over 100 repetitions and 100 epochs per run.

Despite the small size of this dataset – acting more as a proof-of-concept example – the results clearly show a good performance. Nevertheless, the price of reduced accuracy is obvious, but faster computation and less space requirements are gained in return. The k-approximation for Patch Relational NG was done with $k = 3$.

Wisconsin Diagnostic Breast Cancer Here, dissimilarities were derived by applying the Cosine Measure

$$d_{cos}(v_i, v_j) = 1 - \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|}.$$

We trained 40 neurons for 100 epochs. As result the accuracy on the test set for a repeated 10-fold stratified crossvalidation averaged over 100 runs is reported. The number of patches chosen for Patch Median NG and Patch Relational NG was 5, i.e. around 114 datapoints per patch. The k-approximation for Patch Relational NG was done with $k = 2$.

Also on this dataset, Patch Relational NG acts marginally worse than the original Relational NG. Though, the reduction in accuracy is clearly observable.

Accuracy on Chicken Pieces Dataset

	Relational NG	Patch Relational NG	Median Batch NG	Patch Median NG	Median k-Means
Mean	84.7	85.4	66.4	68.8	72.9
StdDev	1.0	1.1	1.9	2.3	1.7

Table 4.10: Classification accuracy on Chicken Pieces Dataset gained from repeated 10-fold stratified crossvalidation over 100 repetitions, four patches were used.

Copenhagen Chromosome Database The methods have been trained using 60 neurons for 100 epochs. As result the accuracy on the test set for a repeated 2-fold stratified crossvalidation averaged over 10 runs is reported. The number of patches chosen for Patch Median NG and Patch Relational NG was 10, i.e. around 420 datapoints per patch. The k-approximation for Patch Relational NG was done with $k = 3$.

Also on this dataset, Patch Relational NG acts well. Though, the reduction in accuracy is clearly observable.

Experiments on Very Large Text Datasets

Here, the patch relational methods were applied to very large text datasets (Hasenfuss et al., 2008a). For all text datasets, the extracted texts were pre-processed by removing stop words and applying word stemming (Porter, 1980). The text documents were then compared by the popular Normalized Compression Distance (NCD) (Cilibrasi and Vitáni, 2005), a measure based on approximations of the Kolmogorov Complexity from algorithmic information theory (Li and Vitányi, 1997). The NCD is defined as

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

where x and y are the document strings, C(x) denotes the compressed size of x and C(xy) the compressed size of the concatenation of x and y using a real compressor. For our experiments the bzip2 compression method was used.

Reuters Dataset The popular Reuters-21578 dataset is a collection of 21578 documents that appeared on the Reuters newswire in 1987. The data was split according to the common 'Modified Apte' split and the most important class of each datapoint's multilabel was taken as single class label. For the experiment the 7 most prominent classes were taken (cf. Ontrup and Ritter, 2001). We trained a Supervised Patch Relational SOM with a hyperbolic grid of 85 neurons on 10 patches for 100 epochs using a 5-approximation and a control parameter $\beta = 0.5$. Figure 4.17 shows the visualization as a simple projection of the hyperbolic grid into the Euclidean plane.

20 Newsgroup Dataset The 20 Newsgroups dataset from the UCI repository consists of 20000 newsgroup articles collected from twenty UseNet newsgroups during 1993, 1000 articles of each newsgroup. The newsgroups were grouped by their overall topics into 7 classes. We processed the data by a Supervised Patch Relational SOM with a hyperbolic grid structure of 29 neurons on 60 patches. Training

Accuracy on Protein Dataset

	Relational NG	Patch Relational NG	Median Batch NG	Patch Median NG	Median k-Means
Mean	92.62	92.61	79.9	77.7	80.6
StdDev	0.92	0.88	1.5	2.4	1.3

Table 4.11: Classification accuracy on Protein Dataset gained from repeated 10-fold stratified crossvalidation over 100 repetitions, four patches were used.

Accuracy on Wisconsin Breast Cancer Dataset

	Relational NG	Patch Relational NG	Median Batch NG	Patch Median NG	Median k-Means
Mean	95.0	94.8	94.7	94.4	94.6
StdDev	0.6	0.7	0.7	0.7	0.7

Table 4.12: Classification accuracy on Wisconsin Breast Cancer Dataset with Cosine Measure gained from repeated 10-fold stratified crossvalidation over 100 repetitions, five patches and a 2-approximation were used.

was done for 100 epochs with a 1-approximation and a supervision control parameter $\beta = 0.2$. Figure 4.18 shows the outcome of the method projected into Euclidean plane.

Large Newsgroup Dataset As an example for a very large dataset, we gathered 183,546 newsgroup articles from 13 different newsgroups in analogy to the 20 Newsgroup dataset.

The full dissimilarity matrix of normalized compression distances for this dataset would occupy approx. 251 GB (!), so it were no option to process it with standard batch methods. Instead, we precalculated NCDs for 183 patches of around 1000 documents each, these dissimilarity matrices were stored to files on hard disk. We then applied the novel Patch Relational SOM with 3-approximation and a hyperbolic grid of 85 neurons. That way, only around 274 megabytes of dissimilarities have to be considered by the algorithm and computation time was around 18h instead of an extrapolated half a year! Also the computation required only a constant space of around 12 megabytes (plus some overhead) and could be performed on a common workstation.

Most time consuming part of the calculation was the construction of the extended patch, here we had to determine the normalized compression distances between neurons and datapoints on the fly. Due to the size of the problem it is not possible to calculate and store those distances in advance.

The outcome is a mapping into 2-dimensional hyperbolic space, that can be projected to the Euclidean plane for visualization and data inspection (see fig. 4.19).

Conclusions

Throughout this section it has been demonstrated that the introduced prototype-based methods perform very well on very different non-Euclidean datasets. The experiments cover topographic mapping as well as clustering and classification tasks.

Accuracy on Copenhagen Chromosome Image Dataset

	Relational NG	Patch Relational NG	Median Batch NG	Patch Median NG	Median k-Means
Mean	89.6	87.0	80.0	67.9	77.1
StdDev	0.6	0.8	1.4	3.1	2.2

Table 4.13: Classification accuracy on Copenhagen Chromosome Image Dataset gained from repeated 2-fold stratified crossvalidation over 10 repetitions, 10 patches and a 3-approximation were used.

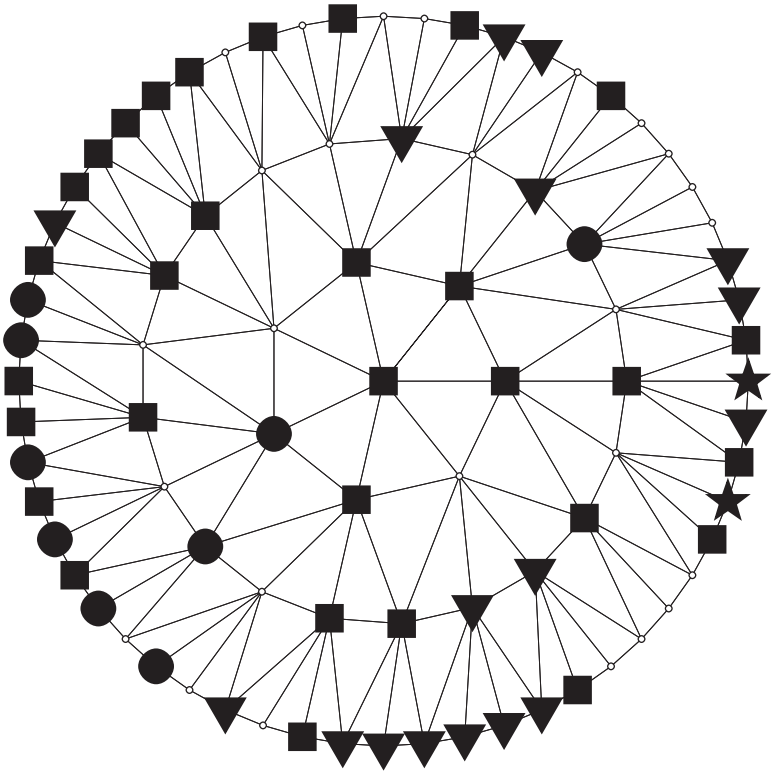


Figure 4.17: Visualization of the 7 most important classes of the Reuters Dataset

The results confirm that the prototype-based methods are very well suited for a broad range of problems. Especially for the topographic mapping of dissimilarity datasets, where concurrent approaches are rare, they pose a promising novel approach.

Also the extensions like supervision, magnification control, and patch processing proved their strength. For instance, the large *Newsgroup* dataset with 180,000 pairwise dissimilarities was processed what is simply impossible if only the original batch algorithms were available. It was also shown that the supervised variants can help to improve the topographic mappings significantly, and that important regions can be magnified, what is very useful in data visualization.

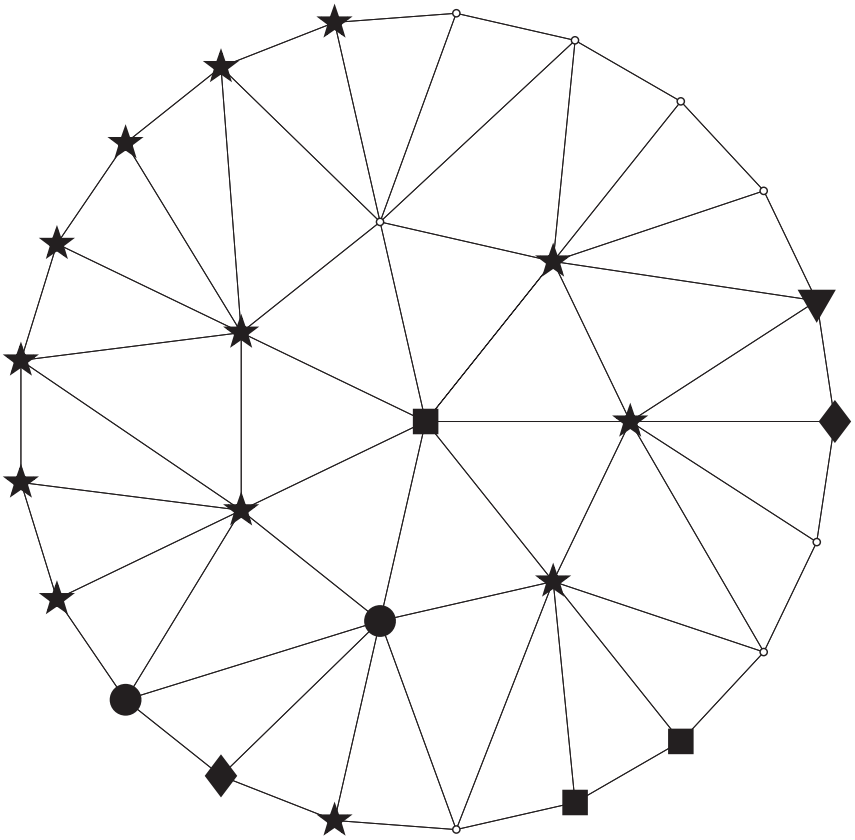


Figure 4.18: Visualization of the UCI 20 Newsgroups Dataset

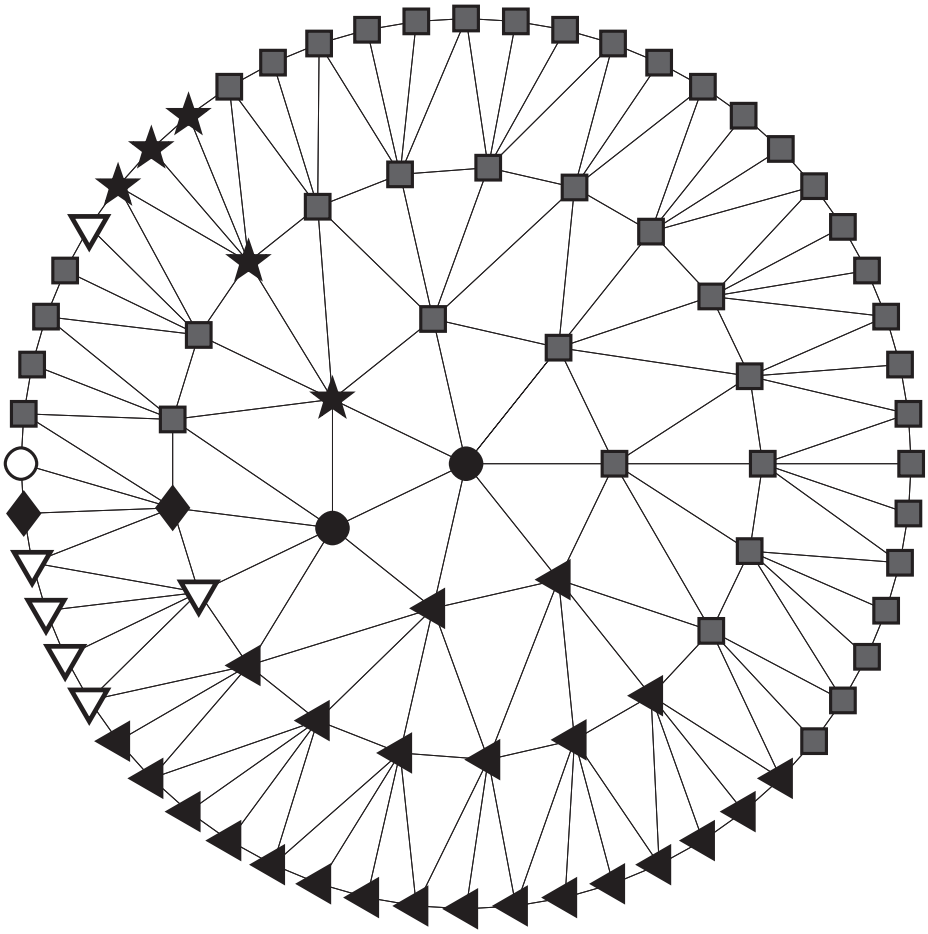


Figure 4.19: Visualization of 183,546 newsgroup articles whose full dissimilarity matrix would occupy approx. 251 GB, instead Patch Relation SOM was applied to 183 patches of around 1000 documents each, reducing drastically the computation time and space needed.

Chapter 5

Summary and Outlook

The concern of the thesis at hand was the processing of data for topographic mapping as well as for clustering and classification. Throughout this work we were exclusively concerned with prototype-based methods because these approaches offer very intuitive learning techniques and their outcomes are meaningful in a way that they can be easily interpreted and visualized. We especially focused on the popular prototype-based methods Neural Gas and Self-Organizing Maps, two representative approaches that have shown their benefits in many applications. Neural Gas is a very robust and reliable vector quantization technique that distributes representative prototypes over a given data manifold in a topology-preserving way. Self-Organizing Maps are a very successful topographic mapping technique that map high-dimensional data into a low-dimensional structure where the data can be visualized. Originally, these methods operate on vectorial data in Euclidean spaces. The ultimate goal of this thesis was the application of Neural Gas and Self-Organizing Maps to data given as pairwise dissimilarities between objects which means in general that there is no vectorial representation available.

In the first part of the thesis, we reformulated the original standard methods Neural Gas and Self-Organizing Maps to run in a so-called *batch* mode. In this mode, all the data points of a finite dataset are considered at once during each update step of the algorithm. It could be demonstrated that the batch variants achieve a higher order of convergence, i.e. they perform much faster while obtaining the same outcome. These fast batch variants can always be used in place of the original ones if the datasets are given in advance.

Next, a few useful extensions of Batch Neural Gas and Batch Self-Organizing Maps were introduced. The first extension aimed for a better control on how the prototypes are distributed according to the data density. We discussed a modification of the update rules that allows for an arbitrary control of the magnification behaviour. This opened the way for outlier suppression, emphasis of rare events, and information optimal maps. In future work, the concept could also be extended to user-interactive magnification of local regions, what would be particularly suited for sophisticated visualization tools in visual analytics.

If further information about the data in form of class labels is available, the additional information can be incorporated into the learning process. It was shown that this supervision helps to improve the classification ability as well as the quality of the topographic maps for visualization.

Especially for the handling of very large datasets, we applied a special process-

ing scheme that is based on patches. In principle this scheme can be applied to all prototype-based methods what is still subject of ongoing work. By taking patches of fixed size one by one, this extension is able to process arbitrary large datasets. Beyond that, the scheme can also be applied to non-stationary distributions in the context of life-long learning systems e.g. in pattern recognition for vision and robotics, or in clustering of data streams.

The main part of the thesis deals with the processing of dissimilarity datasets that are in general of non-Euclidean character. In particular, we adapted the prototype-based methods Neural Gas and Self-Organizing Maps to be able to work on non-vectorial datasets. In a first step, we introduced exemplar-based variants that use datapoints as representative prototypes. Unfortunately, the exemplar-based techniques feature the inherent drawback that the prototypes can only be placed on fixed discrete locations what are in general suboptimal positions, particularly if the data space is sparse. Therefore the main result of the thesis dealt with a further variant that overcomes those limitations.

How to do continuous updates in non-vectorial discrete spaces is not obvious at first feel. For that reason, we assumed an existing embedding into an Euclidean space and introduced prototypes as convex combinations of data points. Based on this assumption, we were able to derive update rules that only rely on the dissimilarities between data objects, and not on any vectorial representation. This led straightforward to Relational Neural Gas and Relational SOM which showed an excellent performance on non-Euclidean dissimilarity datasets in the experiments. We also proposed extensions for magnification control and supervision. Furthermore, the patch scheme was adapted to the relational approach what rendered it possible to generate prototype-based topographic mappings of very large dissimilarity datasets that are far too large to compute all the pairwise dissimilaritites in advance. To present viewable prototypes to the practitioner, the relational prototypes can be approximated by one or more nearest data points which are then presented instead as representatives.

All in all, the relational methods that were developed within this thesis are very well suited for a broad range of applications in bioinformatics, experimental physics, web mining, robotics, and many other fields of science. Because of their simplicity, the practitioner only needs to define a dissimilarity measure for the data objects at hand and yet she has a collection of tools available providing her with intuitive prototype-based topographic mapping and clustering techniques. It should be noted here that the crucial point in that process is the evaluation of the defined dissimilarity measures. This evaluation has to be done by experts in the field who can be supported by proper visualizations of prototypes and data generated by the relational methods.

The research that was done in this thesis opens the way to interesting research directions. We would like to mention just a few:

Since the proposed relational methods showed excellent performance in practise also on non-Euclidean datasets, it would be very interesting to accompany the algorithms by a deeper mathematical investigation of their behaviour in pseudo-Euclidean spaces.

Obviously, for all relational methods the choice of an appropriate metric or dissimilarity measure is crucial for their performance. Since an optimum choice depends very much on the application at hand, it would be valuable to study whether an adaptation of the metrics is possible in this framework. Alternatively, an al-

gorithm to pick an appropriate metric from a given finite set of metrics could be fruitful, which could be based on general principles like clustering quality, for instance. Moreover, experts could be provided with a toolbox of metrics for common data structures or application areas such as they are already partially available in bioinformatics. A combination of these three approaches in an interactive framework seems to be very promising for the next generation of data analysis tools.

As it was demonstrated, the patch paradigm constitutes an intuitive and efficient tool which turns standard prototype-based methods into an applicable model for huge datasets. Unfortunately, it seems that formal guarantees for the approximation quality are difficult to obtain. Since the parameter k of the k-approximation is important for the quality of the results, heuristical approaches are imaginable to automatically determine k such that a proper balance between accuracy and efficiency is obtained according to the application at hand. Also a variation of the patch size over the course of learning seems to be useful, since more information should be preserved in earlier stages of the process. It has been demonstrated that non-stationary distributed incoming data can be handled using the patch scheme. Thus, the patch methods contribute not only to the processing of very large datasets but they also opens the way to life-long learning.

The methods developed within this thesis provide one of the rare linear prototype-based schemes for the processing of dissimilarity datasets which are suitable for data analysis in many different fields of science. To finally prove this claim it remains to apply and evaluate the methods together with experts from different application fields. For this reason, the methods should be integrated into a toolbox that provides nearly parameter free and robust access to the proposed algorithms for the practitioners.

Bibliography

N. Alex and B. Hammer. Parallelizing single patch pass clustering. In *Proc. European Symposium on Artificial Neural Networks (ESANN 2008)*, pages 227–232, 2008.

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

F. Aurenhammer. Voronoi diagrams - a study of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

F. R. Bach. Graph kernels between point clouds. In *Proc. ICML '08*, pages 25–32. ACM, 2008.

H.-U. Bauer and K. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

H.-U. Bauer and T. Villmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8:218–226, 1997.

H.-U. Bauer, M. Herrmann, and T. Villmann. Neural maps and topographic vector quantization. *Neural Networks*, 12:659–676, 1999.

C. M. Bishop, M. Svensén, and C. K. I. Williams. Developments of the generative topographic mapping. *Neurocomputing*, 21:203–224, 1998a.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: the generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998b.

I. Borg and P. J. Groenen. *Modern Multidimensional Scaling – Theory and Applications*. Springer, New York, 2005.

L. Bottou and Y. Bengio. Convergence properties of the kmeans algorithm. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Denver, 1995.

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.

P. Brucker. On the complexity of clustering algorithms. In R. Henn, B. Corte, and W. Oletti, editors, *Optimierung und Operations Research – Lecture Notes in Economics and Mathematical Systems*, pages 44–55. Springer, 1978.

J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7(4):845–865, 1995.

K. Chang and J. Ghosh. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41, 2001.

K. Chang and J. Ghosh. Probabilistic principal surfaces. In *Proc. IEEE International Joint Conference on Neural Networks*, 1999.

Y. Cheng. Convergence and ordering of Kohonen's batch map. *Neural Computation*, 9(8):1667–1676, 1997.

R. Cilibrasi and M. Vitáni. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

J. C. Claussen and H. G. Schuster. Asymptotic level density of the elastic net self-organizing feature map. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 939–944. Springer, 2002.

B. Conan-Guez, F. Rossi, and A. E. Golli. Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19(6):855–863, 2006.

M. Cottrell, J. Fort, and G. Pagès. Two or three things that we know about the Kohonen algorithm. In *Proc. European Symposium on Artificial Neural Networks (ESANN 1994)*, pages 235–244, 1994.

M. Cottrell, J. C. Fort, and G. Pagès. Theoretical aspects of the SOM algorithm. In *Neurocomputing*, pages 119–138, 1998.

M. Cottrell, B. Hammer, A. Hasenfuß, and T. Villmann. Batch neural gas. In *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM 2005)*, pages 275–282, 2005.

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

R. Durbin and D. Willshaw. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326:689–691, 1987.

A. El Golli, B. Conan-Guez, and F. Rossi. A self organizing map for dissimilarity data. In D. Banks, L. House, F. R. McMorris, P. Arabie, and W. Gaul, editors, *Classification, Clustering, and Data Mining Applications (Proceedings of IFCS 2004)*, pages 61–68, 2004.

E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biol. Cybern.*, 67:47–55, 1992.

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

M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

J. Fort, P. Letrémy, and M. Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In *Proc. European Symposium on Artificial Neural Networks (ESANN 2002)*, pages 223–230, 2002.

J.-C. Fort. SOM's mathematics. *Neural Networks*, 19(6-7):812–816, 2006.

J.-C. Fort and G. Pagès. On the a.s. convergence of the Kohonen algorithm with a general neighborhood function. *Ann. Appl. Probab.*, 5(4):1177–1216, 1995.

J.-C. Fort and G. Pagès. Quantization vs. organization in the Kohonen S.O.M. In *Proc. European Symposium on Artificial Neural Networks (ESANN 1996)*, pages 85–89, 1996.

A. Forti and G. L. Foresti. Growing hierarchical tree SOM: An unsupervised neural network with dynamic topology. *Neural Networks*, 19(10):1568–1580, 2006.

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

B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

B. Fritzke. A self-organizing network that can follow non-stationary distributions. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks (ICANN 1997)*, volume 1327 of *Lecture Notes in Computer Science*, pages 613–618, Berlin, 1997. Springer.

J. Geffroy. Sur l'estimation d'une densité dans un espace métrique. *C.v.Acad.Sci.Paris Sér.A*, 278:1449–1452, 1974.

A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

I. Gohberg, P. Lancaster, and L. Rodman. *Indefinite Linear Algebra and Applications*. Birkhauser, 2005.

L. Goldfarb. A unified approach to pattern recognition. *Pattern Recognition*, 17(5): 575–582, 1984.

J. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, 1986.

J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338, 1966.

T. Graepel and K. Obermayer. A stochastic self-organizing map for proximity data. *Neural Computation*, 11(1):139–155, 1999.

T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in neural information processing systems (NIPS)*, pages 438–444, 1999.

S. Graf and H. Luschgy. *Foundations of Quantization for Probability Distributions*. Springer, Berlin, 2000.

S. Graf and H. Luschgy. Rates of convergence for the empirical quantization error. *Ann. Probab.*, 30(2):874–897, 2002.

P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9:189–208, 1983.

R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998.

B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In *Pattern Recognition - Proc. of the 26th DAGM Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 220–227, Berlin, 2004. Springer.

B. Hammer and A. Hasenfuss. Relational neural gas. In J. Hertzberg, M. Beetz, and R. Englert, editors, *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 190–204, Berlin, 2007. Springer.

B. Hammer and T. Villmann. Mathematical aspects of neural networks. In *Proc. European Symposium on Artificial Neural Networks (ESANN 2003)*, pages 59–72, 2003.

B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, 21:21–44, 2005.

B. Hammer, A. Hasenfuss, F.-M. Schleif, and T. Villmann. Supervised median clustering. In C. H. Dagli, editor, *Smart Systems Engineering: Infra-Structure Systems Engineering, Bio-Informatics and Computational Biology, and Evolutionary Computation, Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2006)*, volume 16 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 623–632, New York, 2006a. ASME Press. ISBN 0-7918-0256-6.

B. Hammer, A. Hasenfuss, F.-M. Schleif, and T. Villmann. Supervised batch neural gas. In F. Schwenker and S. Marinai, editors, *Artificial Neural Networks in Pattern Recognition, Second IAPR Workshop (ANNPR 2006)*, volume 4087 of *Lecture Notes in Artificial Intelligence*, pages 33–45, Berlin, 2006b. Springer.

B. Hammer, A. Hasenfuß, and T. Villmann. Magnification control for batch neural gas. In *Proceedings of the 14th European Symposium on Artificial Neural Networks (ESANN 2006)*, pages 7–12, 2006c.

B. Hammer, A. Hasenfuss, F. Rossi, and M. Strickert. Topographic processing of relational data. In *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)*, 2007a.

B. Hammer, A. Hasenfuss, and T. Villmann. Magnification control for batch neural gas. *Neurocomputing*, 70(7–9):1225–1234, 2007b.

J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

A. Hasenfuss and B. Hammer. Relational topographic maps. In M. R. Berthold, J. Shawe-Taylor, and N. Lavrac, editors, *Advances in Intelligent Data Analysis VII, Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA 2007)*, volume 4723 of *Lecture Notes in Computer Science*, pages 93–105, Berlin, 2007. Springer.

A. Hasenfuss and B. Hammer. Single pass clustering and classification of large dissimilarity datasets. In *Artificial Intelligence and Pattern Recognition (AIPR 2008)*, pages 219–223, 2008.

A. Hasenfuss, B. Hammer, F.-M. Schleif, and T. Villmann. Neural gas clustering for dissimilarity data with continuous prototypes. In F. Sandoval, A. Prieto, J. Cabestany, and M. Grana, editors, *Computational and Ambient Intelligence – Proceedings of 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, volume 4507 of *Lecture Notes in Computer Science*, pages 539–546, Berlin, 2007. Springer.

A. Hasenfuss, W. Boerger, and B. Hammer. Topographic processing of very large text datasets. In C. H. Dagli, editor, *Smart Systems Engineering: Computational Intelligence in Architecting Engineering Systems (ANNIE 2006)*, volume 18 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 525–532, New York, 2008a. ASME Press.

A. Hasenfuss, B. Hammer, T. Geweniger, and T. Villmann. Magnification control in relational neural gas. In *Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN 2008)*, pages 325–330, 2008b.

A. Hasenfuss, B. Hammer, and F. Rossi. Patch relational neural gas - clustering of huge dissimilarity datasets. In L. Prevost, S. Marinai, and F. Schwenker, editors, *Artificial Neural Networks in Pattern Recognition, Third IAPR Workshop (ANNPR 2008)*, volume 5064 of *Lecture Notes in Artificial Intelligence*, pages 1–12, Berlin, 2008c. Springer.

T. Hastie. *Principal Curves and Surfaces*. PhD thesis, Stanford University, 1984.

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

R. J. Hathaway and J. C. Bezdek. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11(4):351–368, 2003.

R. J. Hathaway and J. C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, 27(3):429–437, 1994.

R. J. Hathaway, J. M. Huband, and J. C. Bezdek. Kernelized non-euclidean relational fuzzy c-means algorithm. In *The 14th IEEE International Conference on Fuzzy Systems*, pages 414–419, 2005.

T. Heskes. Transition times in self-organizing maps. *Biological Cybernetics*, 75(1): 49–58, 1996.

T. Heskes. Energy functions for self-organizing maps. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 303–315. Elsevier, Amsterdam, 1999.

T. Heskes and B. Kappen. Error potentials for self-organization. In *IEEE International Conference on Neural Networks*, volume 3, pages 1219–1223, 1993.

H. Hirsh. Data mining research: Current status and future opportunities. *Statistical Analysis and Data Mining*, 1(2):104–107, 2008.

V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

T. Hofmann and J. M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, 33:617–634, 1999.

H. Hotelling. Analysis of a complex statistical variable into principal components. *J. Educational Psychology*, 24:417–441, 1933.

P. Indyk and J. Matoušek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106 (4):620–630, 1957a.

E. T. Jaynes. Information theory and statistical mechanics ii. *Physical Review*, 108 (2):171–190, 1957b.

I. Jolliffe. *Principal Component Analysis*. Springer, New York, 2002.

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 2000*, volume 2, pages 680–683, 2000.

H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. In B. Schoelkopf, K. Tsuda, and J. Vert, editors, *Kernel Methods in Computational Biology*, pages 155–170. MIT Press, 2004.

S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1981-1997. *Neural Computing Surveys*, 1:102–350, 1998.

S. Kaski, J. Nikkila, M. Oja, J. Venna, P. Toronen, and E. Castren. Trustworthiness and metrics in visualizing similarity of gene expression. *BMC Bioinformatics*, 4 (48), 2003.

S. Kaski, J. Nikkila, E. Savia, , and C. Roos. Discriminative clustering of yeast stress response. In U. Seiffert, L. Jain, and P. Schweizer, editors, *Bioinformatics using Computational Intelligence Paradigms*, pages 75–92. 2005.

D. A. Keim, F. Mansmann, D. Oelke, and H. Ziegler. Visual analytics: Combining automated discovery with interactive visualizations. In J.-F. Boulicaut, M. R. Berthold, and T. Horváth, editors, *Discovery Science*, volume 5255 of *Lecture Notes in Computer Science*, pages 2–14. Springer, 2008.

T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15:945–952, 2002.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–94, 1980.

R. Linsker. How to generate maps by maximizing the mutual information between input and output signals. *Neural Computation*, 1:402–411, 1989.

S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

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

S. P. Luttrell. A Bayesian analysis of self-organizing maps. *Neural Computation*, 6 (5):767–794, 1994.

D. MacDonald and C. Fyfe. The kernel self-organising map. In R. J. Howlett and L. C. Jain, editors, *Proc. KES 2000*, volume 4, pages 317–320. IEEE, 2000.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(3):507–522, 1994.

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.

J. Matoušek. *Lectures on Discrete Geometry*. Springer, New York, 2002.

J. Matoušek. Open problems on low-distortion embeddings of finite metric spaces. `http://kam.mff.cuni.cz/~matousek/metrop.ps`, 2007.

E. Merényi and A. Jain. Forbidden magnification? ii. In *Proc. European Symposium on Artificial Neural Networks (ESANN 2004)*, pages 57–62, 2004.

H. Mevissen and M. Vingron. Quantifying the local reliability of a sequence alignment. *Protein Engineering*, 9:127–132, 1996.

C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, 2000.

B. Mokbel, A. Hasenfuss, and B. Hammer. Graph-based representation of symbolic musical data. In A. Torsello, F. Escolano, and L. Brun, editors, *Graph-Based Representation in Pattern Recognition (GbRPR 2009)*, volume 5534 of *Lecture Notes in Computer Science*, pages 42–51, Berlin, 2009. Springer.

F. Mulier and V. Cherkassky. Self-organization as an iterative kernel smoothing process. *Neural Computation*, 7(6):1165–1177, 1995.

M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10), 2006.

E. Oja. Oja learning rule. *Scholarpedia*, 3(3):3612, 2008.

E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992.

M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. *Neural Computing Surveys*, 3:1–156, 2003.

J. Ontrup and H. Ritter. Hyperbolic self-organizing maps for semantic navigation. In *Advances in neural information processing systems 14 (NIPS)*, pages 1417–1424, 2001.

U. Ozertem. *Locally Defined Principal Curves and Surfaces*. PhD thesis, Oregon Health & Science University, 2008.

M. Parsana, S. Bhattacharya, C. Bhattacharya, and K. R. Ramakrishnan. Kernels on attributed pointsets with applications. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1129–1136. MIT Press, Cambridge, MA, 2008.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

E. Pekalska and R. P. W. Duin. *The Dissimilarity Representation for Pattern Recognition*. World Scientific Publishing, 2005.

E. Pekalska, P. Paclik, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2:175–211, 2001.

M. Pöllä, T. Honkela, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 2002-2005 addendum. *Neural Computing Surveys*, 2007.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

Y. Prudent and A. Ennaji. An incremental growing neural gas learns topology. In *Proc. IJCNN 2005*, 2005.

A. K. Qinand and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. In *Proc. ICPR 2004*, volume 17(4), pages 617–620. IEEE, 2004.

J. Qiu, M. Hue, A. Ben-Hur, J.-P. Vert, and W. S. Noble. A structural alignment kernel for protein structures. *Bioinformatics*, 23(9):1090–1098, 2007.

B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge, 1996.

H. Ritter. Self-organizing maps in non-euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–108. 1999.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2239, 1998.

K. Rose, E. Gurewitz, and G. Fox. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990.

K. Rose, E. Gurewitz, and G. Fox. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, 1992.

F. Rossi. Model collisions in the dissimilarity SOM. In *Proc. European Symposium on Artificial Neural Networks (ESANN 2007)*, pages 25–30, 2007.

F. Rossi, A. Hasenfuß, and B. Hammer. Accelerating relational clustering algorithms with sparse prototype representation. In *Proceedings of 6th International Workshop on Self-Organizing Maps (WSOM 2007)*, 2007.

V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25:2003, 2003.

J. Rynkiewicz. Self-organizing map algorithm and distortion measure. *Neural Networks*, 19(6-7):830–837, 2006.

A. Saalbach, T. Twellmann, A. Wismüller, J. Ontrup, H. Ritter, and T. W. Nattkemper. A hyperbolic topographic mapping for proximity data. In *Proc. of the IASTED International Conference on Artifical Intelligence and Applications*, pages 106–111. ACTA Press, 2005.

A. A. Sadeghi. Convergence in distribution of the multi-dimensional kohonen algorithm. *Journal of Applied Probability*, 38(1):136–151, 2001.

A. A. Sadeghi. Asymptotic behavior of self-organizing maps with nonuniform stimuli distribution. *The Annals of Applied Probability*, 8(1):281–299, 1998.

J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, 18(5):401–409, 1969.

C. Saunders, J. Shawe-Taylor, and A. Vinokourov. String kernels, Fisher kernels and finite state automata. In *Advances in Neural Information Processing Systems 15*, pages 633–640. MIT Press, 2003.

V. Sauvage. The T-SOM (tree-SOM). In *AI '97: Proceedings of the 10th Australian Joint Conference on Artificial Intelligence*, pages 389–397, 1997.

F.-M. Schleif, B. Hammer, and T. Villmann. Margin based active learning for LVQ networks. *Neurocomputing*, 70(7-9):1215–1224, 2007.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons, 1992.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

S. J. Simoff, M. H. Boehlen, and A. Mazeika, editors. *Visual Data Mining – Theory, Techniques and Tools for Visual Analytics*, volume 4404 of *Lecture Notes in Computer Science*. Springer, 2008.

M. Strickert, U. Seiffert, N. Sreenivasulu, W. Weschke, T. Villmann, and B. Hammer. Generalized relevance LVQ (GRLVQ) with correlation measures for gene expression analysis. *Neurocomputing*, 69:651–659, 2006.

H. Tong. *Dimension Estimation and Models.* World Scientific Publishing, 1993.

W. S. Torgerson. Multidimensional Scaling: I. Theory and Method. *Psychometrika*, 17(4):401–419, 1952.

W. S. Torgerson. *Theory & Methods of Scaling.* Wiley, 1958.

J. Venna and S. Kaski. Local multidimensional scaling. *Neural Networks*, 19(6-7): 889–899, 2006.

J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. SOM toolbox for MATLAB 5. Report A57, Helsinki University of Technology, April 2000.

N. Villa and F. Rossi. A comparison between dissimilarity SOM and kernel SOM for clustering the vertices of a graph. In *Proc. WSOM 2007*, 2007.

T. Villmann. Controlling strategies for the magnification factor in the neural gas network. *Neural Network World*, 10(4):739–750, 2000.

T. Villmann. *Neural Maps and Learning Vector Quantization for Data Mining – Theory and Applications.* Habilitationsschrift, University of Leipzig, 2004.

T. Villmann. *Topologieerhaltung in selbstorganisierenden neuronalen Merkmalskarten.* Ph.d. thesis, University of Leipzig, 1997.

T. Villmann and J. C. Claussen. Magnification control in self-organizing maps and neural gas. *Neural Computation*, 18(2):446–469, 2006.

T. Villmann and A. Heinze. Application of magnification control for the neural gas network in a sensorimotor architecture for robot navigation. In H.-M. Gross, K. Debes, and H.-J. Bohme, editors, *SOAVE 2000 – Selbstorganisation von adaptivem Verhalten*, pages 125–134. VDI-Verlag, 2000.

T. Villmann, R. Der, M. Herrmann, and T. Martinetz. Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, 1997.

T. Villmann, E. Merényi, and B. Hammer. Neural maps in remote sensing image analysis. *Neural Networks*, 16(3-4):389–403, 2003.

T. Villmann, B. Hammer, F. Schleif, T. Geweniger, and W. Herrmann. Fuzzy classification by fuzzy labeled neural gas. *Neural Netw.*, 19(6):772–779, 2006.

C. von der Malsburg and D. J. Willshaw. How to label nerve cells so that they can interconnect in an ordered fashion. *Proceedings of the National Academy of Sciences of the United States of America*, 74(11):5176–5178, 1977.

R. Weller. Inner sphere trees for proximity and penetration queries. Technical report, TU Clausthal, 2008.

D. J. Willshaw and C. von der Malsburg. How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 194(1117):431–445, 1976.

T. Winkler, J. Drieseberg, A. Hasenfuss, B. Hammer, and K. Hormann. Thinning mesh animations. In O. Deussen, D. Keim, and D. Saupe, editors, *Proceedings of Vision, Modeling, and Visualization (VMV 2008)*, 2008.

W. Wolberg, W. Street, D. Heisey, and O. Mangasarian. Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology*, 26:792–796, 1995.

Q. Yang and X. Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(4):597–604, 2006.

G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–21, 1938.

P. L. Zador. *Development and Evaluation of Procedures for Quantizing Multivariate Distributions*. Ph. D. Dissertation, Stanford University, 1963.

P. L. Zador. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, 28(2):139–148, 1982.