University of Osnabrück
Institute of Computer Science
Department of Mathematics and Computer Science

Self-Organizing Neural Networks for Sequence Processing

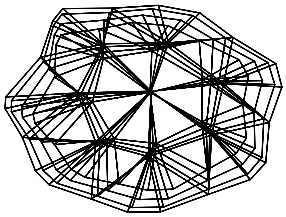
Thesis by

Marc Strickert

In partial fulfillment of the requirements

for the degree of

Doctor of Computer Science (Dr. rer. nat.)



Osnabrück, Germany
June 7, 2004

Referees:

Prof. Dr. Barbara Hammer,
Research Group 'Learning with Neural Methods on Structured Data' (LNM),
Technical University of Clausthal, Germany.

Prof. Dr. Helge Ritter,
Faculty of Technology, Neuroinformatics Group,
University of Bielefeld, Germany.

Self-Organizing Neural Networks for Sequence Processing

Arbeitsgruppe LNM
Institut für Informatik
am Fachbereich für Mathematik, Informatik
und Angewandte Systemwissenschaft
Universität Osnabrück
D-49069 Osnabrück, Germany

Arbeitsgruppenleiterin:
Dr. Barbara Hammer

Institutsdirektor:
Prof. Dr. Oliver Vornberger

Autor:
Dipl.-Systemwiss. Marc Strickert

Osnabrück, Juni 2004

Abstract

This work investigates the self-organizing representation of temporal data in prototype-based neural networks. Extensions of the supervised learning vector quantization (LVQ) and the unsupervised self-organizing map (SOM) are considered in detail. The principle of Hebbian learning through prototypes yields compact data models that can be easily interpreted by similarity reasoning. In order to obtain a robust prototype dynamic, LVQ is extended by neighborhood cooperation between neurons to prevent a strong dependence on the initial prototype locations. Additionally, implementations of more general, adaptive metrics are studied with a particular focus on the built-in detection of data attributes involved for a given classification task. For unsupervised sequence processing, two modifications of SOM are pursued: the SOM for structured data (SOMSD) realizing an efficient back-reference to the previous best matching neuron in a triangular low-dimensional neural lattice, and the merge SOM (MSOM) expressing the temporal context as a fractal combination of the previously most active neuron and its context. The first SOMSD extension tackles data dimension reduction and planar visualization, the second MSOM is designed for obtaining higher quantization accuracy. The supplied experiments underline the data modeling quality of the presented methods.

Keywords: vector quantization, self-organization, relevance learning, classification, clustering, sequence processing, context, fractal representation.

Preface

Efficient data processing is one of the ageless topics in computer science. Numerous algorithms for the exact storage, sorting, and retrieval of data are available for dealing with electronic information. Precise data handling is required for many man-made categories like for those used in filing systems, dictionaries, or financial transactions; however, potentially lossy methods can and should be used for processing noisy or redundant data sources such as large images, blurry data streams, or medical data sets. Yet, lossy data representation bears two major chances: first, in order to capture the essence of what shall be represented, common properties of the data entities must be integrated to characteristic features, this way allowing to focus on relevant and generalizable information also under noisy conditions; second, the computational complexity for operating precisely on data collections can be very high, like for the construction of optimum decision trees, although in practice, faster attainable sub-optimum solutions based on reduced representations might already be sufficient for a given task.

While there are many respectable database algorithms for storage and retrieval, such as k-d-trees for spatial data or partial matching of discrete sequences through dynamic programming, there is still an ongoing quest for finding regularities and particular features according to which large data collections can be made easily accessible to humans. Conceptually different models that use alternative data representations can be found in the domain of *neural* data processing. The term 'neural' is motivated biologically by the fact that, similar to the brain functionality, many simple cooperating units pass each other bits of information, thereby transforming input stimuli into characteristic output states. These states can be further used for classification and association, for the estimation of intermediate positions, or for the prediction of new inputs.

One claim of neural computation is the simplicity of methods – many units operate and pass on data in an elementary processing manner, this way producing distributed and often compact representations. A child recognizes the scope of the word 'apple' usually by presenting a few times a prototypical red exemplar, and then saying that a green apple is also one, the rotten apple, too, and a bitten one is also still an apple. This way, the child's model for pattern identification is adapted, either by taking the presented, so far dissimilar pattern, into the set of prototypes, or by extending the notion, that is, the best existing measure for identifying this pattern is made a little more sloppy to include it next time. Thus, new inputs can be related to already learned items by abstraction: apples need not be stored separately in a costly manner and then recalled by lookup, but only a small number of fundamental apples together with some allowed modifications are required to identify the entirety of apples.

Technically, the task of apple recognition is more difficult than it seems, because different features have to be processed at the same time, for example color, size, shape, and possibly weight and odor. Some of these attributes can be further split into physical units,

such as the size that might be approximated by three values describing the height, the width, and the depth. The description of odor and shape is even more complex for which no standard solution exists. Consequently, attributes exist as inputs in different domains, and they must be made mutually compatible for further processing. Observations may include symbolic features, or natural or real numbers covering certain intervals; therefore, a particular encoding and normalization is required before these attributes can be input to a neural network model. In biology this works naturally: the input of different sensory channels, like visual and auditory stimuli presented to the brain are translated into a universal code of nervous cell spiking and then fused into a unified perception. In theoretical neuroscience, this accomplishment, which is not yet fully understood, is related to the binding problem. In most algorithms, input data are just prepared by transformation into normalized real number vectors of a fixed dimension.

Sequences, however, require different processing methods than vectors of predefined dimension. It is important to manage sequences, because they are the natural domain of physical, chemical, and biological processes described as a function of time. As a matter of interest, repeating patterns of state changes are a first indication of structured underlying driving forces. Sequence and subsequence classification is strongly connected to pattern detection and requires the temporal history, the context, of the current observation to be considered. It is an important task to represent characteristic states and to possibly identify novel situations: challenging applications are the diagnostics in process supervision, the prediction of state changes, and the linkage of predictive models to empiric values.

Putting the parts together, this work will focus on processing labeled and unlabeled sequential data, using biologically motivated self-organizing learning models with adaptive prototypes and metrics. The aims are to reveal important aspects of sequence processing, to introduce new methods for dealing with sequences, and to provide some ideas for the design of future context learning architectures.

This work has been realized in the research group (Forschernachwuchsgruppe) ‘Learning with Neural Methods on Structured Data’ (LNM) funded by the Ministry for Science and Culture (MWK) of Lower Saxony, Germany.

Feeling gratitude and not expressing it
is like wrapping a present and not giving it.

William Arthur Ward

Acknowledgments

Intelligent data processing is the main motivation of my work. I have realized that the field of neural computation with its numerous and very different techniques is a romping place for seminal paradigms and concepts. Self-organizing data models with emergent and adaptive characteristics point into promising directions of future applications and further topics of research.

Prof. Volker Sperschneider has initiated my interest in neural networks with his lectures on this issue. A subsequent programming project, together with Sascha Teichmann, on a network-based realization of the inverse kinematics for 3D-positioning of a robot arm has shown that the practical application of neural methods is challenging and far from trivial.

Dr. Barbara Hammer has encouraged me to apply for a position in her research group 'Learning with Neural Methods on Structured Data' (LNM). I would like to thank her for that leap of faith. The opportunity to work in the group from September 2000 to August 2004 has offered many intriguing perspectives on neural computation. Barbara's analytic expertise, her devotion to scientific activity, and her undamped cheerful disposition created a delightful and inspiring working atmosphere.

Dr. Thomas Villmann has not only been a prolific coauthor of many papers, but he has also contributed fresh and unconventional ideas; may his enthusiasm, his modesty, and his kindness have a lasting effect on me.

Prof. Helge Ritter has done groundbreaking research on artificial neural networks, specifically on self-organizing maps, and I am very glad that, despite his tight schedule, he has agreed on giving his expert opinion about the present work.

Sascha Teichmann's profound knowledge on programming and design concepts have led to many stimulating discussions and to extensive coffee consumption.

M.A. Gundula Wilke has done a great job in giving me hints and advice for improving my usage of the English language in this thesis. Thanks for her patient explanations and corrections.

A number of students have brought additional aspects into my focus of interest. Thanks to them and to the staff of the faculty of Computer Science for their contributions and for supporting my work.

Contents

Part I

Self-organizing neural networks and sequence processing

Chapter 1

Introduction

I don't have any solution, but I certainly admire the problem.

Ashleigh Brilliant

This work refines and proposes artificial neural approaches to process possibly labeled data, structured by spatio-temporal characteristics. Data processing refers to a bunch of operations, like efficient data storage, data lookup and matching similar data, and — on a higher, more aggregated level — information retrieval and extraction.

Natural processing of real-life data

While crisp and accurate data might be best stored in data bases for further analysis, noisy and experience-driven data or very large data sets might be more compactly represented at a higher level of abstraction. Thus, a robust alternative to table-based data storage is sought for an efficient representation of noisy real-life data exhibiting also temporal structure.

A natural source of inspiration for tackling this problem is the brain of higher order mammals which has developed the ability to extract the most interesting aspects from an environment with a variety of sensual stimuli: within the focus of attention, the tracking of actions refers to spatial and temporal changes which, by the laws of physics, are related to a continuous flow of similar sensory information for a certain period of time. For example, listening to a speaker is essentially observing a temporal frequency-intensity modulation of highly redundant acoustic data. At first, a baby cannot assign a high level meaning to the sounds, but it learns templates from the sentence melodies, like different voices, reoccurring patterns in the transition of consonants and vowels, or the lowering and rising pitch. Over time, a rough structure of the articulation space emerges which is characterized by similar or dissimilar states. The next step is to assign meaning to the states, to put a label on them, as the sound of the name 'Timmy' coinciding with the visual presence of the speaker talking to the little child, this way consolidating the association of feeling

addressed by a name. Finally, expert listeners have learned to assign the correct meaning to known words and to memorize new meanings from given definitions. Thus, the natural stages are from the unsupervised raw structure learner via the self-supervised probabilistic associator to the supervised classifier.

Another aspect of the flexibility of the human brain is that different speakers, although producing different sounds, might still convey exactly the same information to the listener. This is a simple yet impressive example of how data are captured efficiently: the input stimuli stream is online reduced to the interesting entities by recognizing characteristic abstract acoustic features eliminating everything not related to the task of understanding the speech or distinguishing the speakers. Instead of complete and costly data storage, a compact model focusing on the representation of certain aspects tackles this real-time problem efficiently.

Exhaustive approaches with precise but computationally very expensive operations are not within the scope of this work. Due to the basic supposition that particularly spatio-temporal real-life data are often redundant, compact models that do not store every detail are discussed in the following. As indicated above, the main focus is put on unsupervised and supervised neural models which yield an efficient representation of the input space.

Classical neural network architectures

Many different types of artificial neural networks exist for data interpolation and classification. Early ideas were formulated by McCulloch and Pitts in 1943, who showed how simple models of nervous cell systems are able to perform logical computations [103]. In 1949, Hebb proposed a neural dynamic which integrated weighted input channels by means of summation nodes. Hebb used the additional assumption that if two of these simple computing units are simultaneously active, the weight between them is increased [66]. Suitably adapted variants of this Hebbian learning paradigm are found in many algorithms.

Very prominent learning architectures are layered feed-forward neural networks which propagate the input vector via weighted connections through internal layers of neural amplifiers to an output vector. Learning takes place in a supervised manner by the adaptation of the connection weights according to the backpropagated error between the network output and the desired output [102]. This network type is known as universal approximator for any functional input-output relation, if enough units exist in the hidden layer(s) [72]. Extensions to temporal data processing can be obtained by unfolding time into neuron layers, a process which is related to error backpropagation-through-time and real-time recurrent learning [115, 170]. A partially recurrent architecture that feeds back the net output to the context input neurons has been suggested by Jordan [75]. Elman's modification provides context feedback neurons associated with each neuron of one internal layer or, in case of hierarchical networks, several layers [39]. The training methods for the above approaches are variants of backpropagation, or they are more general gradient descent techniques for minimizing the error. Kolen and Kremer give an overview of the

methods in [85]. A serious open problem of backpropagation networks is the exponential error decay which leads to slow training convergence and to possibly suboptimum solutions especially for networks with many layers [50]. A specific survey of neural network models for time series processing is given by Dorffner [35]. As a precondition for the supervised backpropagation, training requires labeled input samples. However, these are not always available; therefore, one focus of this work is the unsupervised processing of temporal data.

Prototype-based data representation

The path taken in this work is an alternative one: instead of training the connection weights between neurons of feed-forward networks, a more direct data representation is obtained by using prototype models. Prototypes are entities that can be adapted towards states encountered in the training set, which means that the similarity of a prototype is increased to what it stands for. Prototype representations store information in plain vector locations rather than in the forward-propagated, transformed network output states. Usually, the represented state is a prominent feature of the input data, and data points near a prototype are supposed to share the prototype's properties, if any given, e.g. a class membership. Different characteristic input states should be handled by different prototypes, therefore the number of prototypes is related to the granularity of the overall data representation. If no property or label is assigned, a single prototype should account for the fact that it is likely to find data in its neighborhood. The collectivity of all prototypes constitutes a rough sketch of the input space. After training, meaning can be manually associated with the found prototype locations. Alternatively, several models can be trained on different aspects of the input data and linked together in order to synchronize unlabeled multi-sensory signals and to generate meaningful self-supervised states.

Throughout this work, it is supposed that the natural redundancy of real life-data can be robustly captured by a relatively small number of prototypes. The basic ingredients for a prototype model are $\langle a \rangle$ an appropriate metric in the input space in order to measure the similarity between data and prototypes, $\langle b \rangle$ a certain number of prototypes, and $\langle c \rangle$ a cooperation strategy to avoid the wasting of prototypes. The first point refers to the description of the data clustering, the second point to the model size, and the last point means that neighboring prototypes occupying the same location must negotiate a specialization precedence or a spreading scheme.

Self-organizing learning

With respect to the prototype-based data representation, the involved methods are predestined to be using concepts of self-organization. On one hand, neural methods are called self-organizing, if they produce structured representations of the training stimuli. An explicit target function might not be given for the ordering process; still, the implicit goal might be a similarity-based mapping of the input to internal states, thereby reducing the adaptation efforts or the costs connected with learning. On the other hand,

self-organization refers to competition or cooperation during the learning process: a neuron that is most responsive to a stimulus is selected as winner, and it takes much of the overall available update energy for specializing on the given input, whereas less matching prototypes are only slightly updated.

While the learning vector quantization (LVQ, in Section 6, page 18) for labeled data is determined by a *supervised* competitive winner-takes-all (WTA) dynamic, the self-organizing map model (SOM, in Section 6, page 68) with its neighborhood interactions is an example of *unsupervised* competitive learning in terms of a metric-based clustering of unlabeled data. Both methods share, as a feature of self-organization, the ability to generate global neural orderings by local updates, and both implement Hebbian learning. The main purpose is to solve classification tasks by exploiting certain data characteristics in order to emphasize similarities and differences: LVQ learns distinct class labels, the SOM assigns new data to the most similar data clusters which exist as projections on a low-dimensional grid. Thus, LVQ can be trained as an expert for a finite set of decisions for given data, and the SOM can be used for unsupervised detection of similarity features in the data.

The two methods SOM and LVQ are simple and robust, and they allow a wide range of applications. Moreover, they are well suited for the formulation of generalizations. For certain tasks, specialized methods like those given by Duda et al. [37] might perform better than the extensions discussed in this work. However, the challenge is to obtain light and good data models for very different domains. Particularly, sequential data are of interest in many scientific disciplines. Some important topics are the DNA sequence screening in biology, complex system characterization in physics and ecology, diagnostics in medicine, energy consumption and market analysis in economics, real-time signal processing in engineering, language processing in telecommunications and linguistics, and trajectory description of end-effectors in robotics. The common goals of these examples are process analysis and control based on feature extraction, subsequence classification, and future prediction. This work will contribute to some of these interesting issues.

Scientific contribution and organization of this work

The structural simplicity of the basic LVQ algorithm, and an intuitive understanding of what's being done in the supervised case, make it a good starting point for self-organizing vector quantizers. Its extensions by a cost function to relevance learning are reported together with experiments in the first part of this thesis. The original formulation of relevance adaptation for LVQ in terms of GRLVQ date back to early work of Hammer and Villmann [65]; however, in cooperation with them and with other researchers, the author has contributed to a number of publications about the successful line of GRLVQ developments: the papers can be split into ⟨1⟩ extensions of GRLVQ to initialization independence, general metrics, and rule extraction [57, 61, 162, 164], into ⟨2⟩ theoretic considerations [60, 62], and into ⟨3⟩ studies and applications [15, 58, 59, 63, 144]. In particular, the author has a share in the extension of GRLVQ to general adaptive metrics, he has provided the corresponding program implementations as well as the subsequent computer experiments.

In the second part, the unsupervised SOM is discussed. Its extension to sequence processing leads to a subdivision into two sections. One section is dealing with the author's modification of Hagenbuchner's SOM for structured data (SOMSD) to possibly hyperbolic target grids for temporal processing [146] as a special instance within a general framework for unsupervised processing of structured data [55]. The other section presents the new merge context model (MSOM) proposed by the author [145, 147], which can be put again into a more general framework [56].

Before specific learning architectures are discussed, some notation conventions will be agreed on, and data preprocessing for temporal neural networks will be briefly revisited.

Chapter 2

Notations and conventions

What usually comes first is the contract.

Ira Gershwin

In order to avoid misunderstandings, the usage of some terms is defined. The word *dimension* refers either to the number of entries in the data vectors or to the data column addressed by a specific index, e.g. the i^{th} dimension is the projection of the data vectors to their i^{th} component. A *pattern* is equivalent to *data point* which may be represented by the *weight vector* of a *prototype* neuron. The prototype closest to the currently presented pattern is referred to as the *winner*, regardless of its represented class. As usual, a *cycle* is an epoch referring to one presentation of all data patterns in a training set; this must not be confused with *iteration* which describes the presentation of only a single pattern from the training set. The temporal *context* refers to the patterns that have just been presented before the currently processed data item, and the spatial context is related to the environment of a hot spot in a pattern vector or in a matrix.

The most common symbols in this work are listed in Table 2.1.

Symbol	Meaning
d	Pattern dimension
m	Number of prototypes
n	Number of data
c	Number of classes
γ	General learning rate
γ^+/γ^-	LVQ learning rate for correct/wrong prototype
$\gamma^{\boldsymbol{\lambda}}$	LVQ learning rate for relevances
η	Learning fade rate
$\mathbf{M}$	Matrix notation of M
$\boldsymbol{v}$	Vector notation of v
$\mathcal{X}$	Pattern space
$\mathbf{X}$	Pattern set
$\mathbf{x}$	Pattern vector
x_j^i	Component with index j of pattern vector i
$\mathcal{W}$	Weight space
$\mathbf{W}$	Weight set
w_j^i	Component with index j of weight vector i
$\mathcal{C}$	Context space
$\boldsymbol{c}$	Context vector
c_j	Component with index j of context vector
α	Context-weight balance for distance calculation
β	Context-weight merging parameter for linear combination
c	Context back-reference representation
$\boldsymbol{\lambda}$	Relevance profile vector
λ_j	Dimension relevance with index j
$\mathcal{Y}$	Label space
$\mathbf{y}$	Label notation of y
f	Feature type setting of f
s	Symbol type setting of s
c	Class
N	Node (neuron) type setting of N
I_j	Winner index when sequence element $\mathbf{a}^j$ is presented
$\chi_{\mathbf{x}}(\boldsymbol{w})$	Contribution of prototype $\boldsymbol{w}$ given pattern $\mathbf{x}$
ξ	Noise variable
d	Distance function
E	Cost function
h	Neighborhood function
e	Quantization error
sgd	Sigmoidal function
rnk	Rank function
$\langle A\rangle$,$\langle B\rangle$	Within text enumeration
xdata	Data set emphasis
http://	URL emphasis
ALGO	Algorithm emphasis

Table 2.1: Table of used symbols and notations.

Chapter 3

Data preprocessing for temporal networks

Learning is finding out what you already know.

Richard Bach

In most cases a data set at hand is too raw to be just presented to a neural network. Real world data are generally

- **misformatted:** having attributes with inappropriate notations or data types;
- **inconsistent:** containing discrepancies in codes or names;
- **incomplete:** lacking attribute values or containing only aggregate data;
- **noisy:** containing errors or outliers;
- **instationary:** exhibiting temporal trends and process changes.

Therefore, data preprocessing should be considered before data are fed to a neural network: any effort to relax a learner's task should be made. Data preparation includes steps of

- **conversion:** turning attribute data types into allowed representation;
- **cleaning:** filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies;
- **transformation:** mapping data to another representation, for example to a normalized one;
- **discretization:** replacing numerical attributes with atomic states.

Although an all-in-one solution for data processing by a single neural network is desirable, training can be greatly enhanced in many cases, if data are carefully put into suitable format. Since students and researchers in the domain of neural computation tend to pay attention on the methods rather than on the data, this section refocuses on several data preparation techniques in order to prevent an application of neural methods to inappropriate data. The steps of data preprocessing and algorithm design are mutually dependent,

both take influence on the final results. Complementary to adapting a neural technique to problem-specific data, results can be significantly improved by selecting algorithm-specific data; however, the data set selection task is beyond the scope of this work, therefore, the interested reader is referred to the contribution of LaLoudouanna and Tarare [92].

Conversion

The most convenient data type for arithmetics on attributes is a real number, making calculations like summations or interpolations possible; therefore, a conversion of attributes into this target format is an appealing operation. However, this rendering to real values may lead to accidental abuse of the gained representations. A standard example is the inappropriate calculation of the expected value of a thrown die by $\mu = (1+\ldots+6)/6 = 3.5$. In this case, the sides of the cube are independent symbols for which, misleadingly, an ordinal semantics is agreed on. Similar considerations apply to ordinal survey data for which answers to the exemplary question "How do You find X — 'bad', 'indifferent', or 'good' ?" might be encoded by the three values $\{-1, 0, 1\}$. The match between empiric scales and equidistant numbers is always problematic though. An example for nominal data is the ordering of colors, for which the physiological perception does not coincide with colors ordered by the distances in an auxiliary RGB color cube.

A very conservative assumption is the distinct enumeration of the possible attribute states and their encoding by unary vectors $\boldsymbol{u}^i$ that accentuate the component index j matching the state number i: $\boldsymbol{u}^i_j = 1$ for $j = i$, else $\boldsymbol{u}^i_j = 0$. For example, the odor representations of a mushroom by almond=$(0, 0, 0, 1)$, fishy=$(0, 0, 1, 0)$, musty=$(0, 1, 0, 0)$, and spicy=$(1, 0, 0, 0)$ can be used to express a mixture of, say, fishy and spicy by the normalized linear combination $(1/2, 0, 1/2, 0)$. Since any weighted linear combination of unary base vectors with coefficients summing up to one remain inside the spanned unit simplex plane, these interpolating vectors can be used to express certain feature combinations. If such a vector is driven by many small steps into the direction of a particular feature, the according value will increase, and accordingly, the overall distribution of values will reflect a temporally smoothed feature histogram; thinking of the cube again, for which all six frequency bins would be equal, this kind of frequency representation is most appropriate for characterizing the average of a number of disjoint states.

Unary vectors are the best representation for clearly separated states, but they induce high-dimensional vectors, if there are many states. For example, 26-dimensional vectors would be required in case of the letter alphabet, and a size of 26^3 =17,576-dimensional vectors for independent letter triplets, which is infeasible in practice. A lossless reduction can be obtained for the Euclidean metric by rotating the simplex hyperplane into the subspace for which the dimension is decremented by one. Regarding a 4-letter DNA alphabet, the unary vectors become corner points of a tetrahedron making possible the visualization of a representation space in 3D; at the same time, the computational costs

are reduced by 25%. For high-dimensional data, the amount of reduction by such a hyperplane rotation becomes pointless. However, the central idea can be reused to find a rotational alignment along those data axes that maximize the discriminative properties. This is related to finding the eigenvectors with the k largest eigenvalues of the unary represented data covariance matrix. The principal component analysis (PCA) explained below or, alternatively, the more robust singular value decomposition (SVD) can be used for this purpose to project the data onto its k linearly most relevant dimensions [26]. If everything fails, just a random matrix can be used to map input vectors into a linear subspace of moderate dimension to obtain approximately disjoint non-overlapping states, as investigated by Kaski [77].

Another useful conversion refers to the angular representation of periodic data for which typically a relationship $f(\phi) = f(\phi + 2 \cdot \pi)$ is assumed: the same function values are produced despite very different radian angle arguments. In the case that only the compact angle representation ϕ of $f(\phi)$ is stored, a straight forward replacement of these angular data are the two-dimensional coordinates $\phi \mapsto (\cos(\phi), \sin(\phi))$, expressing the equivalence $\phi \equiv (\phi + 2 \cdot \pi)$ for all ϕ.

Cleaning

Especially for sequential data, interpolation schemes are useful cleaning operations: splines can provide the filling of data gaps, and Bezier curves can be used for smoothing [120]. More general smoothing operations can be obtained by a wide class of filters, for example the moving average or the Savitzky-Golay filters [131], but their choice is very specific to the data, and they should rather be seen as a transformation step discussed in the next section. A scenario related to filling is resampling which applies, for example, to turning input sequences of different lengths into vectors of equal dimension; the available sample points define interpolation functions that can be evaluated at concerted possibly equidistant positions. Outlier detection might not be feasible beforehand, but inconsistencies in labeled data, such as ambiguous class assignments, should be cleared.

Transformation

Data transformation is a crucial preprocessing step, aiming at normalization and base changes. The simplest normalization step is the *mean subtraction* $\tilde{\mathbf{x}} = \mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}}$ which moves the data set $\mathbf{X}$ into the origin. A widely used but not very robust normalization is the *lower–upper* bound normalization $\tilde{\mathbf{x}} = \boldsymbol{l} + \boldsymbol{u} \cdot (\mathbf{x} - \mathbf{min}_{\mathbf{X}})/(\mathbf{max}_{\mathbf{X}} - \mathbf{min}_{\mathbf{X}})$, with scalar operations $\cdot$ and / on the vector components. This forces data into the hypercube $[\boldsymbol{l}; \boldsymbol{u}]$ with lower bounds $l_j \in \boldsymbol{l}$ and upper bounds $u_j \in \boldsymbol{u}$. Such an operation is heavily suffering from the influence of data outliers onto the extreme values $\mathbf{min}_{\mathbf{X}}$ and $\mathbf{max}_{\mathbf{X}}$.

Much more recommended is the *z-score* transformation which defines a rescaling of the components $\tilde{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{X}})/\boldsymbol{\sigma}_{\mathbf{X}}$ by the inverse standard deviations $\sigma_j \in \boldsymbol{\sigma}_{\mathbf{X}}$. This operation prepares data to have zero mean and unit variance in all dimensions; it is an adequate processing for the use of rotation invariant metrics like the Euclidean distance, and many algorithms with this metric profit from such kind of standardization. The z-score transformation can be generalized to *whitening* also known as *sphering*. This operation addresses a cross-dimension standardization by linearly transforming a mean subtracted input $\mathbf{x}$ to $\tilde{\mathbf{x}} = \mathbf{C}\,\mathbf{x}$, for which the expected covariance matrix over all transformed input vectors $\mathbf{x} \in \mathbf{X} \mapsto \tilde{\mathbf{x}}$ becomes unity: $E(\tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^{\mathsf{T}}) = \mathbf{I}$. The matrix $\mathbf{C}$ can be easily determined by a principal component analysis (PCA) on $\mathbf{X}$ by decomposing the original expected covariance matrix $E(\mathbf{x}\,\mathbf{x}^{\mathsf{T}}) = \mathbf{E}\,\mathbf{D}\,\mathbf{E}^{\mathsf{T}}$, and by using its obtained orthogonal matrix of eigenvectors $\mathbf{E}$ and its diagonal matrix of eigenvalues $\mathbf{D}$ to express $\mathbf{C} = \mathbf{E}\,\mathbf{D}^{-1/2}\,\mathbf{E}$, where $\mathbf{D}^{-1/2} = (1/\sqrt{d_{ii}})$. As a result, the transformed data $\tilde{\mathbf{X}}$ is rotated and aligned to the axes providing maximum linear independence with variances scaled to unity. However, for multi-modal data such a rotation may lead to an unfavorable rearrangement of the point clusters. For data attributes with different semantics, for example one being a chemical concentration and the other the temperature, such a rotation with inherent dimension mixing seems counter-intuitive. Additionally, data labels are not taken into account. However, PCA has proved to be well-suited for unlabeled data with attributes representing the same kind of observations such as time window vectors. In applications like sound and image compression, this PCA preprocessing is very common and successful. A complementing technique that assumes non-Gaussian and also unlabeled sources is the independent component analysis (ICA) which computes an affine rotation and shearing transformation of the original data for getting an alignment along the axes for maximum non-Gaussian target distributions [73]. The ICA method exceeds the scope of simple preprocessing though.

The purpose of data transformation is to obtain a better compatibility with the target algorithm. Since the central ingredient to the design of self-organizing networks is the choice of a suitable data *metric*, any invertible functional transform — not only a linear function like the z-score — is allowed that helps to make the data fit the metric. Natural observations, including medical and physiological data, often exhibit exponential and power law relationships; therefore, the logarithmic or root function can be applied to such an attribute to deskew the corresponding unfavorable data distribution for further clustering. These inversely transformed data can be much more appropriately handled by standard metrics like the Euclidean distance. This is illustrated in Figure 3.1 for the two features `TSH` and `FTI` of the hypothyroid benchmark data set which is publicly available from the UCI repository [11]: data clusters become more pronounced and separable after transformation.

Especially for temporal sequences, derivatives can be useful: for example, the two point *differencing* $x'_t = x_t - x_{t-1}$ approximates the first derivative that purges linear trends. Two things must be kept in mind when derivatives are used: ⟨1⟩ the physical units change, for

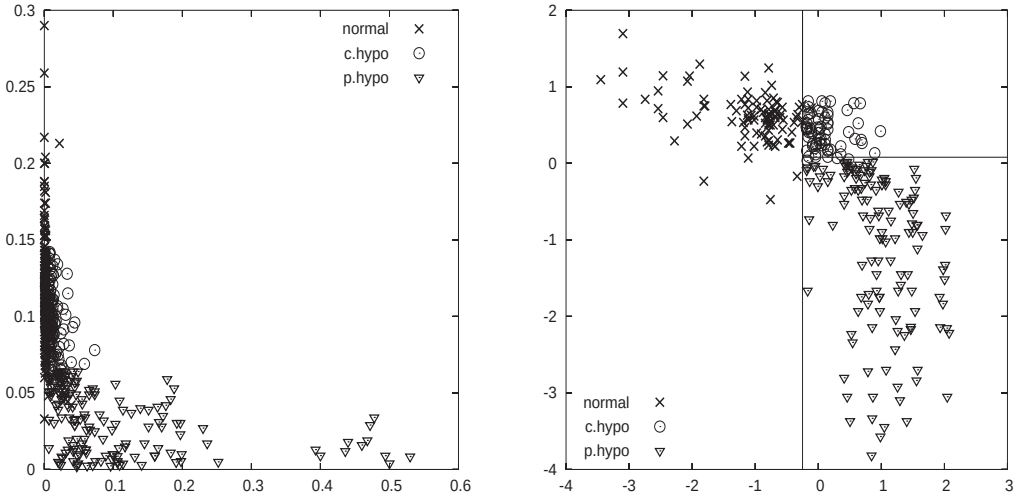


Figure 3.1: Data preparation. Left: scatter plot of original data, hypothyroid features `TSH` versus `FTI`. Right: same data after a z-score subsequent to a logarithmic transform.

example from displacement to velocity, and ⟨2⟩ noisy fluctuations in the original data are magnified. More complex sequence transformations can be obtained by changing the representation base, one example is a Fourier transform that turns data streams or data windows from the time domain into the frequency domain. If data windows are chosen sufficiently large, the Fourier transform is a first step towards a shift invariant representation, because only the presence of a certain period matters and not its location. Other popular bases are the space of spline and Bezier coefficients or the Wavelet coefficient matrix. Mathematical backgrounds, implementation details, application areas, and drawbacks and merits of alternatively chosen bases acting on behalf of the original data are, for example, briefly described in the Numerical Recipes [120].

Discretization

Two main reasons for the discretization of data can be identified before processing takes place. The first one is a histogram density representation. This representation can be conveniently used in computer implementations for frequency-based information measures such as the Shannon entropy or the Kullback-Leibler divergence [31, 98]. The second reason for discretization is the generation of labels that can be used for supervised classification and association learning.

Label generation is a mapping from a real-value attribute of an unlabeled data set into a specified number of integer outputs. Two canonic partitioning strategies are ⟨1⟩ the equal-interval (equiwidth) binning, splitting the whole range of numbers into intervals with equal size, and ⟨2⟩ the equal-frequency (equidepth) binning, using intervals containing equal number of values. A more meaningful partitioning of continuous features by means of a self-organizing data model has been suggested by Vannucci and Colla [156].

If a supervised discretization of the data is wanted, the values of the class variable are used to find appropriate break points; this task is related to the construction of decision trees [17, 121], and in its simplest iterative realization all possible splits are calculated and the one selected that provides minimum information with respect to explaining the class label.

Further aspects of data preprocessing

Two considerations are added related to badly separable data and to data from dynamic sources. The first topic refers to small and noisy data sets for which the training might be boosted by regularization, the second issue refers to the difficult task of online processing in contrast to the offline training scenario.

Regularization Usually, noise is considered to have a negative influence on learning. While this is true for intrinsic noise, it has been proved beneficial, if noise is externally added to the data. This noise injection method, known as *jittering*, produces more instances of the available data. If an appropriate jittering amount in terms of Gaussian noise is applied, network training produces better generalization abilities in many cases. This simple *regularization* ensures that the function computed by the network is no more curved than necessary. In the domain of feed-forward networks, equivalent techniques which change the cost function and which require specific implementations are the weight decay and the ridge regression, the latter also known as Tikhonov regularization [10]. Jittering should be considered, if there are only few data, or if the quality of the learned data abstraction is poor.

Online and offline processing Different stages of data preparation have been discussed above. The appropriate choice of methods depends on aspects regarding the desired data types, and on whether physical units should be maintained, or whether operations are reversible. Most techniques require a global view on the data, thus making an *offline* data preparation necessary. In case of dynamically observed or generated sequences, *online* preprocessing is preferred [165]. In an online scenario with possibly instationary series, for example attribute jittering is still easy, however, mean subtraction, albeit simple in the offline case, would demand a temporally adapted estimate of the average value to be subtracted; PCA as a more sophisticated preprocessing can only be realized offline when the original matrix algebra is used. Therefore, smart online reformulations are required, such as Sanger's rule for PCA [128], but these online formulations are usually not available for other methods.

Part II

Supervised LVQ-type learning

Chapter 4

Learning Vector Quantization (LVQ)

A teacher is one who makes himself progressively unnecessary.

Thomas Carruthers

The learning vector quantization algorithm (LVQ) proposed by Kohonen [81, 83] is a supervised classifier designed to find proper data representatives in the input space. In the presence of data labels, a good prototype location must make a compromise of $\langle a \rangle$ pointing to a possibly local center of gravity of the data belonging to the prototype's class and of $\langle b \rangle$ well-separating data of the prototype's class from those with different labels.

Typical applications of vector quantization are classification and data compression. Both features have been successfully realized by means of LVQ — or an extension of it — for image compression and image analysis, optical character recognition, speech analysis and recognition, signal waveform classifications, and robotics sensory implementations [82]. Kohonen points out that the main purpose of LVQ is statistical classification or pattern recognition. Specific examples of successful applications are EEG classification [40, 118] and surgery [22] in medical diagnostics, molecular sequence classification in bio-informatics [171], monitoring of piston engines in industrial production [13], and remote sensing image analysis in geoscience [163].

4.1 Vector quantization

Vector quantization is not restricted to labeled data, and a short introduction to the unsupervised case, to be detailed in Section 6, is helpful before the discussion of supervised learning is started. The essential idea of vector quantization is that a distortion function

$$\mathrm{E} = \sum_{\boldsymbol{w} \in \mathsf{W}} \int_{\mathbf{x} \in \mathsf{X}} \chi_{\boldsymbol{w}}(\mathbf{x}) \cdot \mathrm{d}(\mathbf{x}, \boldsymbol{w}) \cdot p(\mathbf{x}) \, d\mathbf{x}$$

is *minimized* for the winning reference vector $\boldsymbol{w}$. Winner selection is indicated by the value $\chi_{\boldsymbol{w}}(\mathbf{x}) = 1$ of the characteristic function if the prototype $\boldsymbol{w}$ is at minimum distance $\mathrm{d}(\mathbf{x}, \boldsymbol{w})$,

else $\chi_{\boldsymbol{w}}(\mathbf{x}) = 0$. This way, a large data set $\mathbf{X}$ with density $p(\mathbf{X})$ is represented by a comparatively small number of prototypes, and a continuum of inputs can be *quantized* to a finite number of characteristic states. These states may be optimum with respect to a density estimation of the data, or with respect to an optimum representation of data clusters, or with respect to any other criterion expressible by the involved metric $\mathrm{d}(\cdot)$. For example, the squared Euclidean distance $\mathrm{d}(\mathbf{x}, \boldsymbol{w}) = (\mathbf{x} - \boldsymbol{w})^2$ leads to a central clustering with high spatial correspondence of the data and their prototypes for Gaussian distributed data [10, 19].

An early and still widely spread quantizer is the Linde-Buzo-Gray (LBG) algorithm which minimizes a squared-error distortion measure by successive center of gravity calculations for the prototype locations. Thereby, both a nearest neighbor condition and a centroid condition are simultaneously taken into account, and the number of prototypes is iteratively increased according to a threshold-based splitting criterion [95]. If the number of target clusters is fixed to k, LBG yields the well-known k-means method [174]. A fuzzy variant of this, known as fuzzy k-means, provides a soft assignment of classes to the data points subject to membership constraints; for an overview see Cherkassky [24].

In contrast to the above offline processing of all training data, online methods adapt prototypes after the presentations of single training patterns. Especially, the neural gas quantizer (NG) of Martinetz, Berkovich, and Schulten [99] will become important in subsequent chapters of this work. Competitive prototype interactions make the NG quantizer robust against prototype initialization and lead to a good convergence.

Neural Gas (NG)

Prototype neighborhood competition is the core of the neural gas method: all prototypes are dragged towards presented data points by an amount corresponding to their distance rank, thereby exhibiting a potential-driven Brownian like motion resembling a gas particle model [99]. The prototype rank describes the number of neurons j which are closer to the presented pattern $\mathbf{x}$ than neuron i is: $\mathrm{rnk}_{\mathbf{x}}(i) = |\{j : d(\mathbf{x}, \boldsymbol{w}^j) < d(\mathbf{x}, \boldsymbol{w}^i)\}|$. This ranking establishes a competition between prototypes, because the amount of moving them towards $\mathbf{x}$ is multiplicatively weighted by the exponential function $f = \exp(-\mathrm{rnk}/\sigma)$. By decreasing the neighborhood size σ during training, competition vanishes and the updated prototypes become more specialized on their locally represented regions in the data space. The final dynamic turns out to follow a stochastic gradient descent on the introduced distortion function E with the extended indicator function f instead of χ for the squared Euclidean distance d. As a result, a data density approximation $p(\mathbf{X}) = p(\mathbf{W})^{\nu}$ subject to the magnification factor of $\nu = d/(d+2)$ is achieved by the adapted prototypes, where d is the effective data dimension. Optionally, a near optimum value of $\nu \approx 1$ can be obtained by including magnification control terms in localized prototype adaptations, in the winner determination, or in the cost function E, which has been thoroughly investigated by Villmann and Claussen [160].

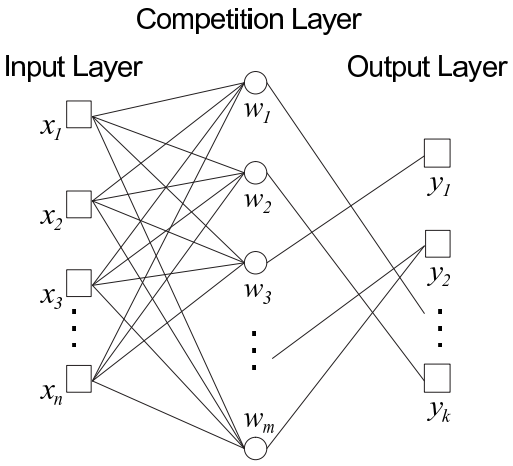


Figure 4.1: LVQ in forward network notation.

An alternative to the NG ranking operation is obtained by arranging prototypes as neurons in a fixed low-dimensional grid and by realizing the current prototype update strength as a function of the grid distance from the winning neuron. Such a grid neighborhood competition establishes the self-organizing map (SOM) that will be discussed in detail in Section 6 on page 68 [80]. At the expense of quantization accuracy, the SOM dynamic can be efficiently computed and at the same time be used for the projection of high-dimensional data onto specialized neurons located in a low-dimensional grid.

Supervised learning

In the supervised case of class prediction for a given data point, the k-nearest neighbor algorithm (k-NN) proposed by Cover and Hart [30] is one of the earliest and still frequently used classifier. It is a lazy learner which memorizes all samples in the training set and which predicts the class of unknown data by a soft majority vote weighted by the distances of the k nearest stored points.

Kohonen's learning vector quantization (LVQ) reduces the high memory requirement of k-NN: LVQ is a sparse classifier with a small number of prototypes; thus, LVQ needs significantly less comparisons for operation than k-NN, but it still leads to crisp classification results of high generality [83]. An LVQ net is determined by the prototype neurons and a distance measure on the input data, which is quite different from the paradigmatic feed-forward neural net architecture where input data are passed through error-reducing adaptive connections between neuron layers to an output layer.

Still, LVQ can be visualized, like in Figure 4.1, as input layer, intermediate layer, and output layer by utilizing the concept of an intermediate competition layer which carries out a global evaluation of all prototypes for the determination of a unique winner neuron and its class label. The LVQ dynamic that aims at improving the classification accuracy is easily obtained by paying tribute to the introduced distortion function E. For realizing

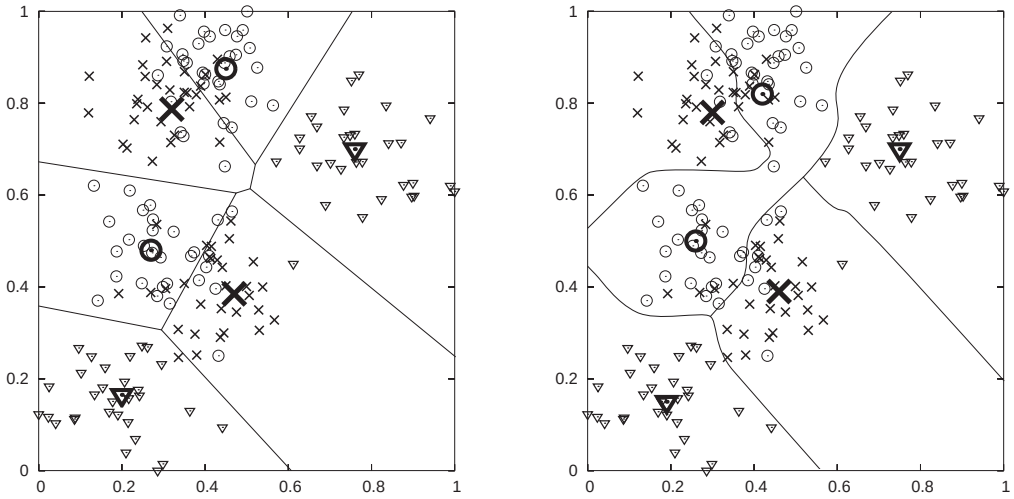


Figure 4.2: Voronoï cells for metric $\sum(x-y)^b$. Left: $b = 2$ (sq. Euclidean). Right: $b = 10$.

supervised prototype positioning with labeled data, an extended indicator function χ can be used, for example, by choosing $\chi = 1$ for the closest prototype representing the same label as the given pattern, $\chi = -1$ for the closest prototype with a different label, and $\chi = 0$ else. With this choice, an intuitive Hebbian distortion minimization strategy is locally given by moving the closest correct prototype a little towards the presented pattern and by moving away the closest wrong prototype.

The class membership of an unknown data point is obtained by assigning the class of the closest best-matching prototype which captures the input pattern inside its region of responsibility. The corresponding boundaries are defined by the relative positions of the prototypes within their neighborhood structure. Learning success is based on the assumption that minimizing the average distance of labeled prototypes to data points is related to maximizing the classification accuracy. However, for strongly asymmetric or discretized data this relation *distortion minimization* $\Rightarrow$ *classification improvement* does no longer hold, unless a metric more appropriate than the Euclidean is chosen.

With respect to the relationship of the chosen metric and the decision boundary, the notion of a prototype's *receptive field*, also called Voronoï cell, is important. Such a field is the subset of the input space

$$R_i = \{\mathbf{x} \in \mathcal{X} \mid \forall \boldsymbol{w}^j \, (j \neq i \rightarrow d(\mathbf{x}, \boldsymbol{w}^i) \leq d(\mathbf{x}, \boldsymbol{w}^j)) \}$$

for which prototype $\boldsymbol{w}^i$ is closer to the corresponding data points than any other prototype. This is illustrated in Figure 4.2 for a synthetic two-dimensional data set, 'closeness' being measured in two ways, $\langle 1 \rangle$ with the standard squared Euclidean distance which is invariant to data rotation, and $\langle 2 \rangle$ with a metric that focuses on the data axes. Prototypes, two for each of the three classes, are displayed by larger symbols; the lines define the decision boundaries of the receptive fields. For LVQ, a presented input pattern falling into the Voronoï cell of neuron $\boldsymbol{w}^i$ makes it the best-matching unit (bmu), the winner neuron.

This way, any vector in the input space can be quantized to a discrete state associated with the receptive field of the winning prototype vector. A key advantage of LVQ is that after training, its final prototypes can be easily interpreted, because they live in the same space as the processed data: prominent modes of the data are characterized by only a few reference vectors. According to Kohonen [82], the decision borders for a two-class LVQ problem approximate the Bayesian optimal decision border. This has been proved for the assumption of Gaussian distributed data and the squared Euclidean distance, which leads to piecewise linear class-separating planes along the receptive fields boundaries, as shown in the left panel of Figure 4.2.

4.2 The basic LVQ algorithm

The original LVQ dynamic is given in Algorithm 1. Hebbian online learning is achieved during iterative pattern presentations by local updates of the winner neurons. Winner prototypes with correct labels are rewarded by making them more similar to the current input, and wrong prototypes are penalized by pushing them away into opposite direction of the pattern. As a result, LVQ locates the real-valued prototype vectors in the input space by interpolating local centers of gravity of data points with common label, thereby accomplishing active separation from other classes.

Algorithm 1 Basic LVQ

repeat
  choose randomly a pattern $\mathbf{x}$
  $k \leftarrow \arg\min_i \{ d(\mathbf{x}, \boldsymbol{w}^i) \}$ { neuron with smallest distance d to pattern }
  if class($\boldsymbol{w}^k$) = class($\mathbf{x}$) **then**
    $\boldsymbol{w}^k \leftarrow \boldsymbol{w}^k + \gamma \cdot (\mathbf{x} - \boldsymbol{w}^k)$ { same class: drag towards pattern }
  else
    $\boldsymbol{w}^k \leftarrow \boldsymbol{w}^k - \gamma \cdot (\mathbf{x} - \boldsymbol{w}^k)$ { different class: push away from pattern }
  end if
until no more major changes

LVQ initialization

One problem of the standard LVQ algorithm is its dependence on a good prototype initialization. The first problem is the choice of the number of prototypes per class. Kohonen states that the optimum number is strongly depending on the data distribution, making iterative trials necessary [82]: too few prototypes won't capture all important data clus-

ters and too many prototypes not only lead to long processing times but also to poor generalization, because too detailed characteristics learned from the training set cannot be transferred to new data. Secondly, the initial placement of the prototypes contributes much to the success of LVQ. It turns out to be an unfavorable strategy to use data vectors as initial prototypes, because some data clusters might be accidentally overlooked while outliers are boosted, this way accounting for suboptimum training results. Kohonen proposes two strategies for choosing the starting locations: $\langle 1 \rangle$ by selecting a subset of reference vectors resulting from a previously trained k-nearest neighbor classifier, vectors for which even a large k-neighborhood represents the same class, or $\langle 2 \rangle$ by initialization with prototypes from the unsupervised SOM discussed in Chapter 6, for which the neurons are turned by majority vote into labeled LVQ prototypes [82]. These two initialization strategies themselves require a number of well-chosen parameters, such as their own initial prototype positions and considerations about the number k for k-NN, or the appropriate architecture for the SOM. Alternatively, the prototypes from the robust neural gas quantizer introduced above can be labeled after convergence and then be fine-tuned by LVQ [20]. In a later chapter, this idea will lead to an integrated model of LVQ and NG.

4.3 LVQ evaluation

Training is the essential step for obtaining a good classifier; therefore, the monitoring of the training progress is desired to identify critical states of learning, and to see, whether classification is developing promisingly. Also, after the training, the final model must be evaluated.

Quantization error

For a single training run, the learning process can be easily tracked by observing the distortion function E. In order to get independence from the data set size, errors should be normalized. A commonly used quantization error describing the average squared Euclidean distance of the data from the closest correct prototypes selected by χ is calculated by

$$e = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x} \in \mathbf{X}} \sum_{\boldsymbol{w} \in \mathbf{W}} \chi_{\boldsymbol{w}}(\mathbf{x}) \cdot \frac{1}{d} \sum_{i=1}^{d} (x_i - w_i)^2 \ .$$

Without modification, this measure can be used for unlabeled data too. However, in the supervised case of LVQ, more appropriate evaluation criteria can be found by taking label misclassification into account.

Generalization error

A very prominent and intuitive measure is the computation of the classification error on the training set and on the test set. The training error can be easily tracked during the online learning process just by counting the number of winning prototypes for a class

different from the presented pattern. A division of this count by the total number of neurons and presentations yields the average training misclassification rate per neuron. Opposed to that, the test set error provides an estimate of how accurate the classification will be for unknown data. Small differences between the training error and the test error indicate both that $\langle a \rangle$ the training set and the test set contain representative patterns and that $\langle b \rangle$ the classifier provides a good generalization from the training data to the test data. These two hypotheses are supported, if only small differences occur also in several independent validation steps. Equivalently, the rate of correct classification, the accuracy, can be considered instead of the error. Figure 5.2 on page 46 gives an example of the improvement of the classification accuracy during learning for the considered training set and the test set.

Receiver operating characteristic curve (ROC)

Another widely used evaluation criterion for binary classifiers is the computation of the receiver operating characteristic (ROC) curve. It is a plot of the classifier's sensitivity given by the true positive rate against the false positive rate expressed by the (1–specificity)-value of the classifier, both for the different possible cutpoints of a diagnostic test. This plotting diagram has been originally developed in World War II for supporting signal detection tasks of radar operators [97]. With counts tp for the true positives, fp for the false positives, tn for the true negatives, and fn for the false negatives, the values of interest are

$$\text{specificity} = \frac{tp}{tp + fn} \qquad \text{and} \qquad \text{sensitivity} = \frac{tn}{tn + fp} \ .$$

High specificity means that the classifier identifies correctly most of the negative examples, thus providing a small false-negative rate. High sensitivity means that most of the potential positives are reported; however, possibly many negatives are also reported, making a good decision still difficult. A ROC curve demonstrates the tradeoff between sensitivity and specificity: $\langle 1 \rangle$ the closer the curve follows the sensitivity axis and then the (1–specificity)-axis in the ROC space, the more accurate the test, and $\langle 2 \rangle$ the closer the curve approaches the 45-degree reference diagonal in the ROC space, the less accurate the test. Equivalently, maximizing the area under the curve maximizes the test accuracy. An illustration of a ROC curve is given in Figure 5.14 on page 62. While for a binary classifier like support vector machines the ROC plot can be used as is, a multi-class task of LVQ must be reduced to the binary decision '$\mathbf{x}$ is of desired class' vs. '$\mathbf{x}$ is not of desired class', no matter how many other classes there are.

4.4 LVQ variants and developments

Two main lines of improvement for standard LVQ can be drawn, one pointing at the rate and stability of the convergence process, the other concerning the classification accuracy.

As a calculation speed-up, Kohonen has proposed an extension to the basic LVQ1 Algorithm 1 by implementing local learning rate factors γ_i optimal for each neuron. The optimum refers to the equalization of the impact of all training samples on the update and it gives the name OLVQ to this extension. As a classification improvement, Kohonen has suggested to use simultaneously the closest correct and the closet wrong prototype to a presented sample in order to get a better local Bayes decision boundary approximation between them. This variant, LVQ2.1, optimizes the boundary, but it suffers from the problem of potentially misplacing prototypes outside the data clusters. LVQ3, an extension of LVQ2.1, is Kohonen's answer that improves data density representation [82].

A variant with noncritical prototype initialization for avoiding local minima of the quantization error is the combination of LVQ with the SOM update: LVQ prototypes are attached to a low-dimensional neural grid, and during training, all prototypes within the lattice neighborhood of the winner are updated into the direction of the sample, if they represent the input's class, else they are pushed away. Due to its grid neighborhood, this LVQ-SOM yields a dimension reduction which is suitable for visualization rather than for accurate classification [82]. Very similarly, the low-dimensional SOM can be replaced by the unsupervised neural gas (NG) for a density-based LVQ prototype initialization [20]. As expected, being freed from the neuron target grid, the NG classification accuracy becomes much higher, but at the expense of the visualization. A very successful integration of unsupervised neural gas and supervised LVQ into the supervised relevance neural gas (SRNG) quantizer with neuron cooperation will be shortly presented in this work.

Flexible metrics

Metric adaptivity, to be explained in the following, accounts for both stabilizing the training and improving the classification. Often data are high-dimensional just because any available information is put into the set of training vectors. This must be done, if the features relevant for classification are unknown in advance. Otherwise, the prior knowledge should be integrated into the metric for distance calculation, for example in terms of factors weighting dimension relevances. Knowing the important data dimensions is useful for dimension reduction, for noise cancellation, for feature detection, for speed-up, and for the formulation of attribute-based classification rules. Since prior knowledge is hard to integrate into models and since it is seldom available, an automatic adjustment of the distance metric according to an optimization of the given task is desirable.

Cherkassky et al. [25] propose the batch constrained topological mapping (CTM) as an adaptive kernel method for unsupervised learning. Batch CTM implements local linear data regression in SOM units, thereby extracting dimension relevances heuristically from

the sensitivity of regressor variables. Opposed to this unsupervised method, Sinkkonen and Kaski use additional information connected with the data, such as class labels, to make both the data distribution and the distribution of prototypes in a generic auxiliary space as small as possible; for example, this can be achieved by minimizing the Kullback-Leibler divergence between these two distributions which both refer to conditional probabilities given the discrete data labels [79, 137]. Prototypes are thus adapted in the possibly low-dimensional auxiliary space to make their overall distribution, with respect to the data label, as close as possible to the one in the original data space. As a consequence, prototypes must be interpreted in the auxiliary space rather than in the data space. Another contribution to automatic feature weighting has been made by Wettschereck, Aha, and Mohri for lazy learning algorithms like the k-nearest neighbor (k-NN) classifier [169].

In combination with LVQ, metric-based relevance learning has been proposed first as distinctive sensitive relevance LVQ (DSLVQ) by Pregenzer [119]. In addition to the standard LVQ prototype update, a relevance vector $\boldsymbol{\lambda}$ belonging to an *adaptive metric* is given as the weighted Euclidean distance $\mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}) = \left(\sum_{i=1}^{d} \lambda_i \cdot (x_i - w_i)^2\right)^{1/2}$. DSLVQ relevance determination is heavily based on the orientation of vectors orthogonal to the linear classification boundaries, which makes a transfer to more general metrics difficult. Moreover, DSLVQ applies the original ad-hoc rotation of LVQ also to the relevance vector $\boldsymbol{\lambda}$ which gets directed orthogonal to the average classification boundaries, thereby the direction of rotation is chosen according to seeing a prototype with a correct or with a wrong label. Relevance LVQ (RLVQ) by Bojer, Hammer, and Toschanowitz [14] is a similar approach with heuristic update of the dimension weights. The generalized RLVQ (GRLVQ) and the SRNG quantizers introduced in subsequent chapters will tackle the relevance learning more systematically by means of a cost function minimization. Complementary to the distance distortion measure, the Gini index, known as informational energy, can be optimized during learning; in conceptual terms, this strategy is discussed by MacKay [98]. Recently, a specific adaptation criterion of the weighted Euclidean metric has been proposed for Energy Relevance LVQ (ERLVQ) by Andonie and Cataron [2].

Sequence data processing

By focusing on relevant data components, metric adaptation alleviates the effects of the curse of dimensionality for high-dimensional data. This feature allows to reconsider data windows of a fixed dimension for temporal sequence classification. Usually, data embedding into a possibly high-dimensional vector space combines two major drawbacks: $\langle 1 \rangle$ a good embedding dimension is in many cases unknown and thus subject to empirical choices, and $\langle 2 \rangle$ a vast redundancy is produced, because if the window shift is small, such as only one step, the previously generated vector is exactly the same as the current one, apart from one additional and one dropped component, but the rest of the vector is just shifted by a single element. Both objections are addressed by flexible metrics. Firstly, the initial dimension of data windows can be generously chosen and the attribute relevance detection can be left

to dynamic adaptation. Secondly, redundant information is reduced to a representation by only a few prototypes. Redundancy, however, is pretty much metric dependent. For example, the Euclidean distance is insensitive to order of the vector components due to the commutative summation of their individual squared differences. Sequence processing alternatives to the Euclidean metric should therefore take temporal or spatial ordering into consideration, in the simplest case, just by weighting the historic context influence for the comparison. This motivates the design of specialized metrics for a classification task, which will be another important issue of the SRNG quantizer discussed in the next part.

Chapter 5

LVQ with cost function

Learn to adjust yourself to the conditions you have to endure,
but make a point of trying to alter or correct conditions
so that they are most favorable to you.

William Frederick Book

It has been pointed out that the learning vector quantization method yields fast and compact classification models based on easily interpretable prototype representations. The present part contributes to the basic LVQ technology and extends it with respect to the

- initialization of the prototypes,
- stability of learning,
- improvement of the classification accuracy,
- design of specialized and flexible data metrics,
- processing of sequential data,
- identification of relevant data dimensions, and the
- extraction of rules from trained classifiers.

These issues are addressed by the central idea that the classification task is formulated in terms of an adaptive *cost function*. In the following, the prototype dynamic and the metric adaptation will be developed for differentiable cost functions by means of a stochastic gradient descent.

A systematic discussion of chances and challenges for a number of cost function-based network types has been brought forward by Hertz, Krogh, and Palmer [68]. In the case of unsupervised SOM learning, Graepel, Burger, and Obermayer formulate a target for the process of the prototype-based cluster self-organization, given by a discontinuous error

function that is minimized by probabilistic annealing [52]. Recently, this approach has been successfully transferred to LVQ networks by Seo and Obermayer who adapt the prototypes by maximizing the probability of a prototype-based Gaussian mixture model through gradient descent [136]. Historically, Sato and Yamada have proposed a first generalized extension to Kohonen's LVQ networks with gradient-driven prototype adaptation. Their GLVQ method makes explicit use of a cost function that measures deviations of prototypes from data. The free parameters of that cost function are the prototype locations which are adapted iteratively, driven by the objective to minimize the average distance to data with the same class labels [129]. The authors show that GLVQ yields superior results compared to LVQ3 and additionally provides a robust update dynamic [130]. In the following, the good results of Sato and Yamada are further improved by going far beyond their GLVQ proposal, without using the fixed assumption of Gaussian mixtures of Seo and Obermayer.

The cost function idea can be enhanced by expressing the costs in terms of specialized distances designed for matching particular data structures during comparisons. In addition to the prototype locations, more free parameters can be included in the cost function; for example, the individual data attributes can be weighted by adaptable factors in order to model the importance of each data dimension for the classification.

A first rough specification of the training procedure for an iterative minimization of the cost function is outlined in Algorithm 2. The following subsections provide an overview of the formal aspects of two recent LVQ variants GRLVQ [65] and SRNG [58, 164]; historically, both methods have been formulated for the squared Euclidean distance, but the naming will be used interchangeably in the case of more general metrics. A procedure based upon relevance factors and prototype locations will be introduced afterwards in order to obtain a BB-tree classification [57] with a possibly simplified set of rules.

Algorithm 2 Extended LVQ outline

Initialization.
repeat
  Present a labeled data point randomly chosen from the training set.
  Determine the closest correct and the closest wrong prototype.
    For the prototype locations on a fast time scale
      move the correct prototype towards the given point and
      move the wrong prototype away.
    For the metric weighting factors on a slow time scale
      adapt the factors
      possibly using normalization constraints.
until the maximum number of iterations exceeded.

5.1 Generalized Relevance LVQ (GRLVQ) and extensions

Given a set of training data $\mathbf{X} = \{(\mathbf{x}^i, y^i) \in \mathbb{R}^d \times \{1, \ldots, c\} \mid i = 1, \ldots, n\}$ to be classified with d-dimensional elements $\mathbf{x}^k = (x_1^k, \ldots, x_d^k)$ and c classes. A set $\mathbf{W} = \{\boldsymbol{w}^1, \ldots, \boldsymbol{w}^K\}$ of prototypes is used for the data representation, $\boldsymbol{w}^i = (w_1^i, \ldots, w_n^i, y^i) \in \mathbb{R}^d \times \{1, \ldots, c\}$, with class labels y^i attached to locations in the data space.

The cost function to be minimized for the classification is given in the general form

$$\mathrm{E}_{\mathsf{GRLVQ}} := \sum_{i=1}^{n} \mathrm{g}\left(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \quad \text{where} \quad \mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i) = \frac{\mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) - \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)}{\mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) + \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)}\,.$$

The purpose of $\mathrm{E}_{\mathsf{GRLVQ}}$ is to express the desired distance relationships between the presented pattern $\mathbf{x}^i$ and two closest prototypes, $\boldsymbol{w}^{i+}$ representing the same label and $\boldsymbol{w}^{i-}$ a different label. The implicit degrees of freedom for the cost minimization are thus the locations of the involved prototypes in the weight space and, additionally, a set of free parameters $\boldsymbol{\lambda}$ connected to the distance metrics $\mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}) = \mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w})$. Distances $\mathrm{d}_{\boldsymbol{\lambda}}^+$ and $\mathrm{d}_{\boldsymbol{\lambda}}^-$ for the correct and for the wrong prototype constitute the quotient $\mathrm{q}_{\boldsymbol{\lambda}} \in [-1; 1]$ which is small (negative), if the correct prototype is closer to the pattern than the wrong prototype, and large (positive) otherwise. In order to obtain a convergent dynamic, it must be assured that the resulting attractive forces towards the correct data are larger than the repulsive forces from falsely classified data. For this reason, Sato and Yamada have wrapped the ratio $\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)$ by a nonlinear decreasing function $x \in \mathbb{R} \mapsto \mathrm{g}(x) \in \mathbb{R}$; stable training has been proved for the sigmoid function $\mathrm{g}(x) = \mathrm{sgd}(x) = 1/(1 + \exp(-x)) \in (0; 1)$ in combination with the squared Euclidean distance [129, 130]. However, the convergence considerations still hold, if the identity function $\mathrm{g}(x) = \mathrm{id}(x)$ is chosen as wrapper.

The iterative update formulas for the closest correct and the closest wrong prototype and the metric weights are obtained by taking the derivatives of the cost function $\mathrm{E}_{\mathsf{GRLVQ}}$:

$$\triangle \boldsymbol{w}^{i+} = -\gamma^+ \cdot \frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{w}^{i+}} = -\gamma^+ \cdot \mathrm{g}'\left(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \frac{2 \cdot \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)}{\left(\mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) + \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)\right)^2} \cdot \frac{\partial \mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i)}{\partial \boldsymbol{w}^{i+}}$$

$$\triangle \boldsymbol{w}^{i-} = \gamma^- \cdot \frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{w}^{i-}} = \gamma^- \cdot \mathrm{g}'\left(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \frac{2 \cdot \mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i)}{\left(\mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) + \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)\right)^2} \cdot \frac{\partial \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)}{\partial \boldsymbol{w}^{i-}}$$

$$\triangle \boldsymbol{\lambda} = -\gamma^{\boldsymbol{\lambda}} \cdot \frac{\partial \mathrm{E}_{\mathsf{GRLVQ}}}{\partial \boldsymbol{\lambda}} = -\gamma^{\boldsymbol{\lambda}} \cdot \mathrm{g}'\left(\mathrm{q}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \frac{2 \cdot \partial \mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i)/\partial \boldsymbol{\lambda} \cdot \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i) - 2 \cdot \mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) \cdot \partial \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)/\partial \boldsymbol{\lambda}}{\left(\mathrm{d}_{\boldsymbol{\lambda}}^+(\mathbf{x}^i) + \mathrm{d}_{\boldsymbol{\lambda}}^-(\mathbf{x}^i)\right)^2}$$

Learning rates are $\gamma^{\boldsymbol{\lambda}}$ for the metric parameters λ_j, and γ^+ and γ^- describe the update amount of the correct and wrong prototype, respectively. Care must be taken when free parameters with different semantics are adapted simultaneously for error minimization: considering dimension relevances and prototype locations, the relevances should be the result of a classifier defined by the receptive fields of the settled prototypes, not vice versa. Therefore, time scales have to be defined that allow a fast prototype update in a slowly changing environment which is quasi-stationary with respect to the dimension relevances. Thus, learning rates should be chosen according to the relation $0 \leq \gamma^{\boldsymbol{\lambda}} \ll \gamma^- \leq \gamma^+ \leq 1$

which refers to ratios of usually some orders of magnitude. The choice of $\gamma^- \leq \gamma^+$ should be considered to prevent a prototype explosion, because the average dragging force on correct prototypes into the data clusters, controlled by γ^+, must be greater than the pushing force γ^- on wrong prototypes. The metric update step $\triangle\boldsymbol{\lambda}$ denotes the change of $\boldsymbol{\lambda}$ given pattern $\mathbf{x}^i$. Depending on the chosen metric, it might be advisable to restrict its free parameters $\boldsymbol{\lambda}$ to a certain domain, if the cost minimization leads to vanishing components $|\lambda_j| \to 0$ or to diverging components $|\lambda_j| \to \infty$.

For the gradient descent-based cost minimization it must be assured that the cost function is differentiable everywhere. For the considered cost function $\mathrm{E}_{\mathsf{GRLVQ}}$ it has been shown by Hammer et al. that the derivatives exist, including the classification boundaries [61, 65]. In the error minimization framework, the function $\mathrm{d}_{\boldsymbol{\lambda}}$ needs not be a strict mathematical metric, its differentiability is already a sufficient precondition. However, metrics are most natural for carrying out comparisons; for this reason, metrics will be considered in the following, although more general differentiable functions are valid as well.

Adaptive metrics

An early approach to data-specific distance calculations has been given by Gustafson and Kessel, who have extended the unsupervised fuzzy k-means algorithm by covariance matrices approximating the ellipsoidal radii of the k data clusters [53]. The computationally costly operations per iteration involve k local Mahalanobis distance and determinant calculations, adding up to minimum computation complexity of $\mathcal{O}(d^{2.69})$ for d-dimensional data. An extension has been proposed by Gath and Geva who improve the modeling of local cluster sizes and densities by a fuzzy Gaussian mixture decomposition of the data [47, 49].

The step away from the local distances to global metrics with adaptive parameters has been realized in several works, some of which outlined on page 25 are batch CTM [25], DSLVQ [119], and RLVQ [14]. These three methods share the common property that they heuristically adapt factors which weight the influence of dimensions on an Euclidean distance term. Sinkkonen and Kaski formulate a mathematically more rigorous approach by means of a function for the classification costs. By definition, this function takes its minimum for the best correspondence of the original data distribution and the distribution modeled with data prototypes, both related to auxiliary data labels [79].

Generally, the formulation of the classifier training in terms of cost function minimization permits the adaptation of any variable contributing to the costs, because a clear objective is given. Parameter adaptation can be based on information theoretic criteria, as done in Sinkkonen's and Kaski's stochastic expectation maximization framework. Alternatively, the parameter update dynamic can be driven by distortion criteria reflected by the cost function for the data and their prototypes. If this function is differentiable, optimum parameters for its minimization can be iteratively obtained with tools from calculus. Adaptive factors for weighting individual data attributes have been first introduced for a cost function variant of LVQ by Hammer and Villmann [65].

Recently, Hammer et al. have linked generalized relevance LVQ (GRLVQ) to the structural risk minimization framework of learning theory [60]. Following ideas of Crammer et al. [32], it has been shown that the GRLVQ algorithm with adaptive diagonal Euclidean metric restrained to $\sum_{i=1}^{d} |\lambda_i| = 1$ leads to *margin maximization*. Thereby, *margin* refers to the hyperplane separating two adjacent data clusters of different classes, and the term *maximization* refers to the amount of freedom for choosing the parameters, i.e. the freedom for placing the two prototypes that determine the plane without changing the classification results. Apart from the Euclidean distance, more generally, any kernelizable operation leads to margin maximization. This property makes GRLVQ competitive to the well-established support vector machine (SVM) classifier [27, 134]. Since SVM usually generates larger and computationally more demanding models of the data than GRLVQ [62], this promising LVQ variant will be further studied in the following, and extensions to more general metrics and to initialization-tolerant prototype cooperation will be considered.

Euclidean type metrics

The simplest metric structure to be plugged into the update equations is a generalized weighted Euclidean distance:

$$\begin{aligned} \mathrm{d}^{\mathsf{EUC}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i) &= \sum_{j=1}^{d} \lambda_j^{b_{\boldsymbol{\lambda}}} \cdot (x_j - w_j^i)^{b_{\boldsymbol{w}}}, \quad \text{integers } b_{\boldsymbol{\lambda}}, b_{\boldsymbol{w}} \geq 0\,,\ b_{\boldsymbol{w}} \text{ even} \\ \Rightarrow \quad \frac{\partial \mathrm{d}^{\mathsf{EUC}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i)}{\partial w_j^i} &= -b_{\boldsymbol{w}} \cdot \lambda_j^{b_{\boldsymbol{\lambda}}} \cdot (x_j - w_j^i)^{b_{\boldsymbol{w}}-1}, \\ \frac{\partial \mathrm{d}^{\mathsf{EUC}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i)}{\partial \lambda_j} &= b_{\boldsymbol{\lambda}} \cdot \lambda_j^{b_{\boldsymbol{\lambda}}-1} \cdot (x_j - w_j^i)^{b_{\boldsymbol{w}}}. \end{aligned}$$

The exponent $b_{\boldsymbol{w}}$ controls how much the mismatch in single dimensions contributes to the cost: large exponents lead to the emphasis of outlier dimensions, whereas small values better tolerate deviations that might be subject to noise. In simple words: large $b_{\boldsymbol{w}}$ focus on dimensions with large differences and, compared to this, small $b_{\boldsymbol{w}}$ focus on dimensions with small differences. In the squared case with $b_{\boldsymbol{w}} = 2$, the derivative for the prototype update $2 \cdot (x_j - w_j^i)$ is recognizable as Hebbian learning term.

The exponent $b_{\boldsymbol{\lambda}}$ takes influence on the adaptation dynamic of the factors λ_j; typical values for $b_{\boldsymbol{\lambda}}$ are 0,1,2, and 4. Disregarding the metric parameters $\lambda_i > 0$ by setting $b_{\boldsymbol{\lambda}} = 0$ and choosing $\mathrm{g}(x) = \mathrm{sgd}(x)$ yields the original GLVQ. If the metric parameters λ_j are taken into account for $b_{\boldsymbol{\lambda}} \neq 0$, they become factors that weight the influence of differences in certain data dimensions. For example, a noise dimension added to a classification task usually does not help to drive the prototypes into the clusters where they belong: since the position of a Voronoï decision boundary between two classes depends on every dimension of the involved two prototypes, noisy components induce an orientation of the borderline different from the one for omitted noise dimensions; this noise-affected borderline may lead to the accidental inclusion of data points with wrong labels.

Throughout this work, metrics are used for which their variables λ_j refer to the weighting of dimension j, although a more general parameter utilization is also possible. The *forced normalization* to $\sum_{j=1}^{d} |\lambda_j| = 1$ is necessary, because the dynamic can drive diverging relevance factors to extreme values. For an odd exponent, such as $b_{\boldsymbol{\lambda}} = 1$, an additional clipping of the λ_j to non-negative values must implemented, because the corresponding derivative of constant 1, scaled by the metric learning rate, can decrease the λ_j to undesired negative values. This clipped normalization has been performed during each step of the training update; thus, a comparison of the different relevance profiles is possible after training. Putting together normalized adaptive metric parameters for $b_{\boldsymbol{\lambda}} = 1$ and GLVQ yields generalized relevance LVQ, GRLVQ for short [65].

The choice of the exponents $b_{\boldsymbol{w}}$ and $b_{\boldsymbol{\lambda}}$ remains to be discussed. Usually, the quadratic distortion measure d^2 with $b_{\boldsymbol{w}} = 2$, $b_{\boldsymbol{\lambda}} = 0$ is applied which yields invariance to data rotations and which provides risk minimization for false classification of Gaussian data. For data attributes with incompatible semantics, the rotation invariance might be unwanted; instead, the intra-dimensional matching should outperform the inter-dimensional fitting. Additionally, it is known that Gaussian data can be described by the first two central moments, the mean and the variance; for non-Gaussian data, though, moments of higher order, expressed by larger metric exponents, are more appropriate for capturing data statistics such as sharp cluster boundaries [9, 105].

As a matter of fact, in many experiments the quartic distance $\mathrm{d}^4_{\boldsymbol{\lambda}^2}$ with $b_{\boldsymbol{w}} = 4$, $b_{\boldsymbol{\lambda}} = 2$ has turned out to be a reliable metric, leading faster to classification results better than the d^2 above and also better than its adaptive counterpart $\mathrm{d}^2_{\boldsymbol{\lambda}^2}$ with $b_{\boldsymbol{w}} = 2$, $b_{\boldsymbol{\lambda}} = 2$. Its derivative with respect to the prototype locations displays a Hebbian term, taken to the power of three; thus, prototypes with badly matching dimensions are nonlinearly dragged towards the centroids of the according attribute, which yields an alignment along the data axes, in contrast to the rotation invariant standard Euclidean metric. Empirically, a choice of larger exponents $b_{\boldsymbol{w}}$ and $b_{\boldsymbol{\lambda}}$ has not further improved the results, but training tends to get numerically unstable. Recent experiments show that using $b_{\boldsymbol{\lambda}} = 1$ with clipping $\lambda_j \geq 0$ leads to better results than $b_{\boldsymbol{\lambda}} > 1$ with built-in self-inhibition: higher exponents tend to boost the most relevant dimensions at the expense of less important, yet contributing attributes; this effect will be illustrated for a 10-dimensional artificial data set.

By means of adaptive metric parameters adjusted to minimize the cost function, the importance of data dimensions for the classification task are determined. After the training, $\boldsymbol{\lambda}$ reflects a dimension relevance profile. This can be used to identify sufficiently large λ_j indicating attributes onto which the data set might be projected for further classification, analysis, or for visualization. Apart from this *dimensionality reduction*, another important benefit of the relevance learning is the *generalization improvement* of the networks: since noise dimensions are rated less important in the training data, false influences will be canceled also during the classification of new data. This way, a direct connection is established between network pruning, relevance learning, and Tikhonov regularization.

One remark about the squared error sheds light on a more general aspect of the design of distance metrics. The term expansion $(x_j - w_j)^2 = x_j^2 - 2 \cdot x_j \cdot w_j + w_j^2$ can be interpreted as follows: x_j^2 is a pattern-specific invariant that contributes unavoidable constant costs, and the product $x_j \cdot w_j$ measures the linear covariance between the vectors $\mathbf{x}$ and $\boldsymbol{w}$ subject to the regularization term w_j^2. The adjustment of w_j must thus find a compromise between possibly small values and twice the maximum negative 'correlation' to the patterns that the prototype is responsible for.

For diagonal metrics with separating dimensions $\mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}) = \sum_{k=1}^{d} \mathrm{v}(\lambda_k) \cdot f_k$, with $\mathrm{v}(\lambda_k) = \lambda_k^2$ and $f_k = f(x_k, w_k) = (x_k - w_k)^2$ for the squared weighted Euclidean distance, the GRLVQ relevance adaptation is briefly studied. Only the numerator of the relevance adaptation term $\triangle\boldsymbol{\lambda}$ is considered for a single dimension i with respect to the direction of change; as notational convention the superscripts $+$ and $-$ are used which indicate correspondence for the closest correct and the closest wrong prototype, respectively, to a given pattern. The term of interest is transformed:

$$\triangle\lambda_i \;\propto\; \mathrm{d}_{\boldsymbol{\lambda}}^{+} \cdot \frac{\partial}{\partial\lambda_i} \sum_{k=1}^{d} \mathrm{v}(\lambda_k) \cdot f_k^{-} \;-\; \mathrm{d}_{\boldsymbol{\lambda}}^{-} \cdot \frac{\partial}{\partial\lambda_i} \sum_{k=1}^{d} \mathrm{v}(\lambda_k) \cdot f_k^{+}$$

$$\Rightarrow\; \triangle\lambda_i \;\propto\; \left(\mathrm{d}_{\boldsymbol{\lambda}}^{+} \cdot f_i^{-} \;-\; \mathrm{d}_{\boldsymbol{\lambda}}^{-} \cdot f_i^{+}\right) \cdot \frac{\partial\, \mathrm{v}(\lambda_i)}{\partial\lambda_i} \;;\; \text{assuming } \frac{\partial\, \mathrm{v}(\lambda_i)}{\partial\lambda_i} > 0$$

$$\Rightarrow\; \triangle\lambda_i \gtreqless 0 \quad\Leftrightarrow\quad \frac{f_i^{-}}{f_i^{+}} \gtreqless \frac{\mathrm{d}_{\boldsymbol{\lambda}}^{-}}{\mathrm{d}_{\boldsymbol{\lambda}}^{+}} \quad .$$

By the example of the relevance augmentation of dimension i, that is $\triangle\lambda_i > 0$, the effect of the involved ratios is illustrated, assuming positive distance values and positive derivatives that indicate the demand for further adaptation. If only the ratio $\frac{f_i^{-}}{f_i^{+}}$ was maximized, this would mean that in the projections, the correct prototype should be as close as possible and the wrong prototype should be far away; this feature is desired for class separation. The dimension relevance is thus increased, if the contribution expressed by that ratio still supersedes the global distance ratio that takes the current dimension scaling into account. The relevance is decreased otherwise. Relevances are not updated in the equilibrium, when the average dimension-specific distance ratios are the same as for the global weighted distances. In contrast to RLVQ, relevance adaptation takes place independent of an explicit distinction of $\mathrm{d}_{\boldsymbol{\lambda}}^{+} \lessgtr \mathrm{d}_{\boldsymbol{\lambda}}^{-}$ for correct or wrong classification. Instead, the Hebbian paradigm is implicitly realized for relevance learning by distance competition.

A further question related to the relevance adaptation is whether a wrapper function $\mathrm{v}(\lambda_i)$ can be found that yields an implicit normalization without degeneration to extreme values. Although theoretic investigations are still missing, experiments that avoid forced normalization show good results for $\mathrm{v}(\lambda_i) = \mathrm{sgd}(\lambda_i)$. However, since this sigmoidal wrapping suffers from slow convergence, large data sets have been trained as discussed above with small integer exponents for the relevance factors, and with normalization.

Related to the weighted Euclidean distance, the more general Mahalanobis metric is computing $\mathrm{d}^{\mathsf{MAH}}_{\mathbf{C_x}}(\mathbf{x}, \boldsymbol{w}) = (\mathbf{x} - \boldsymbol{w})^{\mathsf{T}}\, \mathbf{C}_{\mathbf{x}}^{-1}\, (\mathbf{x} - \boldsymbol{w})$ with a covariance matrix $\mathbf{C_x}$ of $\mathbf{x}$ that compensates for the cross-dimension correlations. The case of uncorrelated dimensions, given by $\mathbf{C_x} := \mathbf{I}\boldsymbol{\lambda}$, results in the GRLVQ metric with the squared Euclidean and the λ_j exponent $b_{\boldsymbol{\lambda}} = 1$. Recently, an application of this distance to the k-means clustering method has been given by Xing et al. [173]. In the present work, the full matrix has not been used for three reasons: $\langle 1 \rangle$ its computational time complexity is $\mathcal{O}(d^2)$ instead of $\mathcal{O}(d)$ for each pattern; $\langle 2 \rangle$ data preprocessing, such as principal component analysis PCA, could be used beforehand to scale and rotate the input data in order to minimize the correlations between the dimensions; and $\langle 3 \rangle$ it is unclear how the positive definiteness of the distance matrix can be obtained as a result of the parameter update dynamic.

Unidirectional metric with exponential fading (SEQ)

A metric taking into consideration the nature of directed sequences is expressed by

$$\mathrm{d}^{\mathsf{SEQ}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i) = \sum_{j=1}^{d} \lambda_j \cdot \left(\sum_{k=1}^{j} \mathrm{e}^{-s\cdot(j-k)} \cdot \sum_{u=1}^{v} (x_{k,u} - w^i_{k,u})^{b_{\boldsymbol{w}}} \right)^{b_B} .$$

Basically, this is the summation of blocks of exponentially weighted differences with an exponential decay rate of $0 \leq s \leq 1$. The third sum from $u = 1$ to v refers to the comparison of the v-dimensional block parts the sequence is composed of. If, for example, the sequence elements are symbols, then their unary encoding are vectors, and v is the size of the alphabet. The saturation of the element distances is integrated by the second sum that depends on the choice of s for the integer mapping $k \mapsto \exp(-s\cdot k)$. This term leads to the limit of the geometric series $\sum_{k=0}^{\infty} \mathrm{e}^{-s\cdot k} = (1-\mathrm{e}^{-s})^{-1}$ characterizing the half-life period of the exponential memory. As a result, a bias is induced at beginning of the sequence for the starting indices, where a superposition with the further past is not available. The focus on the exponentially weighted sequence blocks, the correlation between them, is controlled by b_B, higher numbers leading to a stronger subsequence influence. Although the nested loops for the first two sums looks like an $\mathcal{O}(d^2)$ algorithm, it can be implemented in $\mathcal{O}(d)$ time complexity by exploiting that the currently calculated inner sum can be divided by $\mathrm{e}^{-s\cdot j}$ before adding the difference that refers to the next summation index $j+1$. Already during the distance calculation information can be collected that is required later to compute the derivatives with respect to the weight w^i_j and the relevance λ_j also in $\mathcal{O}(d)$ time. The metric parameters λ_j measure the influence of the exponentially fading memory for the sequence element at position j. Despite the sophisticated structure of the metric $\mathrm{d}^{\mathsf{SEQ}}_{\boldsymbol{\lambda}}$, an improvement over simple Euclidean type metrics could not be achieved though. Still, in a DNA classification experiment discussed later, this metric demonstrates how the nature of sequences can be taken into account by one-way processing.

Bidirectional local window metric (LIK)

A metric similar to $\mathrm{d}^{\mathsf{SEQ}}_{\boldsymbol{\lambda}}$ but more successful can be derived from the locality improved kernel (LIK). This metric, working bidirectionally on subsequences, has been proposed by Schölkopf et al. [133, 175] for a very accurate classification of protein sequences with support vector machines (SVM). Subsequences within a symmetric radius of length l around the current sequence position are compared, and their correlations across the whole sequence are computed by

$$\mathrm{d}^{\mathsf{LIK}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i) = \sum_{j=1+l}^{d-l} \lambda_j \cdot \left(\sum_{k=-l}^{l} \frac{h_k}{h_{norm}} \sum_{u=1}^{v} (x_{j+k,u} - w^i_{j+k,u})^{b_{\boldsymbol{w}}} \right)^{b_B} .$$

As for the SEQ distance, the third sum from $u = 1$ to v refers to the comparison of v-dimensional sequence elements. Around a central element in a symmetric data window of extent $2 \cdot l - 1$, the contribution of single differences are weighted by a decay function $h_k = 1/(s \cdot |k| + 1)$ of the distance $|k|$, normalized by $h_{norm} = \sum_{j=-l}^{l} h_j$. The width of the resulting scaling function is controlled by $s > 0$, producing wide hat shapes for small values of s and narrow hats for large s.

The derivatives for the prototype and relevance update formulas are ([61]):

$$\begin{aligned}
\frac{\partial \mathrm{d}^{\mathsf{LIK}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i)}{\partial w^i_{j,z}} &= \sum_{q=j-l}^{j+l} \lambda_q \cdot \left(\sum_{k=-l}^{l} \frac{h_k}{h_{norm}} \sum_{u=1}^{v} (x_{j+k,u} - w^i_{j+k,u})^{b_{\boldsymbol{w}}} \right)^{b_B - 1} \\
&\qquad \cdot \quad \frac{h_{j-q}}{h_{norm}} \cdot -(x_{j,z} - w^i_{j,z})^{b_{\boldsymbol{w}} - 1} \cdot b_{\boldsymbol{w}} \cdot b_B \, , \\
\frac{\partial \mathrm{d}^{\mathsf{LIK}}_{\boldsymbol{\lambda}}(\mathbf{x}, \boldsymbol{w}^i)}{\partial \lambda_j} &= \left(\sum_{k=-l}^{l} \frac{h_k}{h_{norm}} \sum_{u=1}^{v} (x_{j+k,u} - w^i_{j+k,u})^{b_{\boldsymbol{w}}} \right)^{b_B} .
\end{aligned}$$

As previously discussed, the relevance factors are normalized to non-negative values summing up to 1. Apart from the scaling, a metric very similar to LIK is the shift invariant metric that is used by Bojer et al. for the monitoring of rotating piston compressors [13].

An interesting special case of the above LIK metric is the radius of $l = 0$ for which correlations are not taken into account; as an extension of the Euclidean distance, data blocks of a fixed dimension v can be considered as a whole. Different exponents b_w for the intra-block weighting and b_B for the inter-block weighting allow the control of substructure matching. This metric will be called the weighted block Euclidean distance $\mathrm{d}^{\mathsf{BEUC}}_{\boldsymbol{\lambda}}$.

Supervised Relevance Neural Gas (SRNG)

No word has yet been said about prototype initialization: since the considered extensions of GLVQ drag prototypes towards data belonging to the same class and push away the wrong prototypes, attention must be paid to prototype configurations for which well-fitted prototypes prevent badly fitted from reaching their data cluster; the dislocated prototypes

might be repelled from wrong clusters, this way driving them into idle regions in the data space. In high dimensions with many degrees of freedoms, such suboptimal cases with an equilibrium of attraction and repulsion are rare. Nevertheless, prototype convergence to proper data space locations is significantly faster for fairly initialized prototypes. This can be achieved by combining the supervised case with neural gas learning as proposed by Hammer, Strickert and Villmann [59, 162].

The key idea for SRNG is to integrate the cooperative element of neural gas NG into the prototype neighborhood of GRLVQ: for a given training pattern, the class-related prototypes perform rank-based updates according to GRLVQ towards this pattern, and the closest wrong prototype is pushed away. As a result, prototypes spread over the data by the combination of the active GRLVQ class separation with the unsupervised NG.

An extension of the GRLVQ cost function by neighborhood cooperation for prototypes of the same class yields

$$\mathrm{E}_{\mathsf{SRNG}} := \sum_{i=1}^{n} \sum_{j \,|\, \mathrm{c}(\boldsymbol{w}^j)=\mathrm{c}(\mathbf{x}^i)} \mathrm{g}\left(\mathrm{q}^j_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^j) \; .$$

The inner sum takes all distances into account for prototypes matching the class of $\mathbf{x}^i$:

$$\mathrm{q}^j_{\boldsymbol{\lambda}}(\mathbf{x}^i) = \frac{\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) - \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)} \quad \text{where} \quad \mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) = \mathrm{d}_{\boldsymbol{\lambda}}(\mathbf{x}^i, \boldsymbol{w}^j) \; .$$

Neighborhood cooperation is modeled by the exponential ranking of neural gas:

$$\mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^j) = \exp(-\mathrm{rnk}(\mathbf{x}^i, \boldsymbol{w}^j)/\sigma) \cdot \mathrm{s}(\sigma, \boldsymbol{w}^j) \quad \text{with}$$

$$\mathrm{rnk}(\mathbf{x}^i, \boldsymbol{w}^j) = \Big| \{k \,|\, \mathrm{c}(\boldsymbol{w}^k) = \mathrm{c}(\boldsymbol{w}^j) \;\wedge\; \mathrm{d}(\mathbf{x}^i, \boldsymbol{w}^k) < \mathrm{d}(\mathbf{x}^i, \boldsymbol{w}^j)\} \Big| \; .$$

The influence of the neighborhood cooperation $\mathrm{h}_\sigma(\cdot)$ is determined by the control parameter $\sigma > 0$; during training σ is decreased from an initial neighborhood size to small values, thus fading from a coarse global ordering to fine local adaptation. The scaling factor $\mathrm{s}(\sigma, \boldsymbol{w}^j)$ provides normalization with respect to the number q of prototypes representing the same class:

$$\mathrm{s}(\sigma, \boldsymbol{w}^j) = 1 \Big/ \sum_{k=0}^{q-1} \exp(-k/\sigma) \quad \text{with} \quad q = \big| \{z \,|\, \mathrm{c}(\boldsymbol{w}^z) = \mathrm{c}(\boldsymbol{w}^j)\} \big| \; .$$

$\mathrm{E}_{\mathsf{GRLVQ}}$ becomes an instance of $\mathrm{E}_{\mathsf{SRNG}}$ for small neighborhoods and $\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) \neq \mathrm{d}^{i-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)$: for $\sigma \to 0$ both the exponential function $\exp(-\mathrm{rnk}(\mathbf{x}^i, \boldsymbol{w}^j)/\sigma)$ and the normalization $\mathrm{s}(\cdot)$ approximate 1. Thus, in the cost function $\mathrm{E}_{\mathsf{SRNG}}$ the neighborhood term $\mathrm{h}_\sigma(\cdot)$ is 1 only for the closest correct prototype. Therefore $\lim_{\sigma \to 0} \mathrm{E}_{\mathsf{SRNG}} = \mathrm{E}_{\mathsf{GRLVQ}}$.

Just as for GRLVQ, update formulas result from the partial derivatives of $\mathrm{E}_{\mathsf{SRNG}}$:

$$\triangle \boldsymbol{w}^{+j} = -\gamma^{+} \cdot \mathrm{g}'\left(\mathrm{q}^{j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^{+j}) \cdot \frac{2 \cdot \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\left(\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right)^2} \cdot \frac{\partial \mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\partial \boldsymbol{w}^{+j}}$$

$$\triangle \boldsymbol{w}^{i-} = \gamma^{-} \cdot \sum_{j \,|\, \mathrm{c}(\boldsymbol{w}^j)=\mathrm{c}(\mathbf{x}^i)} \mathrm{g}'\left(\mathrm{q}^{j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^{j}) \cdot \frac{2 \cdot \mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\left(\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right)^2} \cdot \frac{\partial \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)}{\partial \boldsymbol{w}^{i-}}$$

$$\triangle \boldsymbol{\lambda} = -\gamma^{\boldsymbol{\lambda}} \cdot \sum_{j \,|\, \mathrm{c}(\boldsymbol{w}^j)=\mathrm{c}(\mathbf{x}^i)} \mathrm{g}'\left(\mathrm{q}^{j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right) \cdot \mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^{j}) \cdot \frac{2 \cdot \partial \mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i)/\partial \boldsymbol{\lambda} \cdot \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i) - 2 \cdot \mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) \cdot \partial \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)/\partial \boldsymbol{\lambda}}{\left(\mathrm{d}^{+j}_{\boldsymbol{\lambda}}(\mathbf{x}^i) + \mathrm{d}^{-}_{\boldsymbol{\lambda}}(\mathbf{x}^i)\right)^2}$$

These SRNG formulas are exactly the same as for GRLVQ, extended by the neighborhood factor $\mathrm{h}_\sigma(\mathbf{x}^i, \boldsymbol{w}^j)$ related to the neural gas dynamic of all correct prototypes. By the same arguments as for the above cost function, GRLVQ update is reobtained for the limit case $\sigma \to 0$ which leads to $\mathrm{h}(\cdot) = 1$ only for the closest correct prototype. Detailed formulas for the derivation of the adaptation rules and a proof of the differentiability of the SRNG cost function are given by Hammer et al. [61].

The training of a given data set can be done in several modes: prototype adaptation only, relevance adaptation only, or both combined. Like for GRLVQ, it turns out that keeping the relevances λ_i fixed at the beginning and releasing them after a number of training cycles on a time scale slower than the prototype adaptation, this will result in $\boldsymbol{\lambda}$ characterizing well the relevance of input dimensions for the more or less settled prototypes. These different time scales can be realized by choosing $0 \leq \gamma^{\boldsymbol{\lambda}} << \gamma^{-} < \gamma^{+} \leq 1$.

5.2 BB-Tree: Rules from trained GRLVQ or SRNG networks

Networks of LVQ type constitute similarity-based classifiers that take *simultaneously* into account all data dimensions for finding the closest prototype to a given pattern and the searched label. Opposed to this, a classification tree successively computes characteristic partitions that lead to class separation for *each individual* dimension. For efficiency, the most relevant dimension is split into partitions first; then, depending on the currently active partition, the second relevant dimension is split, and so forth, until a possibly unique label can be assigned to a pattern of which the components have seeped through the different stages of decision.

Rule-based descriptions are particularly well suited for labeled symbolic data, since frequency and information measures can be obtained from their discrete domains in a straightforward manner without prior partitioning; symbol frequencies make statistical and information theoretical classification schemes possible [17, 38], and operations on symbols allow case-based reasoning techniques [138], prolog-like knowledge-based artificial neural networks with rule induction KBANN [150, 151], and inductive logic programming [110].

Rule acquisition for continuous data is possible after turning real-valued attributes into an intervals by means of partitioning. For standard LVQ, a suitable method has not yet been proposed to obtain rules from the trained prototypes; a major difficulty is that the order of the attribute processing is unclear. However, the presented LVQ extensions to

GRLVQ and SRNG yield a built-in detection of the data attributes' relevances from which a partitioning order can be derived for components of adjacent prototypes. For the case of the weighted Euclidean diagonal metric, a method is presented to convert a trained network into a classification tree.

5.3 LVQ and rule extraction

LVQ classification is determined by the Voronoï cells which are convex and which possess piecewise linear hyperplane boundaries in case of the Euclidean metric. Since the input space allows oblique boundary orientations, a compact and intuitive class characterization is not easily attainable. It is much more convenient to think in terms of axes-parallel input space cuts that define hypercubes which capture particular data. Such a cube is a rule expressed by the AND-concatenation of intervals belonging to the considered dimensions. A set of rules can be layered by the dimensions and then be compactly represented as a decision tree which can be interpreted easily. The proposed technique for the conversion from Voronoï cells into hypercubes is called the BB-tree algorithm and, as far as known by the author, it is the first rule extraction technique for LVQ-based architectures [57].

It is not the aim of this work to create classification trees better than those resulting from the traditional programs like CART of Breiman et al. [17], the C4.5 method of Quinlan [121], QUEST of Loh and Shih [96], or DIPOL of Wysotzki et al. [172]. Instead, the BB-tree method demonstrates that sub-symbolic information contained in a trained extended LVQ net can be commuted into easily interpretable rules with hight classification accuracy [57]. Historically, the topic of rule extraction from neural networks started with the KBANN algorithm proposed by Towell and Shavlik [150]; later, McGarry et al. have formulated rule extraction for radial basis function networks [104]; for modified multilayer perceptrons, Duch, Adamczak, and Grabczewski have developed the MLP2LN learning scheme [36]. A survey of rule extraction from classical neural network architectures is given by Tickle et al. [148]; more recently, Núñez et al. [111] suggest a simple approach to rule extraction from support vector machines.

The information required for tree construction can be obtained in a straight forward manner from a trained GRLVQ/SRNG network: the order of splitting is taken from the descendingly sorted dimension relevances given by vector $\boldsymbol{\lambda}$, and the interval boundaries are derived from natural splitting locations between the prototypes. Due to the relevance ranking, only the most important dimensions are candidates for the corresponding intervals of interest. In a recursive manner, starting at the empty root node, a decision tree is assembled with these properties: each node N of the tree may possess an arbitrary number k_{N} of children; an interior node T is assigned a valid dimension index $1 \leq I_{\mathsf{T}} \leq d$ for which the splitting points are represented by real values $b_1^{\mathsf{T}} < \ldots < b_{k_{\mathsf{T}}-1}^{\mathsf{T}}$; each leaf L is labeled with a class number $1 \leq c_{\mathsf{L}} \leq c$. Classifying a data point requires to take the decision at the current node T, which child to choose, i.e. which path to follow. This decision

refers to the projection of the data point to its $I_{\mathsf{T}}{}^{th}$ component and to the subsequent selection of the index of the surrounding interval $(b_j^{\mathsf{T}}; b_{j+1}^{\mathsf{T}}]$ from the node's ordered list of interval boundaries b_j^{T}. Since the obtained interval refers to the next chosen child node, a given point iteratively percolates to a leaf node, where finally the searched class label is found. For the efficient tree construction, the most selective dimensions should be discriminated first and should constitute the root node or near root top level nodes; the important dimensions are obtained by ranking the trained GRLVQ/SRNG dimension relevances λ_k. The nodes' entries b_j^{T} describing the interval borders are taken as projections of the midpoints of two adjacent prototypes to the currently processed dimension.

Technically, the BB-tree generator is assembled as follows: Λ is the list of indices j, sorted according to the magnitude of the weighting factors λ_j; its first entry is $\mathit{first}(\Lambda)$, the rest of the list is denoted by $\mathit{rest}(\Lambda)$; $\mathbf{X}$ refers to the training set and $\mathbf{W}$ is the set of prototypes. For reasons of efficiency, it is assumed that all dimensions j which, due to a small weighting factor λ_j, do not contribute to the classification are ignored, and all prototypes with empty receptive fields are removed before starting Algorithm 3. As

Algorithm 3 BB-Tree ($\mathbf{X}$,$\mathbf{W}$,Λ)

if *STOP* **then**

output leaf L with class $\mathrm{argmax}_c|\{\mathbf{x}^i \mid y^i = c,\ (\mathbf{x}^i, y^i) \in \mathbf{X}\}|$

else

output an interior node T with $|\mathbf{W}|$ children

choose $I^{\mathsf{T}} := \mathit{first}(\Lambda)$

compile a sorted list $[a_1, \ldots, a_{|\mathbf{W}|}]$ from $\{w^i_{I^{\mathsf{T}}} \mid \boldsymbol{w}^i \in \mathbf{W}\}$

set splitting points $b_0^{\mathsf{T}} := -\infty$, $b_{|\mathbf{W}|}^{\mathsf{T}} := \infty$, $b_i^{\mathsf{T}} := (a_i + a_{i+1})/2,\ i = 1, \ldots, |\mathbf{W}| - 1$

choose the i^{th} child of T, $i = 1, \ldots, |\mathbf{W}|$, as the output of

$$\text{BB-Tree}\left((\mathbf{x}, y) \in \mathbf{X} \mid x_{I^{\mathsf{T}}} \in (b_{i-1}^{\mathsf{T}}, b_i^{\mathsf{T}}],\ \mathbf{W},\ \mathit{rest}(\Lambda) \bullet [\mathit{first}(\Lambda)]\right)$$

end if

indicated by the list concatenation operator $\bullet$ and by the recursive formulation of the algorithm, the height of the tree is technically not limited by the number $\tilde{n}$ of entries in the rank list. Due to the recursive splitting, the same dimension can be visited multiple times for partitioning and still produce different data subsets, depending on the previously selected partitions. Such a dimension revisitation might be useful, if an alternative splitting heuristic is used, such as executing a split only in the case that the resulting interval is large enough, e.g. larger than the average splitting interval [57]. In the experiments, the complete midpoint splitting has been used, and a *STOP* has been triggered for trees higher than $d_S \leq d$ without wrapping-around the dimensions. Empirically, d_S is chosen according to satisfying classification requirements for both training set and test set without

producing too many rules. Other stopping criteria are useful, if the splitting intervals are calculated differently. By varying both constituents the tree height and the splitting point calculation, classifiers of higher accuracy might be obtained; however, a lower model generality has been observed for very high trees.

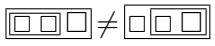
The presented method for rule learning takes place at two stages: the first one is a similarity-based pattern classification, using GRLVQ for prototype positioning and dimension relevance detection, the second one is the replacement of the thus obtained Voronoï cells by rectangular subsets of the data space. Such a data processing integrates both soft pattern representations and crisp rules, this way constituting a tentative approach towards the realization of a hybrid data model.

Rule simplification

As said before, single paths in the tree correspond to a rule that is established by the AND-combination of individual decisions at each node. This class representation is equivalent to a hypercube decomposition of the input space, possibly restricted to a subspace of only the relevant dimensions. Different cubes containing the same class can be merged by OR [90]. Technically, the merging is achieved for adjacent cubes of the same class simply by extending the intervals corresponding to a certain data dimension to the two most distant boundaries. For disjoint cubes of the same class, a merging to their common convex hull or a suitable product space of the relevant dimensions may be possible, if only a negligible amount of points belonging to other classes is located in this wrapped-up region. Optimum subspace selection has not been considered, because this is a computationally expensive operation: a number of d dimensions would require $\mathcal{O}(2^d)$ probing operations. However, some simplifications can be still obtained automatically: due to the boundary extension, iterative merging produces intervals of the form $(-\infty;\infty)$. Especially for large d, many intervals can just be omitted then. The whole procedure with its greedy rule merging component is given by Algorithm 4.

A purity measure is a central criterion in the algorithm for the decision if the merging of two boxes representing the same class should be done or avoided. This measure refers to the evaluation of the generated bounding box of both source cubes: training samples are presented and the number of points falling into this cube are counted with respect to the correct and wrong labels. This counting statistics is used to maximize the relative number of correctly classified points $z = N_{correct}/(N_{correct} + N_{wrong})$ with $0 \leq z \leq 1$.

In the field of geometric data optimization a problem very similar to the merging decision can be found: collision detection for 3-dimensional geometric shapes involves the construction of optimal bounding boxes around the points that define an object. Automatic bounding box generation requires the construction of the convex hull of point clusters, which may result in axes parallel bounding boxes (AABB) or boxes oriented along the principal components (OBB) of the geometry [51]. Then, rough geometry collision can be efficiently tested by just calculating the intersections of the corresponding

bounding boxes. Bounding boxes for parts of an object can be contained in a high level box for the whole object. For such hierarchical setups, the super bounding box can be obtained by pairwise putting together boxes that minimize the wrapped-up volume, as proposed for the BOXTREE system [4, 5].[1] If now volume minimization in 3D is replaced by purity maximization in the high-dimensional data space for axes-parallel boxes, this explains the basic idea of the rule reduction Algorithm 4. Since the successive bounding box construction is non-associative, i.e.  $\neq$, the iterative greedy procedure cannot produce optimum results with respect to the purity maximization. However, in high dimensions, a lot of merging can be done before the misclassification rate of the simplified rule sets becomes unacceptably high.

In a final step, the order of rule evaluation for the classification must be given. If the hypercubes found are disjoint, an arbitrary order can be used, but for overlapping hypercubes that represent different classes, the boxes with higher purity are preferred, because the associated rules are more reliable. Hence a purity-based rule ordering is recommended. As additional default, each data point that does not fall into any cube after the rule reduction will be assigned the class dominantly present in that case.

Patterns and Rules

The presented extraction of a classification tree from a prototype-based classifier contributes to the interesting discussion of integrating patterns and rules [16]. Thereby, the patterns can be identified by the variform convex shapes of the prototypes' Voronoï cells and the rules are the axes parallel input space cuts. Thus, Voronoï boundaries account for an intuitive notion of cross-dimension, similarity-based relationships, while the axes aligned hyperboxes capture intra-dimensional separation criteria. As in real life, sometimes the rule-based data representation leads to a compact model, and sometimes only a few patterns can express what would require a lot of particular and badly generalizing rules. This rating will be supported by experiments.

[1] Thanks to Sascha Teichmann for providing a 2D JAVA implementation of BOXTREE.

Algorithm 4 Rule simplification

Initialization: take training data and an according tree obtained from the BB algorithm

{ use: BB-tree for obtaining initial set of hypercubes with class labels }

{ use: training data for purity calculation }

{ Greedy probing for rule merging }

for all classes c **do**

  repeat

    $z \leftarrow 0$

    for all pairs $(h^i, h^j)_{i \neq j}$ of hypercubes for class c **do**

      $h \leftarrow$ bounding-box(h^i, h^j)

      $z \leftarrow \max(z, \text{purity}_c(h))$

    end for

    if $z < purity_{crit}$ **then**

      $(i, j) \leftarrow$ index pair of hypercubes belonging to z

      $h^i \leftarrow$ bounding-box belonging to z

      purge hypercube h^j

    end if

  until $z \geq purity_{crit}$

end for

{ Strip trivial dimensions }

for all hypercubes h^i **do**

  purge dimensions m with interval $h^i_m = (-\infty; \infty)$

end for

Good judgement comes from experience, and experience — well,
that comes from poor judgement.

Cousin Woodman

5.4 SRNG experiments

The performance of supervised relevance neural gas will be demonstrated for different kinds of data: ⟨1⟩ a synthetic real-value multi-modal data set is used for general illustration purposes; ⟨2⟩ the discrete nominal mushroom data set from the UCI learning repository [11] introduces rule extraction; ⟨3⟩ the UCI hypothyroid data are used for demonstrating data set augmentation and the simultaneous processing of real and Boolean data; ⟨4⟩ discrete linguistic data referring to diminutive prediction in Dutch tackles the processing of syllable sequences; ⟨5⟩ a time series from the Lorenz dynamic system introduces the concept of relevance embedding; ⟨6⟩ a speaker identification data set from UCI points out the excellent classification performance for sequences of 12-dimensional real-value elements; and ⟨7⟩ DNA sequence data for splice site recognition demonstrate the applicability of the proposed methods to large real-life data sets.

Data preparation for GRLVQ and SRNG

Before training can be started, data must be given appropriate shape. Many data sets exhibit inhomogeneous class distributions: some classes may be represented by only a small number of data points, but other classes may be dominantly available. For GRLVQ and likewise SRNG, such an imbalance would yield a biased prototype update, because the frequent presentation of data points belonging the largest class would implicitly push away prototypes of smaller classes in the neighborhood. Therefore, the training set should be augmented to provide a fairer class representation. As discussed in Section 3 on page 10, instead of simple pattern replication a decent amount of jittering can be added in order to provide different training data for better learning and generalization.

Attribute normalization is also advisable for preparation; as discussed in Section 3 on page 10, especially the (squared) Euclidean distance metric profits from transforming the data components according to the z-score method.

Synthetic 10-dimensional data

For a first illustration, an artificial data set with overlap is used. Points $\mathbf{x} \in \mathbb{R}^{10}$ have been generated by adding to the 2-dimensional Gaussian clusters (x_1, x_2) depicted in Figure 5.1 further dimensions $x_3 = x_1 + \xi_3, \ldots, x_6 = x_1 + \xi_6$, where ξ_i is distributed according to Gaussian noise, with variances $\sigma_3^2 = 0.05$, $\sigma_4^2 = 0.1$, $\sigma_5^2 = 0.2$, and $\sigma_6^2 = 0.5$; attributes

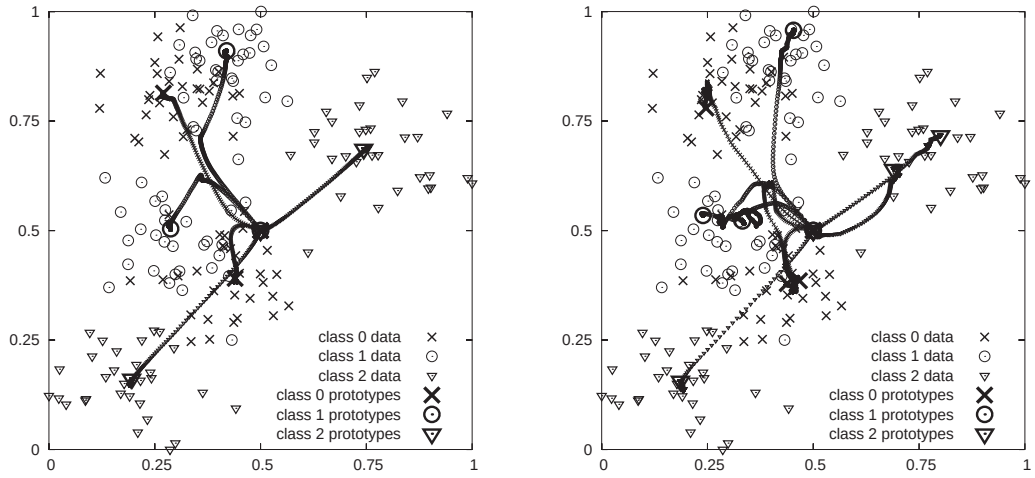


Figure 5.1: Artificial 10D data set with three classes, projected to the generating dimensions. Left: one prototype per cluster. Right: one extra prototype per class.

$x_7, \ldots, x_{10}$ contain randomly scaled white noise. This generator has been used to produce a total number of 180 samples in $[0; 1]^{10}$ describing bi-modal distributions for the three classes around the driving components (x_1, x_2).

Training with SRNG has been carried out for the squared Euclidean metric with squared relevance factors λ_i and a number of two prototypes per class. The training parameters were set to $\gamma^+ = 10^{-4}$, $\gamma^- = \frac{1}{2} \cdot \gamma^+$ and $\gamma^{\boldsymbol{\lambda}} = 10^{-6}$ with an initial neighborhood size of 2 that exponentially decayed to 10^{-5}. A number of ten training runs with 5,000 epochs have produced an average accuracy of 83.8%±1.0%. For 3×3 prototypes, the accuracy has increased to 85%±0.9%. Figure 5.1 shows projections of the prototype trajectories during training; the left panel shows the case for one prototype per cluster, the right panel illustrates the case of three prototypes for only two clusters per class. Training has started with prototypes $w^i_{1\ldots10} = 0.5$ and the initial relevance vector$\lambda_{1\ldots10} = 0.1$. Typically, weighting factors λ_i have converged under the normalization constraint $\sum_{i=1}^{d} |\lambda_i| = 1$ to $\boldsymbol{\lambda} = (0.29, 0.29, 0.29, 0.13, 0, 0, 0, 0, 0, 0)$. These values show that the important first two data dimensions are clearly found; also, the small variance of 0.05 on the third component has no negative effect on the classification, while the other noise components, except for the fourth, are completely canceled. For the quartic Euclidean distance with λ_i^2, an average accuracy of 86.6%±0.7% has been obtained by using only six prototypes with $\boldsymbol{\lambda} = (0.24, 0.16, 0.23, 0.17, 0.11, 0.09, 0, 0, 0, 0)$; this result indicates that the first component is heavily exploited for the classification, and that also its noisy replications are interpreted.

Curves of both training and test accuracy, tracked during the learning phase, indicate the generalization capabilities and they can be used for model validation. Figure 5.2 shows a typical plot that has been obtained for a splitting of the whole data set into a 75% training partition and 25% testing partition. In the plot, snapshots of 100 sampling stages have been smoothed by natural Bezier interpolation, and the first 500 epochs have been omitted. For the training set, saturation at about 81% is found after 1,500 cycles.

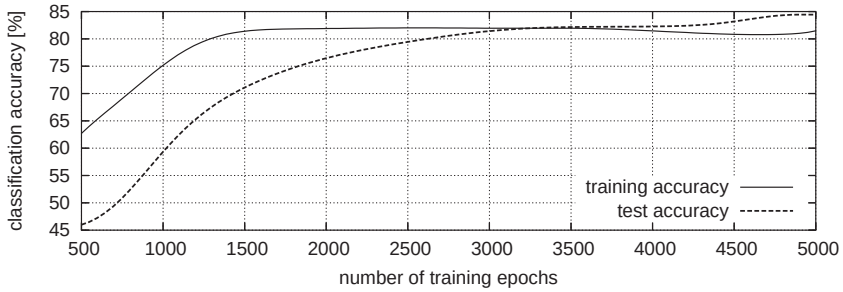


Figure 5.2: Training error and test error for the synthetic data set.

Further relevance adaptation leads to a successive increase of the test set accuracy while lowering the overspecialization on the training set. Usually, and as a desirable feature of SRNG, relevance adaptation helps to reduce overfitting data.

Another experiment has focused on the influence of only the relevance factors on the classification. For this purpose, the prototype locations $\boldsymbol{w}^k$ have been frozen by setting $\gamma^+ = \gamma^- = 0$, thus allowing only λ_i to adapt. Two prototypes per class have been randomly picked from the data, and 5,000 epochs have been trained. Table 5.1 summarizes the parameters and the classification results for the weighted squared Euclidean metrics with relevance exponents $b_{\boldsymbol{\lambda}} = 1, 2$. The reported averages accuracies (avg. acc.) have been obtained from 50 independent runs for each metric; the given learning rates maximize the accuracies. Two things are remarkable: $\langle 1 \rangle$ a significant classification improvement can be obtained already without changing the prototype locations, and $\langle 2 \rangle$ the unit exponent yields significantly better results. The first observation makes it attractive to think about utilizing individual metric parameters for each prototype, because already the single global relevance vector $\boldsymbol{\lambda}$ leads to a good improvement; however, future work on this idea must take regularization strategies into account in order to avoid that the adjustment of many free parameters relies on too few data. A possible explanation for the very recent second finding is the race condition between the competing dimensions: the update influence on the λ_j by a unit exponent is constant, but for an exponent of 2 it is in Hebbian terms proportional to the current value. Proportionality boosts already large dimensions, while constancy leads to more fairness. The fairness associated with the exponent $\boldsymbol{\lambda} = 1$ comes along with better tolerance for the choice of the learning rate γ^- which proved to be harder to tweak for larger exponents. For all subsequent experiments, the seemingly disadvantageous exponent of 2 has been predominantly used due to historic reasons that aimed at the 'natural' Hebbian update term for the dynamic of the dimension factors. Hence, there is still potential for improving the following results by systematically applying the $\boldsymbol{\lambda}$-exponent $b_{\boldsymbol{\lambda}} = 1$ as originally suggested in Hammer and Villmann [64, 65].

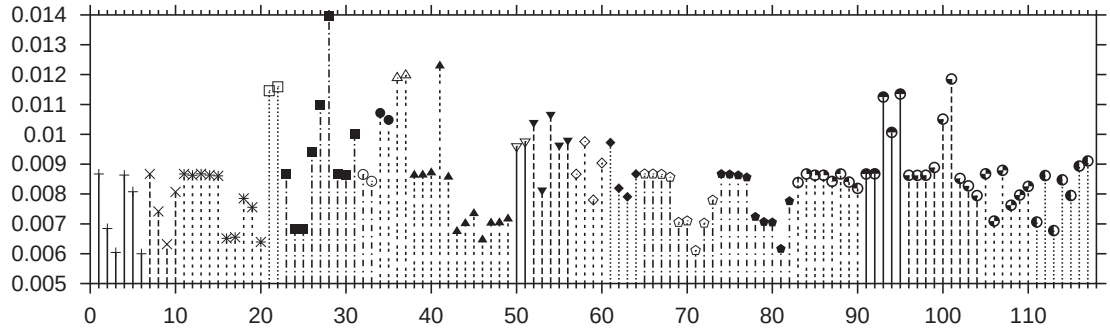


Figure 5.3: Adapted weight factors $\lambda_{1\leq i\leq 117}$ for the mushroom data. Different line styles and points indicate the membership to the 22 different attributes.

Mushroom data

The mushroom data set from the UCI repository consists of 8,124 vectors representing 22 symbolic attributes, some are binary, others take up to 12 different states. These attributes have been converted by unary encoding into partial vectors which have been concatenated to 117-dimensional pattern vectors. Again, the z-score transformation has been applied to the obtained vector set. Labels for the two classes are **e**=edible, which belongs to 51.8% of the data, and 48.2% of **p**=poisonous examples. A portion of 75% of this data set has been randomly taken for training, 25% for validation. In a first training step with 1,000 epochs, ten prototypes have been used for each class. Learning rates are $\gamma^+ = 0.001$, $\gamma^- = \frac{1}{2} \cdot \gamma^+$, and no dimension weight adaptation $\gamma^{\boldsymbol{\lambda}} = 0$; the neighborhood size is annealed from 10 to 10^{-5}. The calculated numbers of data in the receptive fields have showed that five of the prototypes could be removed without significantly affecting the classification accuracy of roughly 92%; thus, three prototypes have remained for the poisonous case and two for the edible. Training has been refined with an initial neighborhood size of practically zero, setting $\gamma^+ = \gamma^- = 0.00075$ and $\gamma^{\boldsymbol{\lambda}} = 10^{-7}$. Tenfold cross-validation produces classification accuracies $\geq 97.5\%$ on the test data. The best set of prototypes with its adapted metric weights $\boldsymbol{\lambda}$ yields 98.7% accuracy. This set has been used for the BB-tree extraction procedure. The relevances for the different components are shown in Figure 5.3. Within the odor attribute block, feature 28.odor:none displays highest discrimination ability. This feature is followed by 41.gill-color:buff, 37.gill-size:narrow, 36.gill-size:broad, 101.spore-print-color:chocolate, 22.bruises:no, 21.bruises:yes, 95.ring-type:pendant, 93.ring-type:large, and 27.odor:foul, respectively. Obviously, for binary items like the bruises, both possible states

$\boldsymbol{\lambda}$–exponent	γ^-	initial avg. acc.	final avg. acc.	avg. gain
1	$5.0 \cdot 10^{-4}$	$47.89\% \pm 5.61\%$	$64.24\% \pm 7.81\%$	16.35%
2	$2.5 \cdot 10^{-5}$	$47.87\% \pm 5.16\%$	$60.41\% \pm 8.29\%$	12.54%

Table 5.1: Results for exclusive relevance adaptation for the artificial 10D-data

complement each other. The relevances are also reflected in the final set of rules which have been extracted by a BB-tree of height 6 and which are given in Table 5.2. These rules explain 97.2% of the test set and 97.5% of the training set. An amount of 88.7% can be classified correctly by using the single rule odor=none → **e**; the inverse rule for mapping to the poisonous case **p** accounts for only for 0.1% of the data. Still, redundancies are visible in the set of rules, because the gill-size attribute appears twice. In that situation, the rule simplification algorithm failed to handle the don't care state (-).

By means of the four rules for the edible case, a good data representation is achieved with six components; the poisonous class is subsumed in the ELSE case. These findings are not quite as good as compared to the results of Duch et al. [36] who obtain 100% accuracy with three rules and five components. But interestingly, the same important attributes have been identified: odor, gill-size, and spore-print-color.

Hypothyroid data

A difficult separation task is given by the hypothyroid data from the UCI repository. There are three classes, **n**=normal, **p**=primary hypothyroid, and **c**=compensated hypothyroid, determined by 21 attributes, 15 of which are binary and 6 are continuous. In the training set there are 3,772 cases, and 3,428 cases are in the test set. Due to the large overlap between the classes and the strong prominence of the **n**=normal class with a portion of about 92%, this data set is known to be difficult for neural classification [132].

The first SRNG runs with the original data and have produced only the trivial 92% accuracy that could have been more easily obtained by classifying all data as the normal state. In a second run with training set and test set augmented to 10,000 cases each, this result has even dropped to only 78%. A z-score transform could not significantly improve the situation.

bruises 22:no	odor 28:none	gill-size 36:broad	gill-size 37:narrow	gill-color 41:buff	spore-print-color 101:chocolate	Class	Freq.
-	0	-	-	1	-	**p**	21%
-	0	1	0	0	1	**p**	19%
0	0	-	-	0	0	**p**	3%
1	0	0	1	0	0	**p**	4%
1	1	0	1	0	-	**p**	0.1%
0	1	-	0	0	-	**e**	16%
1	0	1	0	0	0	**e**	8%
1	1	-	0	0	-	**e**	25%
0	1	0	1	0	-	**e**	3%

Table 5.2: Rules from a BB-tree of height 6 explaining 97.2% of the mushroom data.

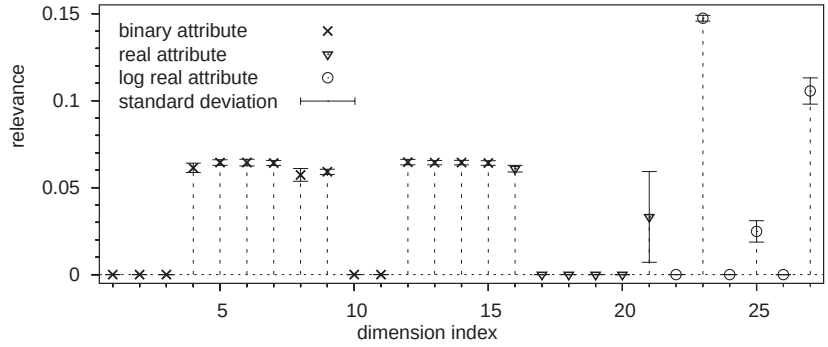


Figure 5.4: Average adapted weight factors $\lambda_{1\leq i\leq 27}$ with standard deviations for 20 experiments on the hypothyroid data. Different line styles and points indicate the membership to the binary (left), real (center), and logarithms of the real attributes (right).

A strong enhancement has been achieved by including logarithms of the real-value components as additional data dimensions. This step has been discussed in the data preparation Section 3, and it is supported by Figure 3.1 on page 14. For the weights shown in Figure 5.4 and with a number of five prototypes per class, the classification accuracy has increased to 97.1%. Interestingly, data augmentation has a negative effect on the correct prediction of the 'trivial' normal class: the worst prototype responsible for the **n** class has misclassified 11.1% of all cases, whereas the worst prototype for **c** has failed only in 5.5%, and for the primary hypothyroid **p** class no misclassification has been encountered at all. In other words: if one of the hypothyroid cases is predicted, this is a very certain result, whereas the diagnosis of the normal case is not; the certainty about exceptional states and the alert rising in the normal cases induced by the data augmentation might be a desirable feature in diagnostics for drawing the attention to potentially interesting situations.

A clear benefit of relevance-based clustering has been shown in this experiment: transformed data can just be added to the data as new dimensions. If these dimensions turn out to be worthless for the classification, their dimension weighting factors will vanish.

Plot 5.4 shows that some of the binary features have been detected as irrelevant in 20 independent runs. A high reliability is indicated by low standard deviations, only attribute 21:`FTI` is uncertain. The most prominent dimensions are 23:`ZLog(TSH)` and 27:`ZLog(FTI)`; these results are confirmed for the non-transformed values by the findings of Duch et al. [36]. As related to the weight ranking shown in Figure 5.4, the simplified extracted rules for a BB-tree of height three can be reduced to these two features. The three obtained rules are:

$$\begin{aligned}
&\texttt{ZLog(TSH)} \in (-0.347;\infty] \quad \wedge \quad \texttt{ZLog(FTI)} \in (-\infty;0.092] \rightarrow \textbf{p} \quad (97\%)\,,\\
&\texttt{ZLog(TSH)} \in (0.366;\infty] \quad \wedge \quad \texttt{ZLog(FTI)} \in (0.068;\infty] \rightarrow \textbf{c} \quad (94\%)\,,\\
&\texttt{ZLog(TSH)} \in (-\infty;-0.347] \qquad\qquad\qquad\qquad\quad \rightarrow \textbf{n} \quad (100\%)\,.
\end{aligned}$$

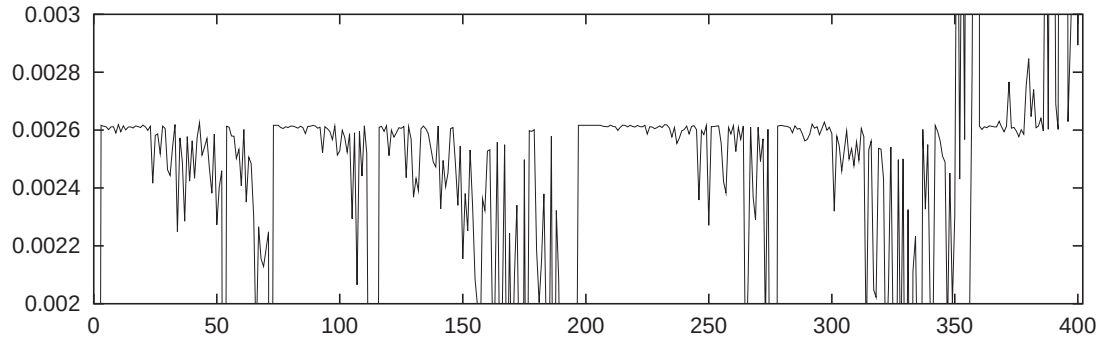


Figure 5.5: Adapted weight factors $\lambda_{1\leq i\leq 403}$ for the diminutive data.

With these rules, 96.2% accuracy is obtained for the test set and 97.0% for the training set. Increasing the height of the extracted tree to 10 includes the binary data; then, a high number of rules with poor generality is generated: the accuracy of the training set increases to 99.1%, but the test set is predicted correctly in only 93.2% of all cases. For comparison, a feedforward net with backpropagation produced an average accuracy of 96.13% on the training set and 96.18% on the test set [112], but original CART classification tree methods improve the results to accuracies of 99.8% and 99.36% for training and testing [168].

Linguistic data

In the next experiment, a classification problem for linguistic data demonstrates sequence processing with SRNG. Thereby, the quality of the SRNG-based BB-tree rule extraction is compared to the nearest neighbor classifier TiMBL which has been designed for the induction of linguistic knowledge [33]. The TiMBL package supplies a test set with SAMPA[2] coded syllable information of Dutch words, for which one of the five possible diminutive forms **-je**, **-etje**, **-pje**, **-kje**, and **-tje** shall be predicted.

Since TiMBL is a nearest neighbor classifier, it stores all 2,999 items of the diminutive training data in memory. New data are classified according to a majority vote of examples with best matching overlap [33]. For the 950 test cases, TiMBL achieved an accuracy of 96.6%. SRNG training has been performed with just four prototypes per class for the unary encoded z-scored data augmented to equal class sizes. The final result for the prototype-based classification already provides 92.6% accuracy on the test set and 92.3% on the training set. Figure 5.5 displays the obtained metric weights. Since patterns are aligned to the word endings and organized in triples of vectors with the four feature components (stress, onset, nucleus, coda), the weights reflect the intuitive fact that the word endings determine the choice of the proper diminutive suffix.

Another interesting result is given: the rules for a BB-tree of height 25 yield an improved classification accuracy of 95.5% and 95.1% for the test and training set, respectively. Unfortunately, this height of 25 corresponds to a set of 117 rules which cannot be simplified

[2]Speech Assessment Methods Phonetic Alphabet. See `http://www.phon.ucl.ac.uk/`

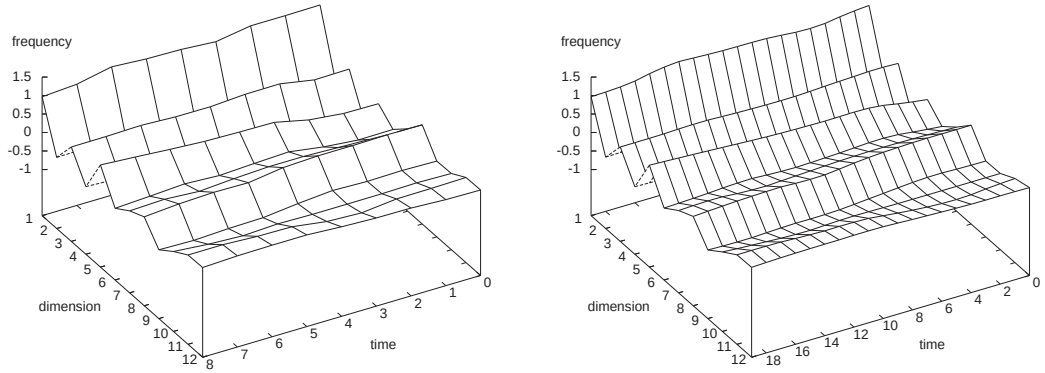


Figure 5.6: Cepstrum vectors for an 'ae' articulation. Left: original. Right: resampled.

easily. After all, this result is not too bad, because it might give a hint for a dilemma in linguistic research, whether human linguistic skills are rule-based rather than analogical pattern-based, or vice versa. Both seems to be valid: On one hand, some rules have a high degree of generality and account for up to 10% of all cases. On the other hand, the large number of special rules indicate that many exceptions have to be learned extra. As a reference, the C4.5rules program [121] has generated 71 rules, and its accuracy is about 97.1% for the test set and 97.5% for the training set.

Speaker identification

The next experiment processes speaker data from the UCI repository[3]. Recordings from nine speakers of the Japanese vowel 'ae' are given as sequences of 12-dimensional frequency vectors, based on a time-window Fourier transform of the sampled waveform. Each utterance comprises a number between 7 and 29 of temporally connected vectors, roughly 16 on average. In the training set, 30 articulations are available for each speaker, adding up to 270 training patterns; the test comprises a total of 370 utterances. In order to cope with the variable number of cepstrum vectors per articulation, cubic polynomial resampling has been applied to obtain a fixed number of 20 vectors per utterance. Figure 5.6 shows both an original utterance from the training data and its resampled counterpart. The resampled vectors have been concatenated to 240-dimensional feature vectors used in the SRNG training set and test set. Only two prototypes per class have been taken for training. Correct prototypes have been updated by an amount of $\gamma^+ = 0.075$. The adaptation of the wrong prototypes has been done with $\gamma^- = \eta \cdot \gamma^+$, increasing η linearly from 0 to 1 during training. Relevance factors have been updated simultaneously with a learning rate of only $\gamma^{\boldsymbol{\lambda}} = 8 \cdot 10^{-7}$. During 350 training cycles, the initial neighborhood influence of $\sigma = 2$ decayed by a factor of 0.995 to $\sigma = 0.346$. Ten runs took about ten minutes of calculation time and produced an average accuracy of $99.22\% \pm 0.21\%$ on the training set and an accuracy of $97.24\% \pm 0.38\%$ on the test set. More than two prototype vectors did

[3] http://kdd.ics.uci.edu/databases/JapaneseVowels/JapaneseVowels.html

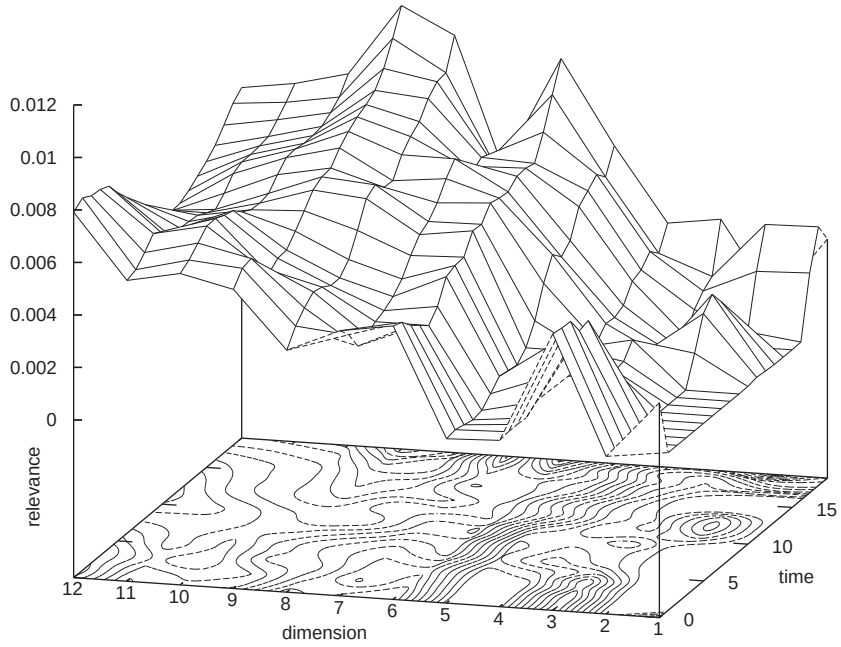


Figure 5.7: Adapted weight factors λ_i for the 'ae' speaker data.

not improve the classification of the test set. The reference accuracies delivered with the data set are 94.1% for a rule-based method and 96.2% for a hidden Markov model, both reported by Kudo et al. [89]. Figure 5.7 shows the profile given by the 240-dimensional relevance vector $\boldsymbol{\lambda}$ reassigned to the 20 feature vectors of dimension 12. Basically, higher dimensions belonging to higher frequencies are more important than lower dimensions, and the articulation endings are more discriminative than their beginnings.

Relevance embedding of the Lorenz attractor

In the domain of nonlinear time series analysis, it is a widely used assumption that a given scalar sequence of observations reflects only one of several differentiable components of an unknown high-dimensional attractor [76]. According to the embedding theorem of Whitney, data from a single component can be used to construct a manifold that is similar to the unknown attractor, exhibiting the same topological properties and also maintaining the temporal structure like periodicities and transient states [93]. Before revisiting the proposal of relevance embedding in an earlier work of the author [144], some preliminaries about the topics are given. For real-life data, a technique suggested by Takens, the so-called 'delay embedding', is commonly used for the manifold reconstruction: equidistant scalar observations $\mathbf{x} = (x_i)_{i=1\ldots n}$ are grouped into d-dimensional vectors according to $\Phi_j(\mathbf{x}) = \{x_j, x_{j+\tau}, x_{j+2\cdot\tau}, \ldots, x_{j+(d-1)\cdot\tau}\}$, $j = 1 \ldots n - (d-1)\cdot\tau$. The construction of these delay coordinates depends on two parameters, the embedding dimension d and the time delay τ. Both values have to be chosen properly in order to yield a good attractor reconstruction [143].

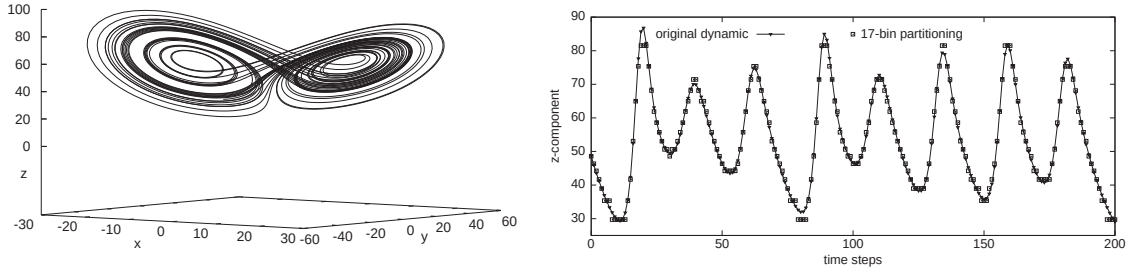


Figure 5.8: Lorenz attractor. Left: phase space portrait. Right: discretized z-component.

The choice of the lag τ is motivated by the desire to maximize the independence of the delay vector components with a minimum value of τ. Some heuristics to determine τ are: the value where the autocorrelation function decays to $1/e$, where it decays to zero, where it falls into the 95% interval of confidence, or where the mutual information exhibits its first minimum [41]. Usually, different lags τ will result from the methods. However, any fixed lag yields equidistant data sampling for the construction of the embedding vectors. Thus, sampling periods may interfere with data periods, leading to aliasing artifacts or Moiré effects. After estimating a reasonable delay τ, the target dimension d is determined by a consideration about topology preservation: as long as there are significant differences for the number of neighbors in ϵ-spheres around the embedded data points when going from the probed dimension k to dimension $k+1$, the dimensionality must be further increased. For example, if one of three variables describing a 3-dimensional spherical manifold is embedded into a 2-dimensional target space, singular point projections will be recognizable by the presence of these *false nearest neighbors*. These accidental projection artifacts and false neighbors will vanish when probing the transition from two to three embedding dimensions, thus making $d = 3$ or above a good dimension choice. Although too large d would not disturb analysis, Ockham demands for the lowest possible dimension.

SRNG provides a data-oriented alternative to the traditional linear delay embedding with three benefits: $\langle 1 \rangle$ the sampling is defined by the metric weighting factors and must not be equidistant, which reduces sampling artifacts; $\langle 2 \rangle$ by dint of the relevance learning, dimensions contribute to the embedding with different influence; $\langle 3 \rangle$ embedding vectors can be constructed according to special requirements, such as the best prediction of the current value with certain components selected from the past.

The alternative embedding is demonstrated for the Lorenz flow which describes a minimum model of atmospheric convection by three differential equations. A phase space portrait for a parameter set leading to deterministic chaos is given in Figure 5.8, but only its z-component will be considered. Relevances are related to those historic influences that maximize the prediction quality for the next sequence value: time window vectors are compared to prototypes representing the current state, and the metric emphasizes dimensions that matter for the explanation of the current state. This way, also the comparison

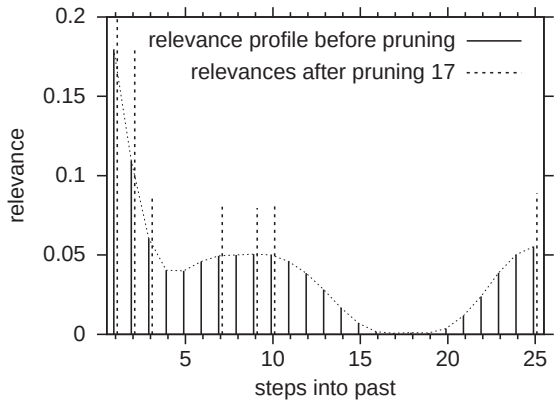


Figure 5.9: GRLVQ relevance profile.

between two data vectors is restricted to the most important dimensions.

In order to deal with a discrete SRNG class prediction, the z-values are discretized by median cuts into 17 states, as shown in the right panel of Figure 5.8, and each of the states is represented by the same number of data. The training of three prototypes per class without neighborhood cooperation and squared Euclidean metric is related to standard GRLVQ training which has been conducted for 5,975 time window vectors $\boldsymbol{z}^i$ of length 25. Further parameters are $\gamma^+ = \gamma^- = 10^{-4}$, $\gamma^{\boldsymbol{\lambda}} = 2.5 \cdot 10^{-6}$, leading to a predictive accuracy of 84% after 50,000 training cylcles. Alternatively, the mean square error could have been calculated by comparing the next continuous time series value with the median value corresponding to the predicted class. As detailed in Strickert, Bojer, and Hammer [144], this error would better account for the fact that the classification result is not just either wrong or right, but that in case of misclassification, often a class is selected of which the corresponding median is neighbored to the desired median in the time series space.

The global relevance profile of observations in the past for a one-step prediction is displayed in Figure 5.9. Apart from the expected importance of the most recent sequence entries, multi-modality arises as a consequence of the quasi-periodicity. A canonic decorrelation length is indicated at the time step τ where the λ_τ exhibits a first minimum. The local minimum at step 5 is associated with a characteristic decorrelation time that is in accordance with a decay of the autocorrelation function to zero at lag $\tau = 6$ and with the minimum of the mutual information found at lag 5, both calculated separately.

As suggested by Bojer, Hammer, and Strickert, relevance learning can be combined with pruning and retraining steps [144]. After a number of training and retraining cycles, the currently smallest relevance value λ_i is set and fixed to 0 for the rest of the training. This pruning proceeds until the classification error becomes unacceptably high. A reason to use this pruning strategy and to rely not only on the built-in relevance weighting, is the desire to explicitly force dimension reduction, even though the target of cost function minimization might be violated then.

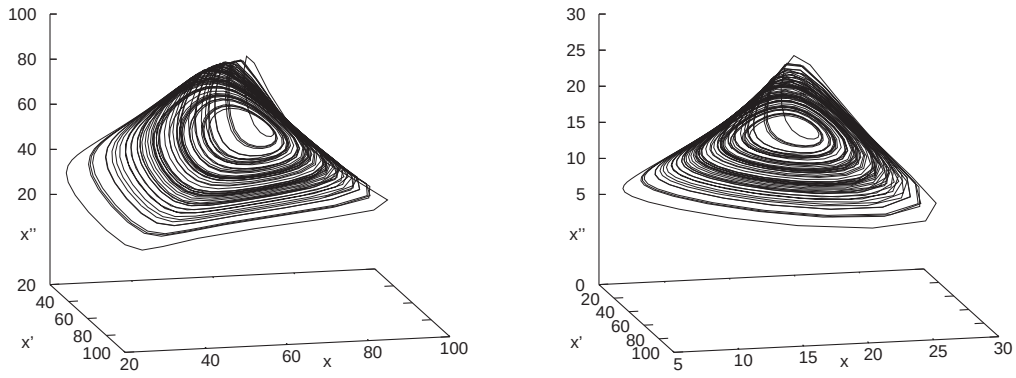


Figure 5.10: Attractor reconstruction. Left: std. embedding. Right: relevance embedding.

For later visualization purposes, a retraining for the three most relevant dimensions of the steps 1,2, and 7 back in the past has been conducted, yielding $\lambda_1 = 0.833, \lambda_2 = 0.077$, and $\lambda_7 = 0.09$ and an accuracy of 76.25% on the training set. With these values, a direct relevance embedding into $\mathbb{R}^3$ is given by $\mathbf{x}^i = (z_1^i \cdot \sqrt{0.833}, z_2^i \cdot \sqrt{0.077}, z_7^i \cdot \sqrt{0.09})$. The operation $x_j^i = z_j^i \cdot \sqrt{\lambda_j}$ undoes the learned implicit metric scaling. More generally, points can be embedded into $\mathbb{R}^d$ in such a way that their unweighted distances are equivalent to the weighted squared Euclidean metric. The required transform for each metric summand is $\lambda_i \cdot (x_i - w_i)^2 = (\tilde{x}_i - \tilde{w}_i)^2 = (g_i \cdot x_i - g_i \cdot w_i)^2$ for $g_i = \sqrt{\lambda_i}$. Of course, a prototype $\boldsymbol{w}$ does not play a definite role at the embedding time, but this mixed notation with the pattern $\mathbf{x}$ refers to the fact that relevances $\boldsymbol{\lambda}$ are related to prototype constellations, and that the vectors $\mathbf{x}^i$ and $\mathbf{x}^j$ to be compared are represented by metric-specific prototypes.

After the preparatory work, the relevance embedding and the standard delay embedding with a lag of $\tau = 6$ are compared by two visual methods, the phase space reconstruction and the recurrence plot technique. The reconstructed 3D phase spaces are given in Figure 5.10. Both plots are similar with respect to the band structure and the rough conic shape, and for reasons of symmetry, the two embeddings fail to capture the original double wing structure of Figure 5.8. Standard embedding exhibits two features, an angular curvature which is a clear embedding artifact, and a pyramidally circumscribed volume that is recognizably larger than the shell shape of the relevance embedding. Since the fractal dimension of the Lorenz attractor is only 2.01, the shell shape of the relevance embedding is much more preferred.

The second qualitative comparison is obtained by means of the recurrence plot technique which basically generates a picture of the quantized distance matrix of all embedding vector pairs [101]. As such, a recurrence plot matrix displays the dynamic independent of the data dimension d. By definition, on the lower left corner the pair furthest back in time is located, on the upper right the most recent pair. A distance less than a given threshold, then called recurrent, leads to drawing a pixel; otherwise the associated coordinate is left blank. Trivially, all points on the diagonal are painted, and the matrix is

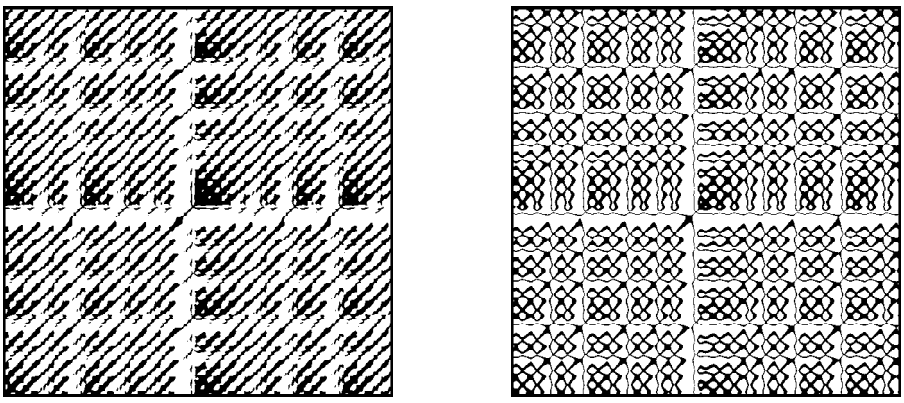


Figure 5.11: Recurrence plots. Left: standard embedding. Right: relevance embedding.

symmetric. Additionally, if the dynamic in the reconstructed phase space gets close to a location where it has already been, potential periodicity is present. Periodicities are indicated as lines parallel to the diagonal, and transient states are found as bands of white; many more details and theoretical foundations are given in Casdagli [21]. An alternative view on the phase space portraits in Figure 5.10 is given by the related recurrence plots in Figure 5.11, where each of the 600×600 pixels represents an embedding vector pair. Both distance thresholds are chosen to display 30% recurrent points. Visual inspection for standard delay embedding essentially displays 24 base periods as parallel lines, some transitions, and a strong recurrence phase in the center of the displayed time. All this is also contained in the relevance embedding, but additionally, the intra-periodic dynamic is detected, represented by fine lines and bubbles.

In the lines above, the potential of relevance embedding has been scratched only, and some promising properties have been demonstrated by the toy example of the Lorenz attractor. A real world example that supports the benefits of this embedding strategy can be found in Bojer, Hammer, and Strickert [144]. Yet, dynamic systems analysis might profit from further studies along the line of relevance embedding.

DNA splice site recognition

So far, standard metrics have been used for dealing with sequential information stored in linguistic data, in speaker utterances, and in the Lorenz time series. Specifically designed sequence metrics are introduced in the following important classification problem: DNA strings shall be separated into two classes, namely into idle intron subsequences and into information-carrying exon strings [18, 117]. This splice site detection task can be tackled by training SRNG. These sites mark transitions from expressed to unexpressed gene regions and vice versa, and they are thus important for the assembly of structural and regulatory proteins that constitute and control a metabolism. Splice site recognition is therefore a topic of interest for the understanding of organisms. Two publicly available data sets are taken for the training task for which a number of comparative results exist.

The `IPsplice` data set contains human DNA data [11]. It consists of 765 intron-exon boundaries, so-called acceptor sites, and of 767 exon-intron transitions which are the donor sites. In addition, 1,654 decoy strings are present, coding no valid transitions. Thus, the problem can be seen as a classification task with the three categories acceptor, donor, and neither. In the data set, sequences with a fixed length of 60 nucleotides are centered around potential splice sites. It is not documented how the decoys have been collected. However, other results from the StatLog project that systematically compare different machine learning approaches are given as a reference [107].

The `C.elegans` data are taken from the Caenorhabditis elegans genome. It is available as nucleotide strings of length 50 in five different data sets for each of the sizes of 1,000 and 10,000 disjoint training examples; each set comes with its own test set containing 10,000 cases [141]. In the following experiments, only the canonic acceptor sites given by the centered adenin-guanin nucleotide (AG) boundary are considered for the separation of true and false acceptor sites. The decoys are centered around arbitrary AG tuples, and there are roughly twice as many decoys as acceptors in the data. Results for this two-class classification problem, obtained by state-of-the-art support vector classifiers with various kernels, have been published in Sonnenburg [141] and Sonnenburg et al. [142].

As indicated in the data preparation Section 3, the four nucleotides A,G,T, and C are encoded as tetrahedron points in a 3-dimensional vector space, thus producing $60 \cdot 3 = 180$ dimensions for the `IPsplice` and $50 \cdot 3 = 150$ dimensions for the `C.elegans` task. Prototype weights $\boldsymbol{w}^i$ are always initialized with small random perturbations around the origin $\mathbf{0}$, and the relevance factors $\boldsymbol{\lambda}$ of the metrics are all equal at the beginning of the training.

`IPsplice`

For `IPsplice`, SRNG is used with eight prototypes for each of the classes acceptor, donor, and neither. The adaptive squared and quartic Euclidean metrics $\mathrm{d}^2_{\boldsymbol{\lambda}^2}$ and $\mathrm{d}^4_{\boldsymbol{\lambda}^2}$ have been considered as well as the exponential memory distance $\mathrm{d}^{\mathsf{SEQ}}_{\boldsymbol{\lambda}}$ (cf. page 35) and the locality improved distance $\mathrm{d}^{\mathsf{LIK}}_{\boldsymbol{\lambda}}$ (cf. page 36). For SEQ, the three exponents for the sequence element distances, the block weighting, and the relevance emphasis have been chosen as $b_{\boldsymbol{w}} = b_B = b_{\boldsymbol{\lambda}} = 2$, and the exponential fading has been assigned to $s = 0.5$. For LIK, the exponents are $b_{\boldsymbol{w}} = 2$, $b_B = 3$, and $b_{\boldsymbol{\lambda}} = 1$; the parameter s in the window scaling function $h_k = 1/(s \cdot |k| + 1)$ is set to $s = 0.5$ and, most importantly, the data window radius for capturing the local correlations is set to $l = 3$, this radius being confirmed by Sonnenburg's findings [141]. In all experiments, an initial neighborhood range of 8 is chosen and multiplicatively decreased by 0.9999 after each training epoch. The remaining training parameters are given in Table 5.3.

Training with random pattern presentation converges after 2,000 epochs. A 10-fold cross-validation has been carried out with randomly compiled training and test sets containing 2,000 and 1,186 data points, respectively. Results are shown in Table 5.4. The upper line contains training set accuracies, the lower line shows the results for the test

Metric	γ^+	γ^- from 0 to ...	$\gamma^{\boldsymbol{\lambda}}$
$d^2_{\boldsymbol{\lambda}^2}$	0.004	$0.075 \cdot \gamma^+$	10^{-8}
$d^4_{\boldsymbol{\lambda}^2}$	0.001	$0.075 \cdot \gamma^+$	$5 \cdot 10^{-9}$
$d^{\mathsf{SEQ}}_{\boldsymbol{\lambda}^2}$	0.004	$0.05 \cdot \gamma^+$	10^{-8}
$d^{\mathsf{LIK}}_{\boldsymbol{\lambda}}$	0.004	$0.5 \cdot \gamma^+$	10^{-8}

Table 5.3: Training parameters for the `IPsplice` data set.

sets. The four leftmost SRNG test accuracies are of interest. Further results for alternative machine learning methods given in Table 5.4 are taken from the StatLog project [107] and from Sonnenburg et al. [142]. The results comprise a decision tree approach (C4.5), a radial basis network (RBF) producing the best result in StatLog, a powerful stochastic hidden Markov model approach (HMM), and three SVM approaches. The SVMs use kernels which have been explicitly designed for splice site recognition: the locality improved kernel for SVM is comparable to the similarity measure LIK which is used in the SRNG experiments. The two other kernels used, the Fisher kernel (FK) and the tangent vector of posterior (TOP) kernel, have been derived from statistical models, and they combine established statistical modeling with the discriminative power of SVMs [152]. Since SVMs and HMMs are used as binary classifiers, three different models for donor, acceptor, and neither class recognition must be trained separately; therefore, errors taken from Sonnenburg et al. being the sums of all three models are worst case estimates [142].

All in all, the SRNG results are very good: especially the LIK metric provides better results than any of those from the StatLog project. The exponential memory metric SEQ does not yield particularly good results, though, and it is even outperformed by the Euclidean type metrics which are direction insensitive. Although the SRNG results cannot be directly compared to Sonnenburg's binary classifiers, his SVM results are supposed to be slightly better. However, the advantages of SRNG over SVM are $\langle 1 \rangle$ that SRNG is a natural multiclass method, $\langle 2 \rangle$ that the final number of prototypes is small, and $\langle 3 \rangle$ that prototypes can be easily interpreted as data representatives. Additionally, the compactness of the model make it feasible to train also large data sets. To give a hint about that: on a Pentium III computer with 700 MHz, only ten minutes are needed to

$\mathsf{SRNG}_{\mathrm{LIK}}$	$\mathsf{SRNG}_{\mathrm{EUC}\wedge 2}$	$\mathsf{SRNG}_{\mathrm{EUC}\wedge 4}$	$\mathsf{SRNG}_{\mathrm{SEQ}}$	RBF	C4.5	$\mathsf{SVM}_{\mathrm{LIK}}$	$\mathsf{SVM}_{\mathrm{TOP}}$	$\mathsf{SVM}_{\mathrm{FK}}$	HMM
97.1 ± 0.3	96.0 ± 0.3	96.7 ± 0.4	95.9 ± 0.4	98.6	96.0	—	—	—	—
96.5 ± 0.3	95.6 ± 0.5	95.7 ± 0.4	95.2 ± 0.3	95.9	92.4	96.3	94.6	94.7	94

Table 5.4: Accuracy (in %) of different methods for the `IPsplice` data. Upper row: training set. Lower row: test set. The SVM and HMM results are taken from Sonnenburg et al. [142], others are taken from StatLog [107].

train and test a SRNG model on the `IPsplice` with the quartic Euclidean distance, and about one hour for the given LIK distance with radius 3. The utilization of a total number of only $3 \cdot 8 = 24$ prototypes yields compact, fast, and accurate classifiers. Contrary to that, the large number greater than 1,000 of support vectors is responsible for the long training times of SVMs.

Another SRNG feature to be taken from the result Table 5.4 is the generalization ability. The large difference between the training and the test accuracy of C4.5 and RBF shows a substantial overfitting. A better performance of the SRNG method is indicated by smaller differences, thus confirming again the good data generalization properties of the SRNG classifier. In future experiments, the LIK metric, already producing excellent results, could be replaced by an even better metric for including higher order correlations.

One major advantage of the considered adaptive metrics is the possibility to obtain dimension relevance profiles which allow insights into the classification. Figure 5.12, upper panel, depicts the relevance factors $\boldsymbol{\lambda}$ within a local window achieved by training with the weighted squared Euclidean metric, the center panel shows the relevance profile obtained for the exponential memory metric SEQ, and the panel at the bottom shows the relevances for the LIK similarity. The mean values of relevance triplets are plotted, because these triplets correspond to the importance of the single nucleotides encoded in three dimensions. As expected, the region around a potential splice site is characterized by particular importance. This location has been identified as important by all three metrics. Obviously, both characteristic dinucleotides GT and AG are relevant, because they heavily determine the distinction of donors, acceptors, and decoys. Furthermore, the profiles show that around the splicing location the left side is a bit more weighted than the right one, a result which is in good correspondence with the known high relevance of the pyrimidine rich region for the splicing process. Strong border effects are displayed for the SEQ metric: accidental overweighting at the sequence start for missing historic context is compensated by small relevance factors, and missing values beyond the scope of the sequences are balanced by large factors. The interpretability of the relevance profile is complicated by this naturally sigmoidal shape, although it is — except for the scaling — similar to the Euclidean profile. For LIK, the bias is mild: the relevances increase linearly within LIK radius at the borders, which is induced by the missing data for LIK windows at the start and at the end of the DNA strings. Comparing the unaffected inner region with the relevances for the Euclidean metric, the LIK profile is smoothed by the averaging influence of neighbored symbols on the distance calculation.

In addition to the visual inspection of relevance profiles, the obtained classifiers can be evaluated analytically. As discussed in Section 4.3 on page 24, the receiver operation characteristic (ROC) plot describes the quality of binary classifiers; ROC curves are frequently used in computational genomics. Since the SRNG classifier produces hard Voronoï boundaries, the cut-points for the decision lines are virtually fixed.

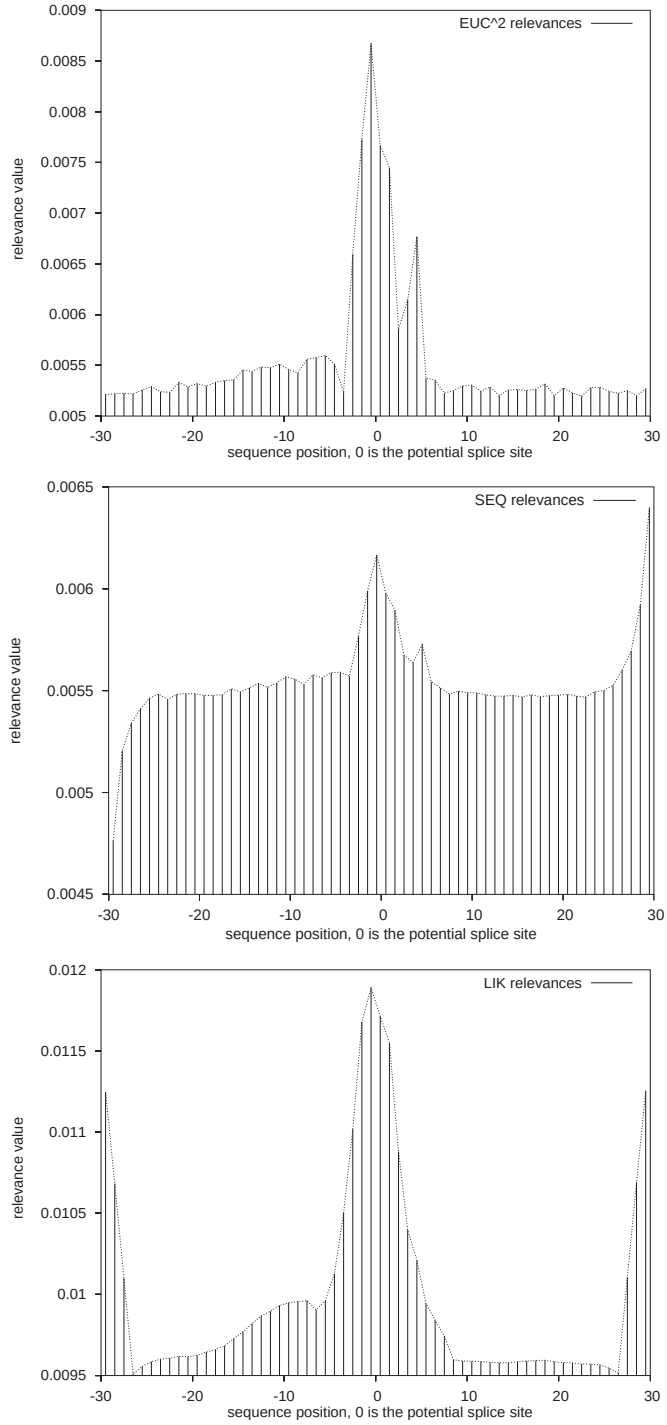


Figure 5.12: Nucleotide relevance profiles for the classification of the `IPsplice` data set. Top: adaptive squared Euclidean metric. Middle: SEQ metric. Bottom: LIK metric.

In order to be able to use ROC analysis, a measure related to the margin size is computed, based on the distances of the closest correct and the closest wrong prototype to the presented pattern:

$$v = \frac{\mathrm{d}^+_{\boldsymbol{\lambda}} - \mathrm{d}^-_{\boldsymbol{\lambda}}}{\mathrm{d}^+_{\boldsymbol{\lambda}} + \mathrm{d}^-_{\boldsymbol{\lambda}}} \ .$$

This classification measure $v \in [-1; 1]$ has been chosen as the test variable of the ROC statistics. The hard SRNG classification maps all data points with values of $v \leq 0$ to the correct class. Other values of specificity and sensitivity can be achieved by varying the classification boundary within the interval.

For the splice site recognition task the classifier's specificity refers to the percentage of decoys correctly identified relative to the total number of decoys in the data set. The sensitivity refers to the percentage of correctly predicted splice sites, compared to all available splice sites, no matter if donors or acceptors. Figure 5.14 displays the general recognition characteristics of splice sites on DNA strings showing a plot of the true positive rate (sensitivity) versus the false negative rate (1− specificity) together with the reference diagonal. The corresponding curves, from which the moderate SEQ metric has been excluded, indicate very high classifier accuracies by their asymptotic alignments to the axes. For the `IPsplice` data set, only a subset of the whole ROC plane $[0; 1] \times [0; 1]$ is depicted in Figure 5.13 in order to make the differences between the single graphs visible. It is possible to achieve the highest values $(0.93, 0.99)$ of specificity and sensitivity for the weighted Euclidean metric at a cut-point of $v = 0$ on the test set; an even better pair of $(0.97, 0.98)$ is obtained for LIK, also at $v = 0$. This finding confirms that SRNG performs best on what it has been designed for, namely for the Voronoï decision boundaries corresponding to the cut-point parameter $v = 0$.

`C.elegans`

The DNA for the Caenorhabditis elegans nematode has been thoroughly sequenced. The non-splice sites, chosen similar to true splice sites, have been added to the data sets by Sonnenburg in a well documented way [141]. Five data sets with results known for Sonnenburg's models are used for comparison of SRNG with SVMs and HMMs.

A preliminary study for the present `C.elegans` data set has showed that the LIK metric, being slow but successful for `IPsplice`, was only as accurate as the weighted quartic Euclidean metric. Sonnenburg also reports a significant loss of the LIK classification performance for `C.elegans` subsets containing more than 100 patterns in comparison to other kernels [141]. Therefore, only two distance metrics have been further studied for SRNG, the weighted quartic Euclidean metric $\mathrm{d}^{\mathsf{EUC}\wedge 4}_{\boldsymbol{\lambda}}$ and the weighted block Euclidean distance $\mathrm{d}^{\mathsf{BEUC}}_{\boldsymbol{\lambda}}$, introduced as a special case of LIK in Section 5.1 on page 36, which accounts for data blocks but not for correlations between the blocks.

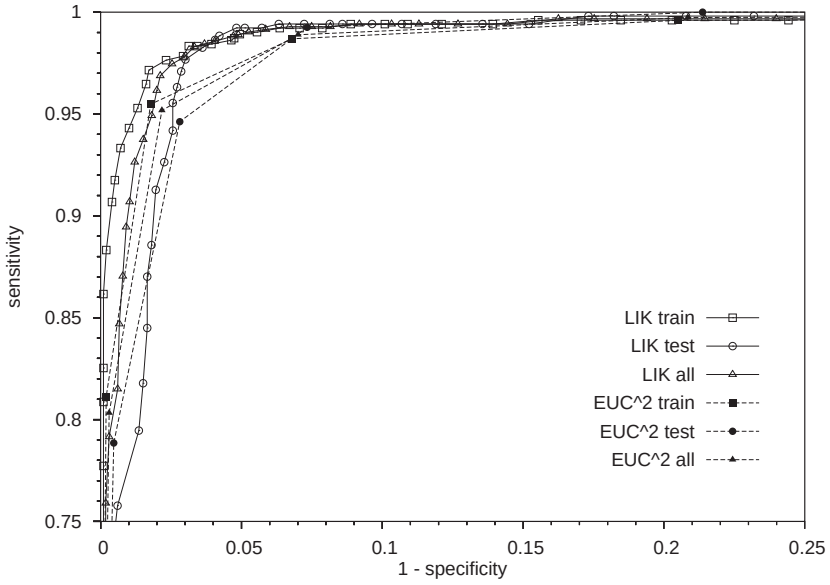


Figure 5.13: ROC curve for the `IPsplice` data set. EUC^2: weighted Euclidean measure; LIK: locality improved similarity metric. Both high specificity and sensitivity can be achieved simultaneously.

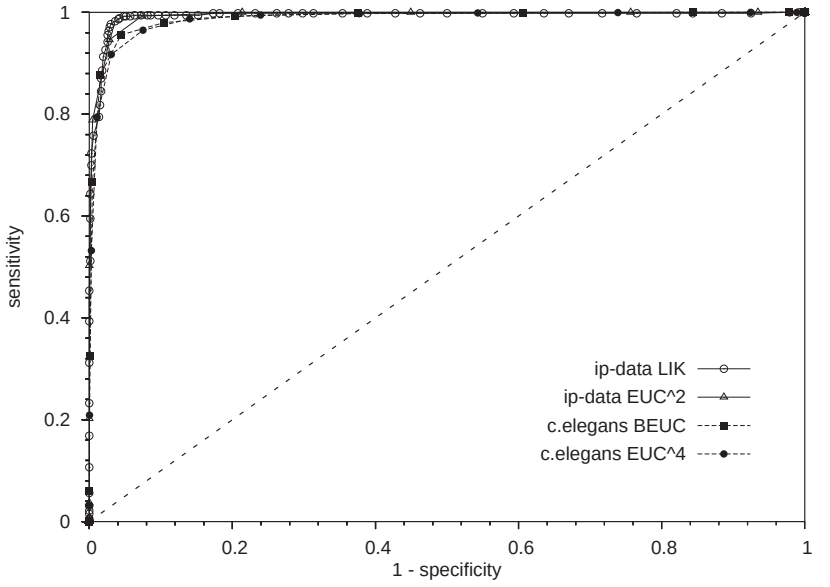


Figure 5.14: Overall ROC curve for the `IPsplice` and `C.elegans` data computed on the test sets for two weighted Euclidean EUC^{2,4} metrics, the block Euclidean BEUC, and the locality improved LIK. The diagonal indicates the random guessing classifier.

$\langle d_{\boldsymbol{\lambda}}^{\mathsf{EUC}\wedge 4}\rangle$ For the weighted quartic Euclidean, only two prototypes are necessary per class; more prototypes do not improve the generalization error. The following model parameters are chosen: the learning rate for the correct prototypes is $\gamma^+ = 0.005$, and the repelling γ^- rate of wrong prototypes is increased from 0 to the small value of $0.05 \cdot \gamma^+$; the learning rate for the relevance terms is $\gamma^- = 3 \cdot 10^{-8}$; an initial neighborhood range of 1 is multiplicatively decreased in each training cycle to a final value of 0.002. For the small data sets with 1,000 points, 600 epochs have been trained, while 60 epochs are sufficient for the large sets with 10,000 points. The achieved classification accuracy is $94.602\% \pm 0.003\%$ for the small data sets and $94.811\% \pm 0.003\%$ for the large sets.

$\langle d_{\boldsymbol{\lambda}}^{\mathsf{BEUC}}\rangle$ Using the block Euclidean metric improves these results. However, some data preparation is necessary to capture correlations between DNA dinucleotides with the block distance. Pairs of adjacent nucleotides are compiled from the original DNA strings; these strings are expanded to all pairs in the secondary and tertiary diagonal of the upper triangular recombination matrix of the sequence elements; for example, a given substring GCAT is expanded to the pairs GC, GA, CA, CT, and AT. Thus, the scope of pairing is restricted to radius 2 for the reasons of computational feasibility. Coding the $4^2 = 16$ possible pairs by blocks of 15-dimensional vectors, this yields data vectors with $(50-1+50-2)*15 =$1,455 dimensions. These vectors are processed with the block Euclidean metric for blocks of size 15, with an intra-block exponent of $b_w = 2$ and an inter-block exponent of $b_B = 4$. Optimum generalization has been achieved with five prototypes. Learning rates for training are $\gamma^+ = 0.015$ for correct prototypes and $\gamma^- = 0.01 \cdot \gamma^+$ for the closest incorrect prototype, and $\gamma^{\boldsymbol{\lambda}} = 3 \cdot 10^{-8}$ for the relevance adaptation. The initial neighborhood range of 4 is kept constant during training. Training takes 600 epochs for the smaller sets and 100 epochs for the larger sets. The accuracy on the test sets is $95.17\% \pm 0.22\%$ for the sets with 1,000 training examples and $95.68\% \pm 0.17\%$ for the sets with 10,000 examples.

A full comparison chart can be found in Table 5.5. SRNG results are competitive to the results achieved by the SVM with locality improved kernel. Statistical models combined with SVM, however, show a better performance, in particular for the large data sets containing 10,000 points. The standard deviation of statistical models is a bit higher, because they can exploit very specific information in different data sets; in contrast to these findings, the few SRNG prototypes have not many degrees of freedom for finding a minimum of the cost function; therefore, the convergence is very stable, and very small standard deviations are obtained. The performances of the SRNG classifiers are expressed by ROC curves that can be directly compared to those of the SVM models. In the left panel of Figure 5.15, the most reliable SRNG classifier can be identified as the block Euclidean distance for the sets with 10,000 training patterns. The right panel shows results of different SVM approaches of which the TOP kernel with large data sets performs best. Differences between the left and the right curve plots are, as revealed only after logarithmic scaling, in favor of SVM.

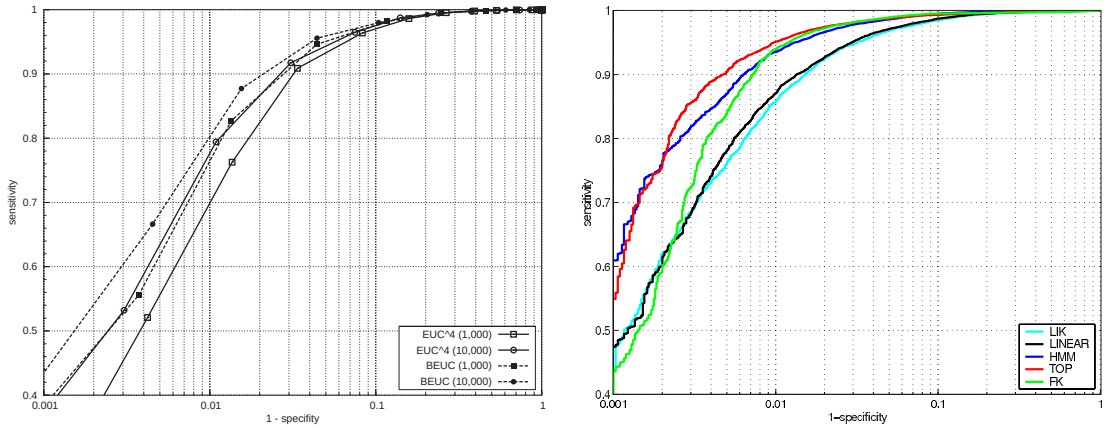


Figure 5.15: ROC curve for the `C.elegans` test data set. Left: for SRNG with the weighted quartic similarity measure EUC^4, and the block Euclidean BEUC. Right: Curve plots taken from [141] for SVMs (with kind permission of Sonnenburg).

Similar to the special kernel design of SVM for splice site recognition, SRNG could be equipped with a statistical similarity measure to improve the results; the block Euclidean metric with the preprocessed data points into that direction. However, the training time for statistical models is quite demanding. Training SRNG models takes only about one hour on a Pentium III (700 MHz) computer for all five sets of 10,000 data points for the quartic Euclidean, and ten hours for the high-dimensional block Euclidean metric. A benefit of SRNG is its sparsity: the maximum of only ten prototypes makes training fast, and the classification of large data sets is possible in only a few seconds. This is different for SVM, for which a number of more than 1,000 support vectors are reported, making classification time about two orders of magnitudes slower than SRNG.

Since compact models of SRNG would also be obtained for prototypes that represent statistical models, problem-adopted fast and accurate online splice site detectors could be realized with SRNG by using these alternative prototypes. In general, further research is necessary to shed light on the performance of SRNG for prototypes that represent adaptive data models which are more general than the interpolating identity functions of the data that have been investigated here.

method	$\text{SRNG}_{\text{EUC}\wedge 4}$	$\text{SRNG}_{\text{BEUC}}$	HMM	SVM_{LIK}	SVM_{TOP}	SVM_{FK}
1,000	94.6±0.003	95.2±0.15	97.2± 0.1	94.8±0.1	95.4±0.4	96.5±0.2
10,000	94.8±0.003	95.7±0.09	97.4± 0.2	96.1±0.2	97.7± 0.1	97.5±0.3

Table 5.5: Test set accuracies (%) by different methods for the `C.elegans` data set.

Part III

Unsupervised SOM-type learning

Chapter 6

Self-Organizing Maps (SOM)

Experience is not what happens to you.
It is what you do with what happens to you.

Aldous Huxley

The self-organizing map (SOM) is a neural learning architecture proposed by Kohonen as a similarity-based clustering method for unlabeled data [80]. Self-organization refers to the Hebbian adaptation of data prototypes which are interacting by neighborhood cooperation. In contrast to the dynamic data-topology of neural gas (NG) introduced in Section 4.1 on page 19, the SOM neighborhood is given by a fixed low-dimensional regular structure onto which high-dimensional input data are mapped during training.

Data visualization and abstraction are related to dimension reduction and to prototype representation. Both issues are tackled simultaneously by means of the SOM [82]. The original motivation of Kohonen was to suggest a simplified concept of biological self-organization in the brain [80]: stimuli originating from different receptors, like the tactile impressions from finger tips and toes, are mapped to different locations on the folded two-dimensional neural cortex, but similar inputs, like from the sensors of a hand, are mapped to neighbored cortex locations. For multi-dimensional features, such as the location on the skin, the local temperature and the pressure intensity, a reduction to a lower-dimensional surface takes place, this way utilizing only a small number of nerve cells. The SOM implements this kind of topology preserving mapping; a concise description is provided by Ritter [126].

The SOM method is related to the supervised LVQ in the way that it is also a prototype-based method, but unlike LVQ, the SOM reference units are arranged with lateral neighborhood competition in a neural output grid onto which the input vectors are mapped. Like LVQ, the SOM algorithm provides a similarity-based Hebbian prototype update mechanism, but in contrast to LVQ, it is essentially designed for handling unlabeled data. This unsupervised processing makes the SOM particularly interesting for data mining, because

many raw data have been recorded and made available; for example, document collections and data bases from the Internet can be visualized and browsed by similar topics with the SOM variants WebSOM and HSOM [91, 114]. Information mining requires clustering, grouping and analysis of possibly high-dimensional data contained in large repositories. By means of the similarity-based neighborhood relationship of the SOM, the data complexity is reduced to a visualizable grid of neurons where similar inputs are grouped together.

The main purpose of the SOM is the low-dimensional clustering and visualization of unlabeled high-dimensional data, useful for a large number of problems. Many applications of the SOM can be found in the book edited by Oja and Kaski [113]; the given examples comprise financial data analysis and market studies, image processing and pattern recognition, image retrieval, document retrieval, medical investigations, document classification with hierarchical maps, and navigation in large data bases. Additional studies in that book refer to SOM-based optimization, and to nonlinear component analysis. More work about the SOM and its usage can be found in the book by Allinson et al. [1].

It is important to mention that the notion of a 'self-organizing map' or a 'SOM network' is wider than just 'the SOM'; the *the SOM* refers to Kohonen's specific idea of implementing a high-dimensional mapping onto a low-dimensional grid, while *a SOM network* possesses the properties of self-organization and vector quantization, which apply to the SOM as well as to the NG approach, and also to the supervised LVQ. This chapter gives an overview of the SOM and of its recent variants for unsupervised SOM-type learning, before extensions to sequence processing will be discussed in subsequent chapters.

6.1 The basic SOM algorithm

The essential constituents of the SOM are $\langle 1 \rangle$ prototype neurons with a memory of input vectors they have specialized on, $\langle 2 \rangle$ a definition of the target space and the neuron neighborhood given by connections in a low-dimensional grid for realizing local interactions, and $\langle 3 \rangle$ a metric to evaluate the similarity of the prototypes to the input stimuli. The policy 'map similar inputs to similar outputs' is realized by neural competition for specialization: the most similar winning neuron adapts strongest and, additionally, informs its grid neighbors to also rotate — just a little weaker — into the direction of the input; with even lower influence, the same request is passed on to the not yet visited neighborhood, and so on, until the amount of adaptation has decreased to a negligible value.

Algorithm 5 describes the training procedure in a generic manner. The parameters are: $\langle 1 \rangle$ the learning rate $\gamma < 1$, possibly decreasing during training to ensure convergence, $\langle 2 \rangle$ a neuron neighborhood function $D(i, j) = D(j, i)$ to measure the shortest grid distances between neurons i and j in the target space, and $\langle 3 \rangle$ an update strength function $f \leq 1$ that calculates the amount of update for the current neuron, depending on its distance to the winner in the target space under the influence of a control parameter σ which is cooled down during training; typically, f is a bell-shaped Gaussian $f = \exp(-D^2/\sigma^2)$.

Algorithm 5 Basic SOM

repeat
  chose randomly a pattern $\mathbf{x}$
  $k \leftarrow \arg\min_i \{ d(\mathbf{x}, \boldsymbol{w}^i) \}$ { neuron with smallest distance d to pattern }
  for all neurons j **do**
    $\boldsymbol{w}^j \leftarrow \boldsymbol{w}^j + \gamma \cdot f(\sigma, D(k,j)) \cdot (\mathbf{x} - \boldsymbol{w}^j)$ { γ, f, σ, D: see text }
  end for
until no more major changes

Before learning starts, the neuron arrangement must be defined, for example, as a rectangular 2D-lattice that defines the neighborhood connectivity of the neurons. Further SOM initialization is quite simple; the learning rate has to be set to a starting value, say $\gamma = 0.1$, and the $\boldsymbol{w}^i$ are initialized with random values located within the range of the training data. Alternatively, neurons can be initialized more expensively by PCA as the centroids of the rectangular tiles belonging to the tessellation of a linear data subspace which is obtained by the projection of the data to the two most important eigenvectors of the data covariance matrix.

Generally, two phases can be observed during training: a contraction of the neurons to the center of gravity of the data for the initially large neighborhood cooperation f, and an unfolding of the neural lattice within the data according to the data density distribution. Usually, the relationship between the data density and the prototype density is described by the generic power law $P(\mathbf{x}) \propto P(\boldsymbol{w})^\alpha$. The exponent with optimum information transfer is given by $\alpha = 1$. For SOM, however, the exponent α depends on the intrinsic data dimension $\tilde{d}$ and the SOM lattice dimension, typically leading to $\alpha < 1$. This discrepancy of prototype density and data density, i.e. $\alpha \neq 1$, is referred to as *magnification*. With neighborhood cooperation, only the case of one-dimensional lattices has been identified as $\alpha = \tilde{d}/(\tilde{d}+2)$ by Ritter [124]. For vanishing neighborhoods, thus for a degeneration of SOM to winner-takes-all vector quantization, an overview of magnification factors for different learning architectures is given by Villmann and Claussen [160]. Local update control strategies have been proposed by Bauer, Der, and Herrmann for obtaining optimally trained SOMs with $\alpha \approx 1$ for arbitrary dimensions [67], and specific magnification strategies have been used by Merényi and Jain for feature enhancement and detection in spectral images [106]. In many other practical applications, however, a discrepancy between data density and prototype density can be accepted.

A potential problem of the SOM occurs in case of an incompatibility between the input data and the representation space. This mismatch may result from using inappropriate metrics or from dealing with multi-modal and high-dimensional data that cannot be

faithfully mapped to a low-dimensional neuron lattice. As a consequence, the lattice unfolding in the input space might produce anomalies like self-penetration and thus unwanted ambiguous mappings. Villmann discusses the issues of topology preservation and magnification control for grid architectures in his PhD thesis [159] and in subsequent work [161]. Van Hulle reuses the idea of magnification control based on information transfer, and he extends it by locally adaptive radial basis functions to obtain faithful data representations and topographic maps [155].

So far, a detailed theory for convergence and stability of SOM training following an optimization of a clear energy function has been made available by Cottrell and Fort only for a mapping of one-dimensional data to an open chain of neurons [28, 29]. No such theory exists for the typically considered high-dimensional situation where the necessary factorization into a product of one-dimensional problems cannot be ensured. Ritter et al. provide further analysis of convergence [127]: it has been showed that in the special case of high-dimensional data from a finite discrete set the training dynamic obeys a clear cost function minimization. Heskes proposes modifications of the original dynamic to generally obtain a cost function based SOM training [69]. For computational ease and for maintaining the straight-forward interpretation of the Hebbian learning mechanism, the original update, tried and tested in applications many times, will be taken as a basis for extensions here.

6.2 SOM variants and developments for sequence processing

The versatility and the simplicity of the SOM algorithm have triggered many developments ranging from associative memory models to feature detectors and route optimizers and other extensions; a number of them can be found in Kohonen's book [82] and in the book of Seiffert and Jain [135]. In the following, the focus is put on variants corresponding to sequence processing architectures. First, some historic approaches will be outlined, then the two major directions of recent SOM developments will be examined that separately take into consideration different metrics and alternative neuron lattice structures.

SOM in the past

A remarkable extension of the standard SOM which can be used for sequence processing is the adaptive subspace SOM (ASSOM) [84]. This algorithm is very close to the original formulation, but it generalizes the view on the data in the way that neurons match templates $F_{\boldsymbol{\varphi}_j}(\mathbf{x})$ for linear subspaces of the data generated by basic transformations. The original SOM is obtained for the identity model $F_{\boldsymbol{\varphi}_j}(\mathbf{x}) = \mathrm{id}_{\boldsymbol{\varphi}_j}(\mathbf{x})$ that yields $\boldsymbol{\varphi}_j$ which act as usual data prototypes $\boldsymbol{w}^j$. A variant for sequence processing is realized, if the parameters $\boldsymbol{\varphi}_j$ represent coefficients of Fourier transformed data windows of a fixed length; in this case, the best matching neuron is in best resonance with the input pattern in the time-shift invariant frequency space. More generally, operators with free parameters $\boldsymbol{\varphi}_j$

can be associated with the neurons for representing adaptive data models with Hebbian learning dynamic [94]. However, computationally intensive operations might be required for the update, or similar results might even be obtainable by means of a priori data transformation.

In order to avoid time window techniques which possibly induce the disadvantages of high dimensionality and redundancy, integrated signals can be processed instead. In combination with the SOM, response filter models that evaluate the exponentially weighted sum of the input data have been studied by several authors. The Temporal Kohonen Map (TKM) of Chappell and Taylor [23], and the Recurrent SOM (RSOM) of Varsta et al. [157] and Koskela et al. [86] both belong to this class of 'leaky integrator' SOM networks. While TKM uses the temporal integration of exponentially weighted errors only for the determination of the best matching neuron, RSOM exploits, for the winner determination, the norm of the vector calculated by the exponentially integrated correction vectors, and the direction of the integrated vector is applied to the prototype update. In both approaches, only the implicit filter response constitutes the temporal context. Further comparison of TKM and RSOM can be found in Varsta et al. [158], and a general taxonomy of different types of memory, such as the leaky integration, is given by Mozer [109].

Miikkulainen complements natural exponential fading by integrating a retention mechanism into the proposed SARDNET architecture, a network of SOM type with Sequential Activation, Retention, and Delay: after its winner selection each neuron remains inactive until its activity status has decreased to a state ready for reactivation, this way realizing disjoint trajectories of neuron activations on the map [74, 108].

SOM with non-standard lattices

The quantization accuracy of the SOM can be improved by using neuron lattices different from the original two-dimensional rectangular or hexagonal structures. For example, under the assumption of exponential context diversification for sequences of increasing lengths, a grid with power law neighborhood scaling at varying radii on the grid is an inappropriate choice. To tackle this problem, Ritter has proposed an hyperbolic SOM (HSOM) [125], where the neurons are located at the mesh nodes of a regularly triangulated hyperbolic plane. The neuron indexation can still be calculated quickly with only two coordinates, and local projections of the hyperbolic to the Euclidean plane allow to browse the map interactively on screen. Large text collections have been successfully processed by this architecture [114].

In order to account for special needs of the data topology in a more flexible manner, other SOM variants implement data-driven adaptive lattices. Prominent methods are Fritzke's Growing Grid and Growing Cell Structures [43, 44, 45], and the growing hypercubical output spaces with adaptive dimensionality of the target grid proposed by Bauer and Villmann [8]. A slightly different direction is taken by Rauber, Merkl, and Dittenbach who have designed a growing hierarchical SOM (GHSOM) that generates individual

SOMs for each level of data granularity and which are arranged in a tree-like manner, running from an abstract root map to more concrete stages; successful applications are the clustering of country descriptions and the automatic ordering of a digital library [122].

As introduced in the LVQ section on page 19, the neural gas NG method makes it possible to get rid of a neural grid to allow a data-driven topology of the optimum neuron constellation. The rank function $D(i,k) = \mathrm{rnk}_{\mathbf{x}}(i)$ required by NG can be plugged into the generic form of the SOM Algorithm 5 to determine the number of neurons that are closer to the presented pattern $\mathbf{x}$ than neuron i is. Although the framework of SOM is used, this approach does no longer refer to a grid target space, but it describes geometric relationships in terms of the dynamic nearest neighbors in the input space [99]. As a matter of fact, Martinetz and Schulten have shown that the original data topology is represented by the trained NG prototypes, if all closest pairs are connected by lines, this way inducing a subgraph of the complete Delaunay graph of the data [100]. In contrast to the SOM, NG does not yield a reduction of the input space dimension to a simplified target space. The NG ranking operation requires a time complexity of $\mathcal{O}(m \log m)$ for sorting the m neurons during training, not only m distance comparisons like the SOM. However, since no topology restriction is given on the output space, the vector quantization is more accurate than for the SOM.

Sequential data may lead to exponential context increase depending on the length; therefore, a hyperbolic grid structure is desired for sequence learning. If accurate context quantization is more important than visualization and fast update, an extension of the neural gas architecture to temporal data is promising.

SOM with non-standard metrics

An obvious alteration to the standard algorithm is to use metrics $d(\cdot,\cdot)$ other than the Euclidean distance for winner determination. These metrics not only allow different data evaluation characteristics, but they are also a handle for processing non-vectorial input data like character strings, if appropriate prototype adaptation schemes are defined too. For example, in order to process data sets that contain sequences of variable lengths, Somervuo uses the time warping method discussed in Kruskal [88] which is based on dynamic programming to compare sequences for online SOM learning of word similarities and bird songs [139, 140].

A structurally different model proposed by Finish researchers implements a semi-supervised SOM with adaptive metrics for incorporating auxiliary information [116]. As previously discussed in the text, LVQ with cost function is a supervised counterpart to the semi-supervised SOM: for both methods a global distortion measure is minimized during the prototype adaptation, and the proofs of convergence are given for semi-supervised SOM by Kaski [78] and for SRNG by Hammer et al. [61].

Graepel, Burger, and Obermayer propose an alternative approach to the SOM: they use a non-differentiable cost function which implicitly provides a topographic mapping from

the data space onto the neuron grid [52]. Due to the craggy error surface, the cost function is minimized by deterministic annealing. This way of finding optimum parameters fits into the framework of expectation maximization which usually requires long computing times. The authors show in small data compression-reconstruction experiments a good signal to noise ratio of their method called soft topographic vector quantization (STVQ).

Related to adaptive metrics is the online determination of irrelevant dimensions that can be successively ignored during training; this is done by Bojer et al. by pruning those dimensions which contribute only little to the average influence on global features, such as changes in topology [15].

A new perspective taken on the problem of temporal processing is obtained when the nature of sequences is appropriately accounted for: the currently observed state is explained by the most recent sequence element and its context. Importantly, context refers again to context to establish a recursive model. Existing recursive SOM models are briefly outlined, before recursive formulations for both HSOM and NG will be presented.

Recently proposed methods with explicit context representation are the Recursive SOM of Voegtlin (RecSOM) [166, 167] and the SOM for Structured Data (SOMSD) by Hagenbuchner et al. [54]. The RecSOM model refers back to the last time step with high accuracy: each neuron is given an additional vector that characterizes the average activation of all other neurons one time step before becoming the winning neuron. Winner selection depends on both the pattern match and a good conformance of the stored context with the current data context. A similar but much more reduced context notion is provided by SOMSD. The essential idea of neuron context specialization implies that the number of paths leading to a succeeding state is small; otherwise, it would not be specialized. Thus, the learning process is accompanied by minimizing the context variability of a potential winner neuron. This justifies that only the previous peak activation — the last winner — is referred to in SOMSD. Actually, this variant can easily be implemented as an extension of a SOM with regular neuron grid, because only the average winner coordinate, usually a two-dimensional lattice index, needs to be stored in order to obtain a simple local context model. More generally, SOMSD is designed for processing directed acyclic graphs; for example, in case of binary trees, this requires the storage of not only one predecessor, but the back-reference to the left and the right ancestor by applying the above recursive addressing scheme.

6.3 Context evaluation

For sequence processing models, the temporal quantization error is a useful performance measure to express the capability to detect context structure. This value refers to the quality of representation for sequence elements located t steps back in time. The error is expressed by the average standard deviation of the given sequence and the winner unit's receptive field, both considered at the temporal position t.

More formally, a sequence processing neural map is assumed as well as a sequence $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ in which $\mathbf{x}^j \in \mathbb{R}^d$ denotes the element of time step j. For neuron i, the number of time steps being selected as winner is $N_i = |\{j \mid I_j = i\}|$, where I_j is the index of the winner neuron when sequence element $\mathbf{x}^j$ is presented. Then, the mean activation of neuron i for step t in the past is

$$a_i(t) = \frac{1}{N_i} \sum_{j \mid I_j = i} \mathbf{x}^{j-t} .$$

The *temporal quantization error* of neuron i for time step t in the past is defined as

$$e_i(t) = \left(\frac{1}{N_i} \sum_{j \mid I_j = i} \left\| \mathbf{x}^{j-t} - a_i(t) \right\|^2 \right)^{1/2} .$$

The map quantization error for all m neurons is the average at step t in the past:

$$e(t) = \frac{1}{m} \sum_{j=1}^{m} e_i(t) .$$

Thus, $e(0)$ just denotes the standard error for the currently presented pattern.

In the case of symbol sequences, the temporal quantization error can be made more crisp and expressed as the *temporal receptive field*. For neuron i this is the set of all sequences which lead to unambiguous winner selection of this neuron. In practice, neuron competition restricts the potentially infinite length of the temporal scope to a length above which other neurons are specialized on the tailing context. Therefore, the effective receptive field of a winner is determined from the training data as the longest string in the intersection of all sets of strings that occur in the data as historic contexts of the neuron. Voegtlin calls the length of the effective string the depth of the receptive field or temporal capacity [167].

Related to the determination of temporal receptive fields is the context reconstruction from trained maps. Instead of using training data for the extraction of the scope of the neurons' responsibilities, introspection can be applied to obtain the learned precedence relationships from the map only. Basically, context reconstructions are obtained by a recursive calculation of the conditional probabilities of symbols — represented by previous neurons — by applying the Bayes rule to the probabilities of neuron contributions of past stages to the current state. Such kind of context backtracking will be illustrated in experiments later. However, it cannot be expected that state transitions coded in the maps coincide perfectly with the transition probabilities of the training data. The reason for this is that both the map architecture and the implemented context model limit the backtracking and the reconstruction accuracy. Magnification effects are known for SOM architectures, and magnification may lead to extraction artifacts which still emphasize the observation that context becomes exponentially uncertain when going back in time.

A technique that refers to context analysis and to rule extraction rather than to context evaluation is the transfer of Ultsch's U-matrix method to recursive maps [153, 154]. This approach, recently presented by Blohm, is not canvassed here, but the interested reader is referred to Blohm's bachelor thesis, where the extraction of finite state machines and rule determination for continuous sequence elements are discussed in detail [12].

6.4 SOM extension proposals

Two directions of research are investigated in the following, the first will focus on an extension of SOMSD from rectangular grids to more general lattices, the second will study an alternative context model to combine neural gas with sequence processing.

⟨1⟩ The standard rectangular lattice of SOMSD will be replaced by a triangular planar graph structure that is more general than the original neuron assembly. Euclidean grids can be reobtained by connecting each neuron with six neighbors, while seven or more neighbors produce lattices with hyperbolic neighborhood branching. In the Euclidean case, the relationship between a neuron's number of neighbors and a given radius is determined by a power law, whereas in the hyperbolic case, the neighborhood grows exponentially with the radius. The generalized SOMSD architecture will be studied for Euclidean and hyperbolic grid topologies. This context as well as SOMSD context refers to the average grid position of the previous winner neuron; hence, this context storage is essentially based on an efficient addressing of grid locations.

⟨2⟩ In order to get rid of the restrictions caused by the topology of the target lattice and a specific indexation scheme, another context encoding strategy will be developed. The proposed Merging SOM (MSOM) will overcome the dependence on orderly indexable output grid topologies for addressing neurons. It does so by referring to the two properties of the last winner rather: location and context [145, 147]. The proposed back-reference to both weight and context is already part of the context model, which results in a recursive context definition. MSOM shares the efficiency of SOMSD due to this compressed context representation. As for TKM, the MSOM context is expressed in the weight space and appears as fractal code [158], but since context is made explicit and with separate adaptation, a better control is obtained, leading to a substantially improved context representation. Investigations on the proposed encoding are carried out on the basis of data sets from different domains.

Chapter 7

The SOM for Structured Data (SOMSD) with lattice topology

My opinion, my conviction, gains immensely in strength and sureness
the minute a second mind has adopted it.

Novalis

In this section, the focus is put on the compact representation of temporal context by using a back-reference to the previous winner location in the map, complementary to the usual weight vector for the currently presented pattern. This type of recursive context model was first proposed as the self-organizing map for structured data (SOMSD) which has been used for processing binary tree structures by taking into account the current node and its recursive left and right descendants [54]. The formulation of SOMSD will be motivated by the discussion of a general framework for the unsupervised processing of structured data. Then, a concrete description of the original SOMSD will be briefly given. Finally, an extension from rectangular neuron lattices to more general triangle meshes will be proposed to account for the exponential context diversification specific to sequence processing.

7.1 Unifying notation for structure processing SOM

As discussed in the previous chapter, various architectures, like TKM, RSOM, or RecSOM exist for temporal data processing. A tentative approach to formulating these methods in a uniform taxonomy for unsupervised spatio-temporal connectionist networks (STCN) has been proposed by Barreto, Araújo, and Kremer [6, 7]. Their four identified constituents are: a network initialization function, a function for adapting the trainable parameters, a state computation function, and an output function. Temporal context is realized by integrating the network output into the state computation. In this sense, the proposed

taxonomy for unsupervised STCN is very general and it can be used for expressing a number of unsupervised architectures in terms of a unified algorithm where method-specific operations are replaced by generic function calls.

At the same time and independent of STCN, a framework for the unsupervised processing of structured data, the general SOM for structured data (GSOMSD), has been proposed by Hammer, Micheli, Sperduti, and Strickert [55]. From the beginning, this framework has been designed for the recursive processing of directed acyclic graphs; therefore, the special case of sequences is automatically covered by this approach. Graphs can be expressed by the STCN formalism too, but they are not mentioned by Barreto, Araújo, and Kremer. More recently, Hammer, Micheli, Sperduti, and Strickert provide interesting theory about convergence properties and context representation capabilities for TKM, RSOM, RecSOM, and SOMSD expressed in the GSOMSD notation [56].

Ingredients for the GSOMSD are ⟨1⟩ a metric for data comparison, ⟨2⟩ a formal representation of trees and a recursive metric for their pairwise comparison, ⟨3⟩ a number of neurons as tree nodes, each containing a data prototype and a fixed-size set of pointers to descendant neurons, and ⟨4⟩ a mapping of node activations to a formal tree representation.

Particularly, the SOMSD for clustering directed acyclic graphs can be expressed by the notation of GSOMSD. As will be discussed in detail, the sequence processing version of SOMSD can be expressed in terms of GSOMSD by: ⟨1⟩ a squared Euclidean metric, ⟨2⟩ a degenerated tree with left (or right) descendants only, indexed by integer value coordinates in a neural grid, compared by means of another squared Euclidean metric, ⟨3⟩ neurons in an indexable grid, containing a weight and a context reference to the previous winner, and ⟨4⟩ a maximum activation search for the determination of winner neurons. This illustrates how the generic formalism can match the requirements of a specific model. Many more examples and proofs can be found in the article of Hammer et al. [55]. In the following, the work will stick to specific methods, starting with an explicit description of SOMSD.

7.2 SOMSD dynamic

Originally, SOMSD has been proposed for processing trees with a fixed fan-out k [54]; the special case $k = 1$ addresses sequential data. Neurons N_ι are arranged on a rectangular q-dimensional grid. Lattice coordinates are used for the neuron indexation

$$\boldsymbol{\iota} = I_{\mathsf{N}_\iota} = (\iota_1, \ldots, \iota_q) \in \{1, \ldots, m_1\} \times \cdots \times \{1, \ldots, m_q\} \ , \ m_i \in \mathbb{N}$$

of a total number of $m = m_1 \cdot \ldots \cdot m_q$ neurons. Each neuron N_ι contains a weight vector $\boldsymbol{w}^{\boldsymbol{\iota}}$ for the state representation. Additionally, k vectors $\boldsymbol{c}_1^{\boldsymbol{\iota}}, \ldots, \boldsymbol{c}_k^{\boldsymbol{\iota}} \in \mathbb{R}^q$ store the context of a tree related to the coordinates of the winner neurons for the previously processed k subtrees. The winner index for a given tree t with root label $\mathbf{x}$ and subtrees $t_1, \ldots, t_k$ is recursively defined by

$$I(t) = \arg\min_{\boldsymbol{\iota}} \{\beta \cdot \|\mathbf{x} - \boldsymbol{w}^{\boldsymbol{\iota}}\|^2 + \alpha \cdot \| I(t_1) - \boldsymbol{c}_1^{\boldsymbol{\iota}}\|^2 + \ldots + \alpha \cdot \| I(t_k) - \boldsymbol{c}_k^{\boldsymbol{\iota}}\|^2 \} \, .$$

The first summand weighted by β refers to the usual SOM comparison of a given pattern $\mathbf{x}$ and the prototype weight vector $\boldsymbol{w}^\iota$. The summands weighted by α refer to the quality of context matching, depending on the difference between the coordinates of the previous winners for the subtrees t_i and the currently stored context representations $\boldsymbol{c}_i^\iota$. The parameters α and β control the influence of context and pattern on the winner determination. For the tree leaves, no context is available; this situation is handled by the representation of the empty tree $I(\xi) = (-1, \ldots, -1) \in \mathbb{N}^q$. Thus, starting at the leaves, the winner is recursively computed for an entire tree.

Hebbian learning is applied for SOMSD. The vectors $(\boldsymbol{w}^\iota, \boldsymbol{c}_1^\iota, \ldots, \boldsymbol{c}_k^\iota)$ attached to the winner N_ι are moved into the direction $(\boldsymbol{w}, I(t_1), \ldots, I(t_k))$ by γ fractions after each recursive processing step. The neighborhood is updated into the same direction with the learning rate γ scaled by a decreasing function of the radius around the winner.

Hagenbuchner et al. have demonstrated the applicability of SOMSD for unsupervised clustering of pictures described by trees [54]. During training, a similarity-based clustering has emerged with respect to the descriptive constituents of the pictures. The objects have been properly arranged according to the similarity of sub-categories; thereby, the empty tree representation $I(\xi) = (-1, \ldots, -1) \in \mathbb{N}^q$ leads to an orientation of the map unfolding. Within the clusters, a further differentiation with respect to the involved attributes could be observed, which reflects the clustering characteristic of standard SOM.

In the following, the tree fan-out is set to $k = 1$ for sequences, and the winner computation is reduced to matching both the currently processed sequence element and the recursively expressed historic context. Without loss of generality, one weight influence factor can be replaced by $\beta = (1 - \alpha)$ to control the influence of the two summands on the winner computation. For continuous sequences, the empty tree representation $I(\xi)$ is only required to express the beginning of the data stream, when the recent past is unknown.

7.3 SOMSD with alternative lattice topology

The original architecture is based on a regular rectangular neuron lattice, for which the introspective back-reference to previous winner is established by ordered grid coordinates. For the practical purpose of visualization, the dimension of the target grid is usually set to two. Here, this standard Euclidean grid topology will be extended to more general neuron triangle meshes [146]. Preferably, the alternative target space should remain planar and convex for the visualization and for an efficient update; also, the grid should provide a possibly small number of pathways along the edges to get from one neuron to another: this supplement enables the well-behaving training update discussed below. The new meshing provides a branching with power law growth for grids with 6-neighbor connections per neuron, this way emulating the properties of an Euclidean target space; for 7 and more neighbors, though, the meshing yields properties of the hyperbolic space, characterized by an exponential neighborhood increase as the function of the radius.

The advantage of the hyperbolic topology becomes apparent for sequential data. In the case of words over an alphabet $\{\mathbf{a}^1, \ldots, \mathbf{a}^M\}$ with an average number of p successors for each symbol, the context size is roughly an exponential function p^l of the word length. While a Euclidean target grid with its power law neighborhood scaling is not suited for a faithful representation of the context branching, the sequential data topology is better preserved by a hyperbolic neuron grid. Its adequate data representation has been demonstrated by Ontrup and Ritter [114] for texts from a movie data base with a non-recursive hyperbolic SOM. However, sequences shall be processed in the following.

Graph Lattice

Here, an alternative topological setup is considered by using a graph description for the neuron grid: connections are realized by assigning each neuron a set of direct neighbors. For the experiments, a grid generator is used to obtain lattices with a circular triangle meshing around a center neuron and a fixed neighborhood degree of $n > 5$. The number of neurons k_i to be added at map radius l is recursively computed by $k_l = (n-4) \cdot k_{l-1} - k_{l-2}$ with $k_0 = 0$, $k_1 = n$. Starting with one center neuron at radius $l = 0$, k_l is just the number of neurons required to generate fully occupied rings at a distance of radius l from the center. If during the incremental graph construction the desired total number of neurons leads to only a sparse filling of the outer ring, a best symmetric solution is taken.

In order to integrate the temporal data characteristic, SOMSD adapts not only the weight vector but also the neurons' representations of the previous winner locations during training. For the original rectangular Euclidean planar grid, a tuple of real values can be easily updated into the direction of the last center of maximum activity. For the proposed triangular mesh structure the adaptation is different. In this case, the continuous representation, necessary for addressing any location within the neural grid during the incremental Hebb-learning, is given by two ingredients: a set of three adjacent nodes and a tuple of coordinates referring to a point inside the spanned triangle. Ritter provides direct 2D-locations for the neurons in the tessellated hyperbolic plane $\mathbb{H}^2$ and he is therefore able to calculate node distances efficiently with hyperbolic geometry [125]. Here, in order to deal with more general 2D-triangulated manifolds, an alternative way is taken for the neuron grid distance calculation.

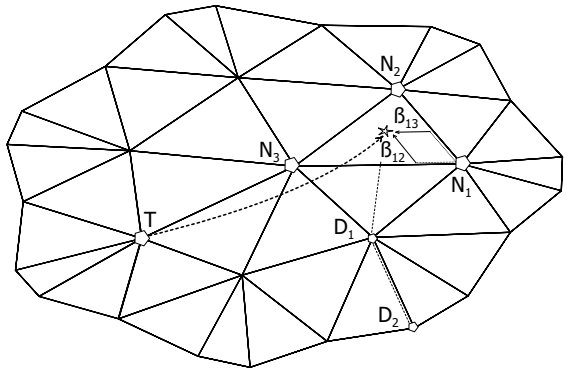


Figure 7.1: Hyperbolic self organizing map with context. Neuron n refers to the context given by the winner location in the map, indicated by the triangle of neurons N_1, N_2, and N_3, and the precise coordinates $ß_{12}$,$ß_{13}$. If the previous winner is D_2, adaptation of the context along the dotted line takes place.

Figure 7.1 illustrates neuron localization by the example of a small hyperbolic grid for the context of neuron T. The context of this neuron is given by a two-dimensional triangle coordinate (β_{12},β_{13}) within the triangle of neurons N_1, N_2, and N_3. The distance $d'_\triangleright$ on the grid between any other winner neuron, say D_2, and the triangle is easily obtained: the shortest length of the three possible paths is computed by adding to the value looked up in the neuron distance matrix the extra path length from the respective corner to the point addressed inside the triangle. The required entries of the distance matrix are calculated once during the grid construction, reflecting the minimum number of edges that have to be visited to get from one node to another. In a scenario with hyperbolic neighborhood, these distances are mixed with locally Euclidean distances for coordinates inside the triangles — a mixture which is supposed to produce no harmful bias, because local projections of the hyperbolic space to the Euclidean space exist. For the update procedure, it is convenient to store, as additional matrix entry, an adjacent edge that points as a signpost into the direction of a shortest path. The range of $d'_\triangleright$ is normalized to $d_\triangleright$ by scaling with the inverse maximum grid distance. In Figure 7.1, the non-normalized distance $d'_\triangleright(D_2,(N_1,N_2,N_3,\beta_{12},\beta_{13})) = |\overline{D_2N_1}| + (\beta_{12}^2+\beta_{13}^2 + \beta_{12}\,\beta_{13})$ is somewhat greater than 2 under the assumption of locally Euclidean trigonometry. Although the nodes N_2 and N_3 do not appear in the distance term, they are implicitly present by defining β_{12} and β_{13}. However, the dotted update path of the context to the target D_2 is different from the one assumed for distance calculation. Since adaptation along the edges alone would impose the restrictions of locally one-dimensional trajectories, an interpolation scheme is given that allows a further degree of freedom, without adulterating the underlying distance too much: the direction inside a triangle patch is taken relative to the distance contribution of the two closest corners that are located on the target path; for this reason, the shown setup with $|\overline{D_2N_1}| = |\overline{D_2N_3}|$ leads to taking a mid-way across the edge $\overline{N_1N_3}$.

Winner computation

The distance d of neuron j to the current pattern $\mathbf{x}^t$ is computed by

$$d_j(\mathbf{x}^t) = (1-\alpha)\cdot\|\mathbf{x}^t - \boldsymbol{w}^j\|^2 + \alpha\cdot d_\triangleright(\mathbf{c}^t, \boldsymbol{c}^j)$$

in which, as explained above, $d_\triangleright$ is the grid distance between the currently stored context given by the 5-tuple $\boldsymbol{c}^j$ and the back-reference to the winner $\mathbf{c}^t$ in the previous time step. The influence of both the context $\boldsymbol{c}^j$ and the weight representation $\boldsymbol{w}^j$ on the distance computation is controlled by the parameter α. Its choice also effects the determination of the winner neuron, for which the obtained distance must be minimum. Since at the beginning of training nothing is known about the temporal structure, this parameter starts at $\alpha = 0$, resulting in the standard SOM without context. During the training, it is decreased to an application dependent value that mediates between the externally presented pattern and the internally gained model about historic contexts.

Training

Training is done by the presentation of an entry $\mathbf{x}^i = (x_1^i, \ldots, x_d^i)$, the determination of the winner k, and the usual Hebbian style update of weight and context. The update is applied to all neurons in the breadth first search graph around the winning neuron, with an adaptation amount being a decreasing function of the grid distances to the winner. Hence, weight $\boldsymbol{w}^j$ is updated for sequence entry $\mathbf{x}^i$ by

$$\Delta \boldsymbol{w}^j = \gamma \cdot \mathrm{h}_{jk} \cdot (\mathbf{x}^i - \boldsymbol{w}^j) \, .$$

The learning rate γ is typically exponentially decreased from a value of 0.1 to 0.005 during training. The amount of neighborhood cooperation h_{jk} describes the spatially decreasing influence of the winner k to the current neuron j; usually, the Gaussian bell function $\mathrm{h}_{jk} = \exp(-\mathrm{d}_{\triangleright}(\mathsf{N}_j,(\mathsf{N}_k,\mathsf{N}_k,\mathsf{N}_k,0,0))^2/\sigma^2)$ with slowly decreasing σ is taken. Context update is analogous: the current context, expressed by the neuron triangle corners and coordinates, is moved by a γ-fraction of the shortest path's distance along such a path towards the previous winner location by adapting $\mathfrak{B}_{12/13}$, a moving operation which possibly induces the exchange of $\mathsf{N}_{1/2/3}$ by grid neighbors to maintain currently valid triangles.

Stern accuracy in inquiring, bold imagination in describing,
these are the cogs on which history soars or flutters and wobbles.

Thomas Carlyle

7.4 SOMSD experiments

Three types of experiments shed light on different aspects of SOMSD. For comparison with existing results from Voegtlin, ⟨1⟩ the real-valued Mackey-Glass time series and a ⟨2⟩ discrete binary automaton have been studied [167]; an additional experiment with the ⟨3⟩ Reber grammar over an alphabet of seven discrete symbols has been conducted in order to overcome the limitations of scalar sequence entries.

Mackey-Glass time series

The first task is to learn the dynamic of the real-valued chaotic Mackey-Glass time series $\frac{\mathrm{d}x}{\mathrm{d}\tau} = bx(\tau) + \frac{ax(\tau-d)}{1+x(\tau-d)^{10}}$ shown in Figure 7.2, using $a = 0.2$, $b = -0.1, d = 17$. This setup has been chosen to make a comparison to Voegtlin's results possible [167]. Three types of maps with 100 neurons have been trained: ⟨1⟩ standard SOM with a 6-neighbor map but no context, ⟨2⟩ a map with six neighbors combined with context (SOMSD), and ⟨3⟩ a 7-neighbor map providing a hyperbolic grid with context utilization (HSOMSD). Each run has been computed with $1.5 \cdot 10^5$ presentations starting at random positions within the Mackey-Glass series, using a sampling period of $\Delta t = 3$; the neuron weights have been initialized with uniform noise taken from $[0.6; 1.4]$. The context influence parameter α has been increased from 0 to 0.03. The learning rate γ is exponentially decreased from 0.1 to 0.005 for weight and context update. Initial neighborhood cooperation is $\sigma = 10$, cooled down to $\sigma = 1$ during training.

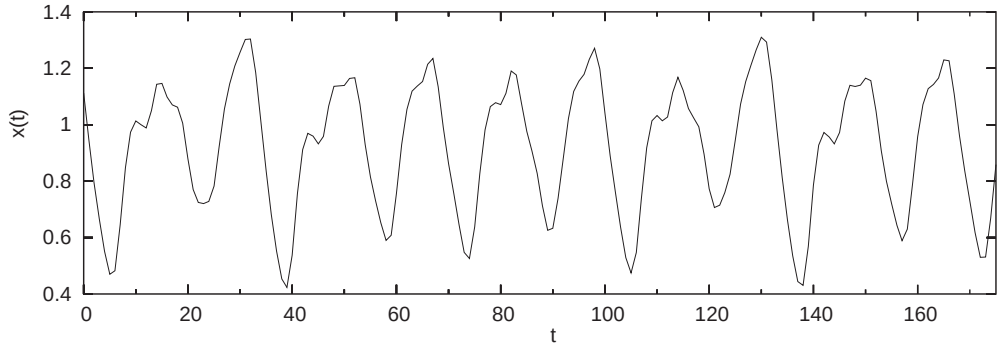


Figure 7.2: Mackey-Glass time series.

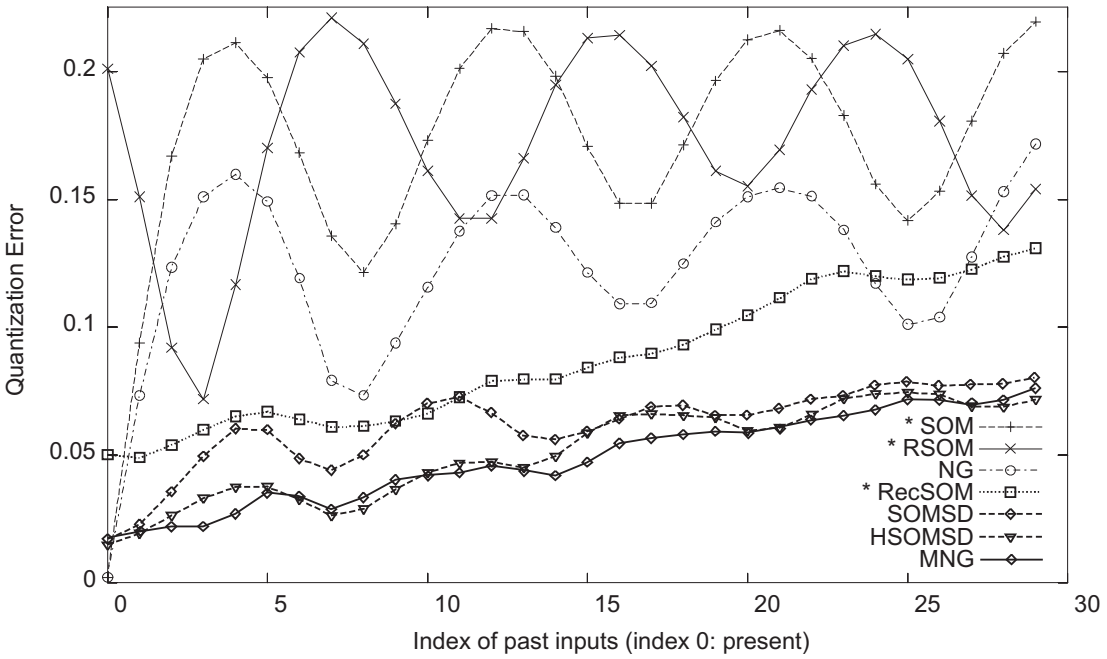


Figure 7.3: Temporal quantization errors of different model setups for the Mackey-Glass series. Results indicated by * are taken from [167]. Method MNG will be explained later.

Figure 7.3 shows the temporal quantization errors for different models with parameters chosen to produce optimum results. At the present instant, that is for time step $t = 0$, the errors for the pattern quantization are small. For increasing t, i.e. for going further back into the past, temporal quantization errors remaining small indicate that the neurons do not only specialize on the current entry but also on the historic context.

Especially the neural gas (NG) and the standard SOM catch the eye: while they yield best results for $t = 0$, large cyclic oscillations are observed for $t > 0$, corresponding to the quasi-periodicity of the training series. Since both methods do not account for context information, these quantization results can be seen as upper bounds for temporal models with and without grid topology: certainly, the data optimum topology of NG allows smaller quantization errors than the SOM. Obviously, the restricted context of RSOM does not yield a significant improvement of the temporal quantization error; however, the displayed error fluctuations are anti-cyclic to the original series.

RecSOM leads to a much better quantization error than RSOM and NG. The increase of the quantization error is smooth and the final values after 29 time steps is still better than the default given by standard SOM. In addition, almost no periodicity can be observed for RecSOM. SOMSD and HSOMSD further improve the results: only little periodicity can be observed, and the overall quantization error increases smoothly for the past values. These models are superior to RecSOM in the quantization task while requiring less computational power. HSOMSD allows a slightly better representation of the immediate past compared to SOMSD due to the hyperbolic topology of the lattice structure that matches

better the characteristics of the input data. The MNG graph will be discussed later in Section 8.4 on page 98, when the merging neural gas is introduced as a learner that is driven by data topology, implementing a compact back-reference to the previous winner inspired by SOMSD. Another example for beneficial application of SOMSD to multivariate continuous data will be given for comparison with MNG in Section 8.4 on page 103 for the `B-1991` time series.

Binary automata

In the second experiment given by Voegtlin, a discrete **0**/**1**-sequence generated by a binary automaton with transition probabilities $P(\mathbf{0}|\mathbf{1}) = 0.4$ and $P(\mathbf{1}|\mathbf{0}) = 0.3$ shall be learned.

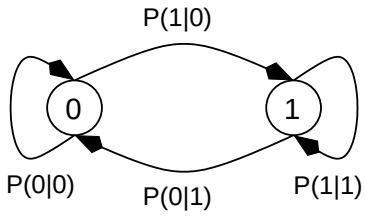


Figure 7.4: Binary Automaton.

By means of Figure 7.4, the other probabilities are gained: $P(\mathbf{0}|\mathbf{0}) = 1 - P(\mathbf{1}|\mathbf{0}) = 0.7$ and $P(\mathbf{1}|\mathbf{1}) = 0.6$; additionally, the temporally stationary solution for the frequency of **0** is given by the Master equation $\partial P(\mathbf{0})/\partial t = P(\mathbf{1}) \cdot P(\mathbf{0}|\mathbf{1}) - P(\mathbf{0}) \cdot P(\mathbf{1}|\mathbf{0}) = 0$ for $P(\mathbf{0}) = 4/7$, inducing $P(\mathbf{1}) = 3/7$. For discrete data, the specialization of a neuron can be defined as the longest sequence that still leads to an unambiguous winner selection. A high percentage of active specialized neurons indicates that temporal context is learned by the map. In addition, one can compare the distribution of specializations with the original distribution of strings generated by the underlying probability.

Context specialization

Figure 7.5 shows the specialization of a trained HSOMSD. Training has been carried out with $3 \cdot 10^6$ presentations, increasing the context influence exponentially from 0 to 0.06 and providing 10^6 test iterations for the determination of the receptive field. The remaining parameters have been chosen as in the Mackey-Glass experiment. Putting more empha-

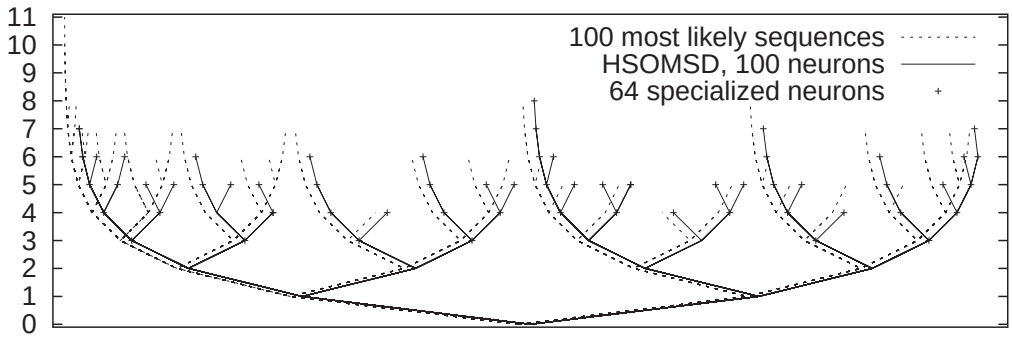


Figure 7.5: Receptive fields of a HSOMSD compared to the most probable subsequences of the binary automaton. Left branches denote symbol **0**, right branches denote **1**.

sis on the context leads to a smaller number of active neurons representing rather longer strings but covering only a small part of the total input space. If a Euclidean lattice is used instead of a hyperbolic neighborhood, the quantizer performances differ only slightly. This indicates that the representation of binary symbols and their contexts in their 2-dimensional product space hardly benefits from exponential branching. In the depicted run, 64 of the neurons express a clear profile, whereas the other neurons have fallen idle during training. A likely reason for this is the training dynamic: the neighborhood cooperation σ is decreased, but the increased context influence α leads to a positive feedback and thus to a preferential treatment of the already good prototypes. However, the specialization displayed in Figure 7.5 corresponds nicely to the 100 most characteristic sequences of the probabilistic automaton as indicated by the graph. Unlike RecSOM,presented in [167], also neurons at interior nodes of the tree are expressed for HSOMSD. These nodes refer to transient states which are represented by corresponding winners in the network.

RecSOM, in contrast to SOMSD, does not rely on the winner index only, but it uses a more complex representation: since the transient states are spared, longer sequences can be expressed by RecSOM.

Context development

The simple two-state structure of the binary automaton is used for a further investigation of the SOMSD context development: snapshots of neuron activities during SOMSD training can be found in Figure 7.6. A line of 101 neurons is the target architecture, in which the context is represented as an interpolating back-reference to a neuron index between 0 and 100. Neural activity is plotted in the upper panel for the first ordering without context: only two neurons at both ends of the neuron chain specialize on the symbols **0** and **1**, reflecting the original symbol frequencies. By paying more attention to the context for the winner calculation, more neurons are getting involved and activated. Finally, a fractal-like iterative bisecting focus on so far unused regions of the neuron space can be observed.

A posteriori map analysis

In addition to the examination of neuron specialization, the representation capability of the map can be characterized by comparing the transition statistics of the input symbols with the learned weight and context relations. While the current symbol is coded by the winning neuron's weight, the previous symbol is represented by the average of weights of the winner's context triangle neurons. The obtained two values — the neuron's state and the average state of the neuron's context — are clearly expressed after training: most neurons specialize on values very close to **0** or **1**, and only few neurons contain values in an indeterminate interval $[\frac{1}{3}; \frac{2}{3}]$. Results for the reconstruction of three automata can be found in Table 7.1. The left column indicates in parentheses the number of clearly expressed neurons and the total number of neurons in the map. Obviously, the automata can be

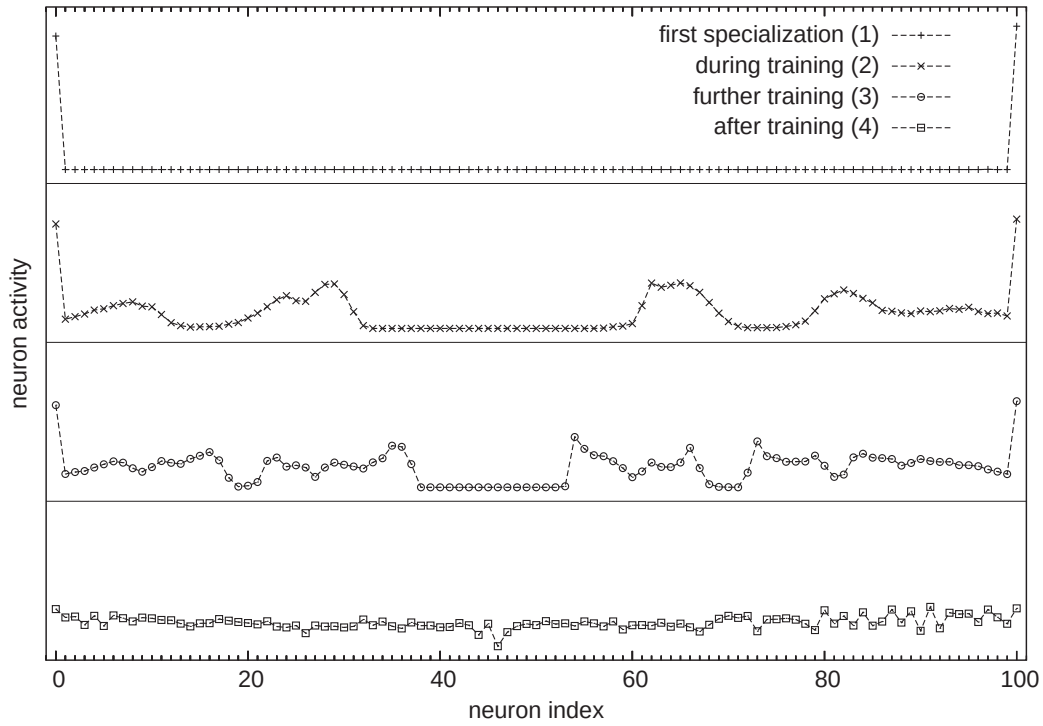


Figure 7.6: Snapshots of 101 neuron activations for SOMSD with open neuron target chain.

Type	$P(\mathbf{0})$	$P(\mathbf{1})$	$P(\mathbf{0}\|\mathbf{0})$	$P(\mathbf{1}\|\mathbf{0})$	$P(\mathbf{0}\|\mathbf{1})$	$P(\mathbf{1}\|\mathbf{1})$
Automaton 1	$4/7 \approx 0.571$	$3/7 \approx 0.429$	0.7	0.3	0.4	0.6
Map (98/100)	0.571	0.429	0.732	0.268	0.366	0.634
Automaton 2	$2/7 \approx 0.286$	$5/7 \approx 0.714$	0.8	0.2	0.08	0.92
Map (138/141)	0.297	0.703	0.75	0.25	0.12	0.88
Automaton 3	0.5	0.5	0.5	0.5	0.5	0.5
Map (138/141)	0.507	0.493	0.508	0.492	0.529	0.471

Table 7.1: Results for the extraction of binary automata with different symbol transition probabilities, recursively calculated by the Bayes rule for the previous neuron activation probabilities. The extracted probabilities clearly follow the original ones.

well reobtained from the trained maps with respect to the overall symbol frequencies and their transition statistics derived by means of Bayes law for the activation probabilities of the previous winner prototypes. These results show that the temporal dependencies are faithfully captured by the maps.

Reber grammar

In a third type of experiment, a more structured symbolic sequence was generated by the Reber grammar [123] shown in Figure 7.7.

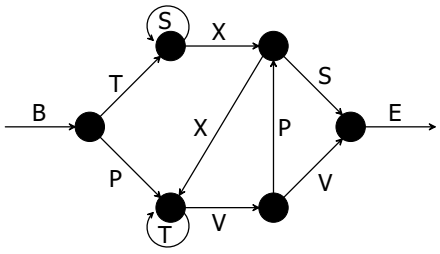


Figure 7.7: Reber grammar state graph.

The seven symbols have been encoded in a 6-dimensional Euclidean space, analogous to the corners of a tetrahedron in 3-dimensional space. The concatenation of randomly generated words led to sequences with $3 \cdot 10^6$ and 10^6 input vectors for training and testing. The chosen map radius of 5 corresponds to $m = 617$ neurons on a hyperbolic grid with seven neighbors per neuron. For the initialization and the training, the same parameters as in the Mackey-Glass experiment were used, except for an initially larger neighborhood range of 14, accounting for the larger map, and the context influence was increased from $\alpha = 0$ to $\alpha = 0.2$ during training. 338 neurons, i.e. 55%, develop a specialization for Reber strings. The average length of the represented words is 7.23 characters. Specialized neurons separate into strict clusters with sector shape on the circular grid, topologically ordered by the last character. In agreement with the grammar, the letter **T** takes the largest sector on the map in Figure 7.8. In comparison, the Euclidean grid produces the polymorphic coherent patches shown in Figure 7.10.

Similar to learning tasks for the binary automata, the map representation was analyzed by the reconstruction of the trained data by backtracking all possible context sequences of each neuron up to length 3. As a result, only 118 of 343 combinatorially possible trigrams are realized. In a ranked table the most likely 33 strings cover all attainable Reber trigrams. These are shown in the log-probability Plot 7.9: a leap between entry number 33 and 34 is observable, corresponding to the valid triplet **TSS** and the invalid word **XSX**, respectively. This is another indication for the presence of the Reber characteristic coded in the map. The correlation of the probabilities of Reber trigrams and their relative frequencies found in the map is $r^2 = 0.75$. An explicit comparison of the probabilities of valid Reber strings can be found in Figure 7.11. The values deviate from the true probabilities, in particular for cycles of the Reber graph, such as consecutive letters **T** and **S**, or the **VPX**-circle. This effect is due to the magnification factor different from 1 for the SOM; magnification gets even more emphasized during recursion, which should be systematically investigated in future work on sequences processing.

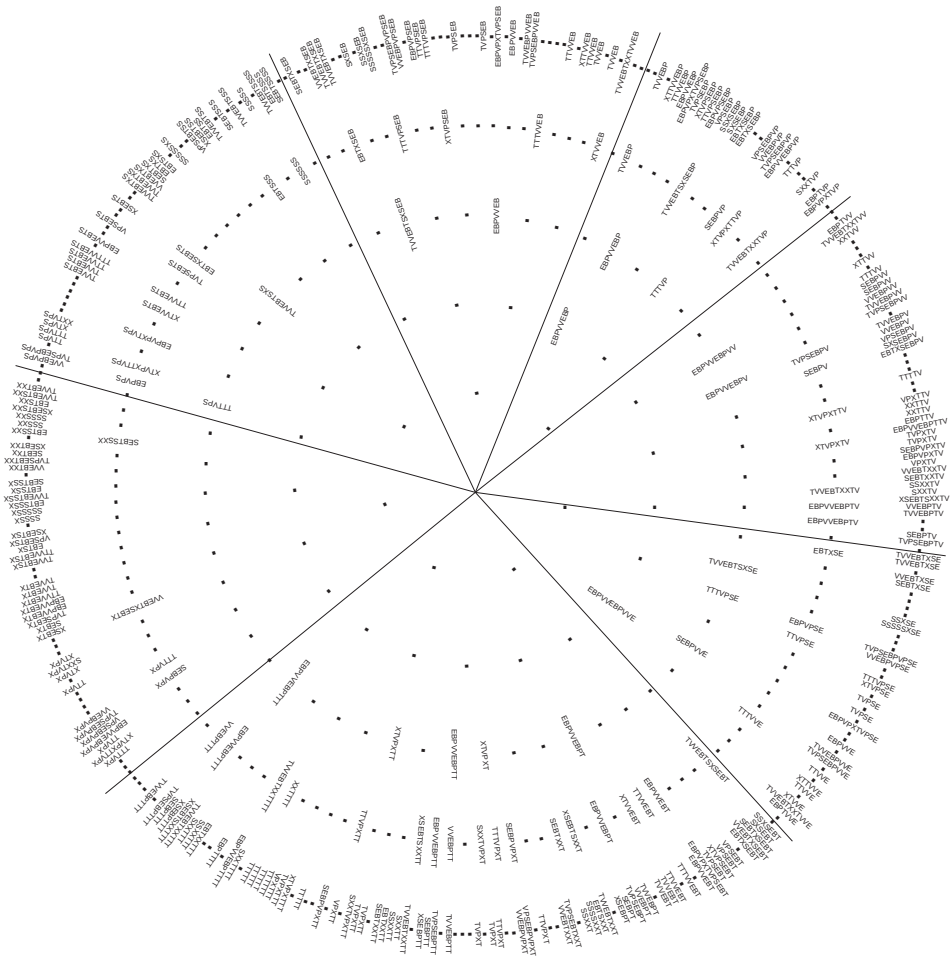


Figure 7.8: Arrangement of Reber words on a hyperbolic lattice structure. The words are arranged according to their most recent symbols (shown on the right of the sequences). The hyperbolic lattice yields a sector partitioning.

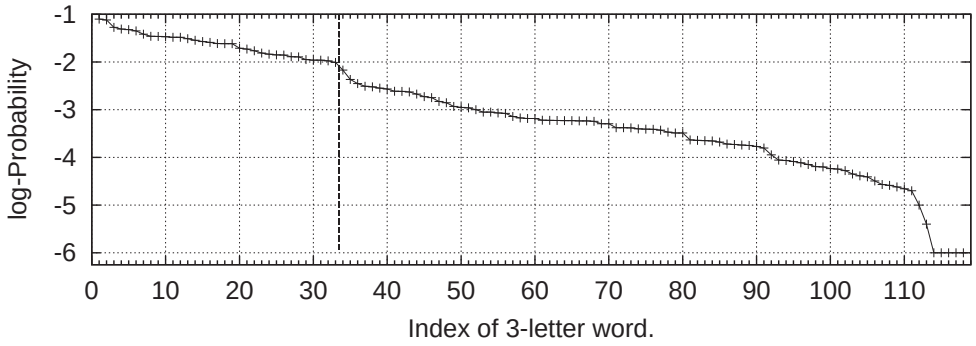


Figure 7.9: Likelihood of extracted trigrams. The most probable combinations are given by valid trigrams, and a gap of the likelihood can be observed for the first invalid combination.

Figure 7.10: Arrangement of Reber words on a Euclidean lattice structure. The words are arranged according to their most recent symbols (shown on the right of the sequences). Patches emerge according to the most recent symbol. Within the patches, an ordering according to the preceding symbols can be observed.

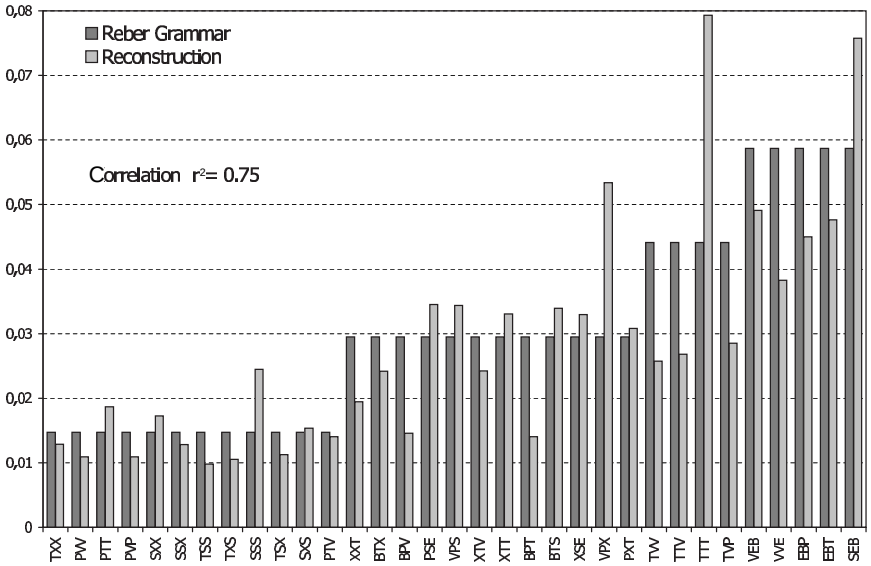


Figure 7.11: SOMSD frequency reconstruction of trigrams from the Reber grammar.

Chapter 8

The Merge SOM (MSOM) with data topology

God does not care about our mathematical difficulties.
He integrates empirically.

Albert Einstein

The merge SOM (MSOM) context model [145] presented in this chapter aims at the combination of two desirable objectives: ⟨1⟩ to free the context representation from its intimate connection to the SOMSD grid indexation scheme, and at the same time ⟨2⟩ to get a yet more efficient update of context and weight than RecSOM. The first request seeks to bridge the gap between the SOM neuron grid topology and the NG data topology. The second one, concerning efficiency, will be accompanied by the reduction of expressiveness of the context to focus on proper representations of immediate precedence relationships. So, neural gas with efficient context representation is the final target architecture, and the merge context model has been designed to meet the requirements [145]. The general formulation of the MSOM model will be preceded by a revisit of a distantly related model, the temporal Kohonen map (TKM) [23]; finally, MSOM will be combined with neural gas [147].

Temporal Kohonen Map (TKM)

An early approach to sequence processing with SOM is the temporal Kohonen map (TKM) proposed by Chappell and Taylor [23]. The TKM neurons implement recurrence in terms of leaky signal integration. For a sequence $(\mathbf{x}^1, \ldots, \mathbf{x}^t)$, the integrated distance of neuron i with weight vector $\boldsymbol{w}^i$ is computed by

$$\mathrm{d}_i(t) = \sum_{j=1}^{t} \alpha \cdot (1-\alpha)^{(t-j)} \cdot \|\mathbf{x}^j - \boldsymbol{w}^i\|^2 \,.$$

This distance expresses the neurons' quality of fit, not only to the current input pattern, but also to the exponentially weighted past. The parameter $\alpha \in (0;1)$ is kept constant to control the rate of decay signal during summation, that is the balance between the currently processed and the historic inputs. Hence, winner selection denotes the best matching compromise between the present and the past.

An equivalent recursive formalization of the integrated distance $\mathrm{d}_i(j)$ of neuron i after the j th time step is given by $\mathrm{d}_i(j) = \alpha \cdot \|\mathbf{x}^j - \boldsymbol{w}^i\|^2 + (1-\alpha) \cdot \mathrm{d}_i(j-1)$ with $\mathrm{d}_i(0) := 0$. TKM training is realized as Hebbian learning, i.e. the weights $\boldsymbol{w}^i$ are adapted each time step with respect to the unintegrated current input $\mathbf{x}^j$ according to the standard SOM update rule $\triangle \boldsymbol{w}^i = \gamma \cdot \mathrm{h}_\sigma(\mathbf{x}^j, \boldsymbol{w}^i) \cdot (\mathbf{x}^j - \boldsymbol{w}^i)$ [23].

The recurrent SOM of Koskela et al. is similar to TKM; it takes into account the integrated direction of change not only for distance calculation but also for weight update [87]. Both RSOM and TKM refer to context by signal integration. Only RSOM stores context information in the weight vectors for further comparison, but due to summation, these vectors do not allow a differentiation between the currently processed pattern and the history. Therefore, the interpretation and analysis of trained RSOMs is a difficult task.

In the following model, a strict distinction is made between the representation of the sequence elements and the context. This beneficial approach doubles the memory requirement of neurons to explicitly store precedence relationships of neuron activations.

8.1 Merge context

Roughly speaking, the merge context refers to a fusion of two properties characterizing the previous winner: the weight and the context of the last winner neuron are merged by a weighted linear combination. In addition to the mandatory weight vector $\boldsymbol{w}^i$ for representing the given pattern, each neuron i possesses a context vector $\boldsymbol{c}^i \in \mathcal{C}$ with the same dimension d as the weight vector. The adaptation target is given by a global *context descriptor* which represents each training step in terms of the currently presented pattern and a recursively expressed back-reference to the past. Thus, the vector tuple $(\boldsymbol{w}^i, \boldsymbol{c}^i) \in \mathcal{W} \times \mathcal{C} \approx \mathcal{W}^2$ of neuron i in the product space of weights and contexts is adapted into the direction of the current pattern and context descriptor according to Hebb learning. The inclusion of $\mathcal{C} \subseteq \mathcal{W}$ is the result of the involved linear combination of the weight vectors which will be explained below.

The winner is the best matching neuron j for which the recursively computed distance

$$\mathrm{d}_j(\mathbf{x}^t) = (1-\alpha) \cdot \|\mathbf{x}^t - \boldsymbol{w}^j\|^2 + \alpha \cdot \|\mathbf{c}^t - \boldsymbol{c}^j\|^2$$

to the current sequence entry $\mathbf{x}^t$ and the context descriptor $\mathbf{c}^t$ is minimum; weight and context contributions to the distance are balanced by the parameter α.
The context descriptor

$$\mathbf{c}^t = (1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}}$$

is the linear combination of the properties of winner $I_{t-1} = \arg\min_{j\in 1,\ldots,n} \mathrm{d}_j(\mathbf{x}^{t-1})$ in the last time step. A typical merging value for $0 \leq \beta < 1$ is 0.5.

8.2 Merge context for neural gas (MNG)

Integration of the merge context into self-organizing networks like the SOM or into LVQ is easily possible. Here, the focus is put on a combination of the context model with neural gas (NG) that will be called merging neural gas (MNG).

The extension of the neural gas dynamic by the merge context is straight forward: after the presentation of sequence element $\mathbf{x}^t$, for each neuron j its rank $k = \mathrm{rnk}_{\mathbf{x}}(j)$ is computed with the recursive distance $\mathrm{d}_j(\mathbf{x}^t)$ from above. The update amount for the weight vector is calculated as usual, involving the exponential function of the rank, and the same kind of update is applied to the context vector:

$$\begin{aligned} \Delta \boldsymbol{w}^j &= \gamma_1 \cdot \exp(-\mathrm{rnk}_{\mathbf{x}}(j)/\sigma) \cdot (\mathbf{x}^t - \boldsymbol{w}^j)\,, \\ \Delta \boldsymbol{c}^j &= \gamma_2 \cdot \exp(-\mathrm{rnk}_{\mathbf{x}}(j)/\sigma) \cdot ({\mathsf{c}}^t - \boldsymbol{c}^j)\,. \end{aligned}$$

As stated above, the context descriptor ${\mathsf{c}}^t$ is updated during training by keeping track of the respective last winner. In experiments, the learning rates have been set to identical values $\gamma_1 = \gamma_2 = \gamma$. The neighborhood influence σ decreases exponentially during training to obtain neuron specialization.

Choice of the weight/context balance parameter α

A crucial parameter of the recursive distance is α which is considered in this paragraph. The initial contribution of the context term to the distance computation and thus to the ranking order is chosen low by setting the weight/context balance parameter α to a small positive value. Since the weight representations become more reliable during training, gradually more attention can be paid to the specific contexts that refer to them. Thus, after an initial weight specialization phase, α can be unfixed and steered to a value that maximizes the neuron activation entropy. In computer implementations, such an entropy-driven adaptation has been realized: α is increased, if the entropy is decreasing, i.e. the representation space for the winner calculation is widened by allowing more context influence to counteract the specialization of the neurons on only the patterns; otherwise, if the entropy is increasing, α is slightly decreased in order to allow a fine tuning of the context influence and of the ordering. As a result, the highest possible number of neurons should — on average — be equally active by the end of training. Figure 8.1 shows neuron histograms that illustrate the involvement of neurons, starting with two specialized neurons shown in the inset, and after further training with entropy-controlled context influence. Thinking in terms of hierarchical neural activation cascades, this heuristic may not be optimal, for example when a small number of often visited root states are branching out to states of decreasing probability. However, the entropy-driven α-control strategy has proved to be very suitable for obtaining good results in the experiments.

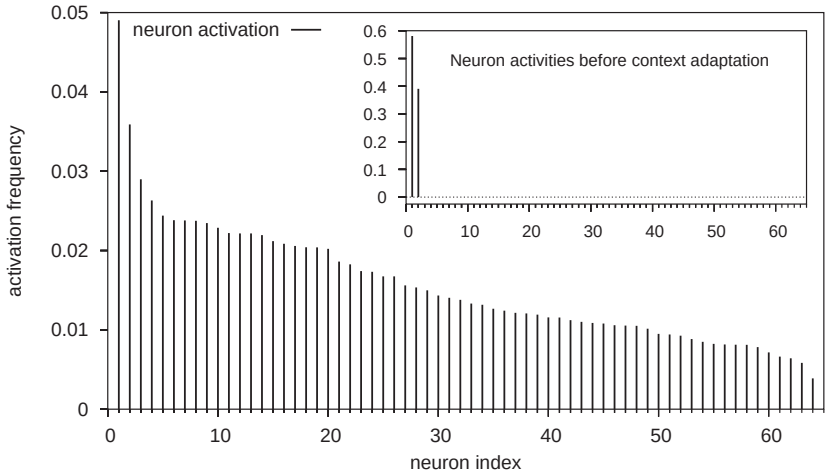


Figure 8.1: Sorted neuron activation frequencies for the α-entropy control strategy for a biased binary automaton. Inset: before context consideration, two neurons specialize, reflecting the sequence element frequencies $P(\mathbf{0}) = 3/7$ and $P(\mathbf{1}) = 4/7$. Large plot: after entropy maximization, all 64 neurons are involved.

8.3 Properties of the merge context

The properties of the merge context is studied in two steps: ⟨1⟩ the optimum choices for the prototype weight and the context are investigated; ⟨2⟩ it is proved that these optima result from the training dynamic as stable fixed points.

⟨1⟩ The best adaptation of neuron j is the one for which $\boldsymbol{w}^j$ and $\boldsymbol{c}^j$ yield $\mathrm{d}_j(\mathbf{x}^t) = 0$:

$$\mathrm{d}_j(\mathbf{x}^t) = (1-\alpha)\cdot\|\mathbf{x}^t - \boldsymbol{w}^j\|^2 + \alpha\cdot\|(1-\beta)\cdot\boldsymbol{w}^{I_{t-1}} + \beta\cdot\boldsymbol{c}^{I_{t-1}} - \boldsymbol{c}^j\|^2 \,.$$

Both squared summands can be considered separately. The left one trivially becomes the minimum of 0, if the weight vector represents the input pattern, i.e. for $\boldsymbol{w}^{opt(t)} = \boldsymbol{w}^j = \mathbf{x}^t$. Then, by induction, the right one expands to

$$\begin{aligned}
\boldsymbol{c}^{opt(t)} &= (1-\beta)\cdot\mathbf{x}^{t-1} + \beta\cdot\boldsymbol{c}^{I_{t-1}} \\
&= (1-\beta)\cdot\mathbf{x}^{t-1} + \beta\cdot\big((1-\beta)\cdot\mathbf{x}^{t-2} + \ldots + \beta\cdot((1-\beta)\cdot\mathbf{x}^1 + 0)\big) \\
&= \sum_{j=1}^{t-1}(1-\beta)\cdot\beta^{j-1}\cdot\mathbf{x}^{t-j} \qquad \text{(note: } j-1 \text{ is exponent, } t-j \text{ is index)}
\end{aligned}$$

with the assumption of zero context at the sequence start $\mathbf{x}^1$.

⟨2⟩ Now, focusing on the convergence of a neuron that is specialized on a particular sequence element within its unique context, asymptotically stable fixed points of the training update dynamic are obtained. The analysis of iterative weight adaptation in presence of the target vector yields:

$$\|\boldsymbol{w}^{I_t} + \gamma\cdot(\mathbf{x}^t - \boldsymbol{w}^{I_t}) - \mathbf{x}^t\| = (1-\gamma)\cdot\|\boldsymbol{w}^{I_t} - \mathbf{x}^t\| \quad \Rightarrow \quad \boldsymbol{w}^{I_t} \to \mathbf{x}^t \,.$$

This describes an exponential convergence because of $\gamma \in (0, 1)$. Analogously,

$$\|\boldsymbol{c}^{I_t} + \gamma \cdot \Big((1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}} - \boldsymbol{c}^{I_t}\Big) - \boldsymbol{c}^{opt(t)}\| \;\Rightarrow\; \boldsymbol{c}^{I_t} \to \boldsymbol{c}^{opt(t)}$$

describes the context convergence, if

$$(1-\beta) \cdot \boldsymbol{w}^{I_{t-1}} + \beta \cdot \boldsymbol{c}^{I_{t-1}} \to \boldsymbol{c}^{opt(t)}$$

can be shown. With $\boldsymbol{w}^{I_{t-1}} \to \mathbf{x}^{t-1}$ and by induction of $\boldsymbol{c}^{I_{t-1}} \to \boldsymbol{c}^{opt(t-1)}$ with $\boldsymbol{c}^{I_1} := 0$:

$$\begin{aligned}(1-\beta) \cdot \mathbf{x}^{t-1} + \beta \cdot \boldsymbol{c}^{opt(t-1)} &= (1-\beta) \cdot \mathbf{x}^{t-1} + \beta \cdot \sum_{j=2}^{t-1} (1-\beta) \cdot \beta^{j-2} \cdot \mathbf{x}^{t-j} \\ = \sum_{j=1}^{t-1} (1-\beta) \cdot \beta^{j-1} \mathbf{x}^{t-j} &= \boldsymbol{c}^{opt(t)} \quad \square\end{aligned}$$

This sum for $\boldsymbol{c}^{opt}$ denotes a fractal encoding of the context vector in the weight space, which is known to be a very compact and efficient representation [149].

Results given in the experiment section show that for binary sequences, a non-overlapping fractal context emerges which resembles a Cantor set, if a parameter of $\beta = 0.5$ is chosen. The spreading dynamic of the context, being initialized to the average data of zero for assumed mean subtracted data, is self-organizing with respect to the density of the context contributions: since the context is a function of the previous winner's weight and context, the adaptation is a moving target problem; therefore, it is generally a good policy to have either a faster weight adaptation than context update, or to put more influence on the pattern matching than on the context matching by choosing $\alpha < 0.5$.

In theory, the length of the neurons' temporal receptive fields is limited by the computing accuracy for discriminating between fixed point solutions of the context dynamic expressed by the fractal summation. In practice, however, for reasonably distributed data, a very large number of neurons can be used without interference, because multi-modality and high-dimensionality yield a reliable context spreading.

History is a race between education and catastrophe.

H. G. Wells

8.4 MSOM experiments

For comparison with SOMSD the same three data sets have been used for training MNG: ⟨1⟩ the continuous Mackey-Glass 1D time series, ⟨2⟩ discrete sequences from binary automata, and ⟨3⟩ words generated with the Reber grammar. In addition, ⟨4⟩ a medical 3-variate series with physiological observations has been investigated, and finally, ⟨5⟩ a promising experiment on speaker identification based on an a posteriori labeling of trained MSOM prototypes has been carried out.

Mackey-Glass time series

An introductory example for MNG refers to the Mackey-Glass series which has already been discussed in Section 7.4 on page 84 for the SOMSD approach. The errors of different quantizing techniques are shown in Figure 7.3 on page 84, and all but one curve have already been discussed in that section. Now, the focus is put on the not yet considered plot for MNG. As for the other methods, the corresponding training involves 100 prototypes. The parameters are $\beta = 0.75$, and the entropy-controlled balance parameter starts at $\alpha = 0$, reaching a final value of $\alpha = 0.18$ after training. The values of the trained MNG context cover the subinterval $[0.7; 1.15]$ of the input domain $[0.4; 1.3]$. As shown in Figure 8.2, all neurons are well spread in the context vs. weight plane; the vertical line at a weight value of 0.93 marks the average of 10^6 Mackey-Glass values which serves as the prototypes' context initialization. At first, context influence is ignored, and the prototype weight distribution is — by a magnification functional — closely related to the data distribution shown in the histogram Plot 8.3. By virtue of the subsequently activated context dynamic, an additional dimension is utilized in order to represent also the history of the currently processed element. This specialization is successful enough to just outperform all other methods in the quantization error Graph 7.3. It must be kept in mind, though, that the quantization ability of MNG is not restricted by a fixed target grid topology as in case of most of the other methods.

The context control parameter α in the distance calculation exploits the entropy of the neural activity during training. Figure 8.4 shows the entropy starting from highly specialized neurons without context: a decent number of active neurons yields a skewed activity histograms which lead to small entropy values. By gradually augmenting the context influence α, a richer state space for the distance computation is obtained. This triggers a more detailed winner selection, and the entropy of neuron activations grows.

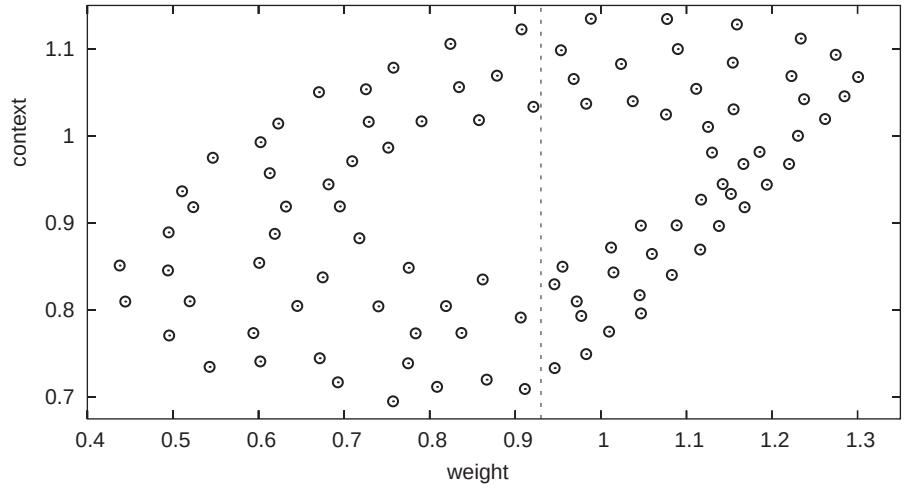


Figure 8.2: Prototypes in the MNG context vs. weight space for the Mackey-Glass series.

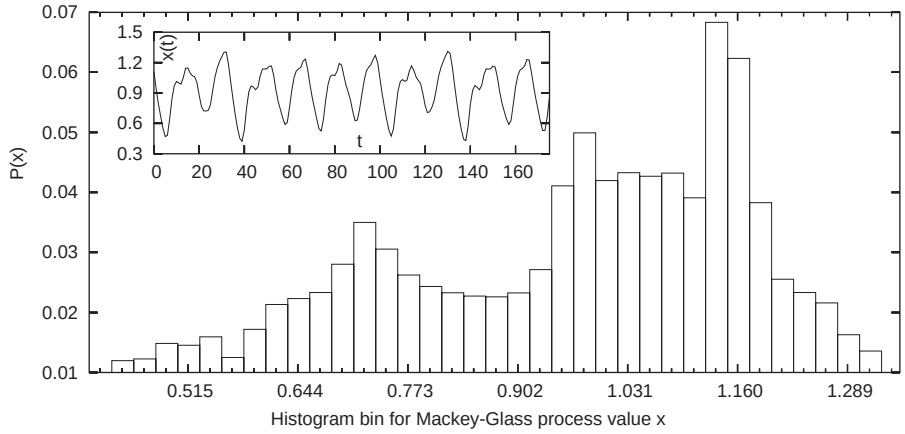


Figure 8.3: Histogram for the Mackey-Glass series. The inset shows a part of the series.

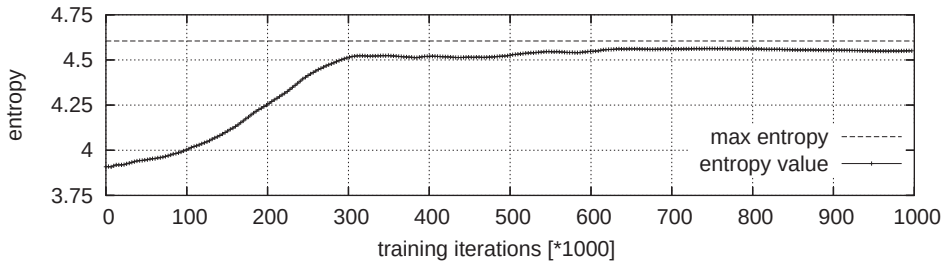


Figure 8.4: MNG entropy during Mackey-Glass training, steering α from 0 to 0.18.

A theoretical upper bound is $H = -\sum_{i=1}^{100} p_i \cdot \log(p_i) = -100 \cdot 1/100 \cdot \log(1/100) = 4.605$ with equal activation probabilities of $p_i = 1/100$ for all neurons. The control strategy for α is to adapt it dynamically at run time, aiming at entropy maximization: α is increased in case of an entropy decay trend and it is decreased otherwise. As shown in Figure 8.4, entropy saturation takes place after about 300 cycles. Spurious entropy fluctuations and instabilities during further adaptation of α have been avoided by means of a momentum term. Empirically, a sampling of about 1,000 entropy values proved to be sufficient for controlling a complete training task, a number that has also been used in the subsequent experiments.

Binary automata

In the second learning task, the focus is put on the representation of the most likely subwords in binary sequences generated by automata with given transition probabilities. Two questions concerning the MNG context are addressed: Can the theoretical fractal encoding be observed in experiments? What is the empirical representation capability?

Context development

Figure 8.5 displays the experimental context space resulting from MNG training of 128 neurons for a random binary sequence that contains 10^6 symbols **0** and **1** independently drawn with $P(\mathbf{0}) = P(\mathbf{1}) = 1/2$. Parameters are a learning rate of $\gamma = 0.03$, a fair combination of context and weight by $\beta = 0.5$, and an initial context influence of $\alpha = 0.001$ that has reached $\alpha = 0.34$ after training. During adaptation, two of the 128 neurons have fallen idle, the others finally represent meaningful sequences. Plot 8.5 is reduced to the 63 active neurons that represent the current symbol **0**. These neurons refer to points in the context space which are located on the lower horizontal line of zeroes. It can be observed that the context fills the input space in the interval $[0; 1]$ with an almost equidistant spacing. The stacked symbols point into the further past of the neurons' temporal receptive fields. In good agreement with the theory Section 8.3, the longest sequences uniquely discriminated by the neurons are arranged in a Cantor-like way in the context space.

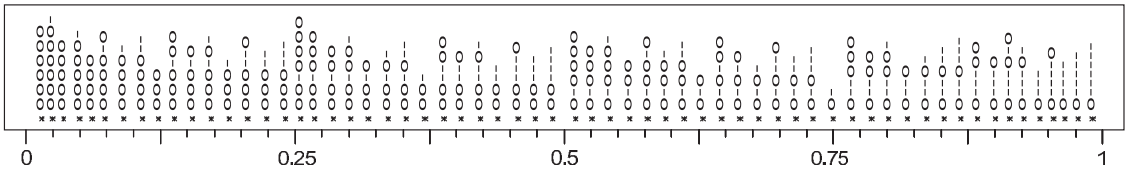


Figure 8.5: MNG context associated with the current symbol **0** of a binary sequence.

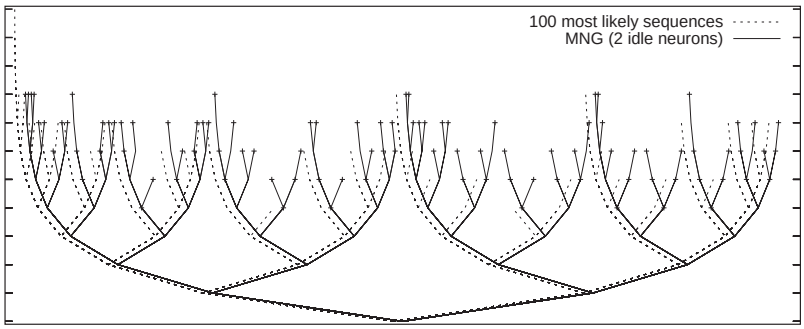


Figure 8.6: MNG receptive fields for the binary automaton. Left branching: **0**; right: **1**.

Representation capability

For comparison, the empirical representation capability has been investigated for the same transition probabilities that has been used for SOMSD in Section 7.4: $P(\mathbf{0}|\mathbf{1}) = 0.4$ and $P(\mathbf{1}|\mathbf{0}) = 0.3$. Training has been carried out with $\gamma = 0.025$, $\beta = 0.45$, and the initial value of $\alpha = 0$ was steered to a final value of 0.37 after 10^6 pattern presentations. Two neurons fall into idle state during training, because the specialization on updating context is a moving target problem which can cause prototypes to surrender at the varying data boundaries. At the beginning, i.e. for zero context influence, the weight specialization to almost crisp values of 0 and 1 can be observed; finally, the contexts are also spread over the interval $[0; 1]$, exhibiting a fractal arrangement. This pattern is similar to the one obtained in the previous experiment, but it is biased with respect to the automaton state probabilities.

Figure 8.6 shows, in tree form, the resulting 100 MNG neurons' receptive fields which correspond to the longest words for which neurons are still unique winners. This tree is compared to the 100 most likely sequences produced by the automaton. As shown in the figure, many neurons have developed disjoint receptive fields, with the exception of transient neurons, indicated by bullets on the interior tree nodes, for which the descendants represent still longer sequences. After all, a total number of 63 longest words can be discriminated by MNG, and they reflect pretty well the most frequent 100 sequences generated by the automaton.

In contrast to the results of HSOMSD, presented in Figure 7.5 on page 86, the context representation for the binary data set succeeds better with MNG. The HSOMSD model, representing a total number of 28 distinct longest sequences, requires many idle nodes for a clear separation of incompatible adjacent regions on the 2D-hyperbolic neural grid.

If a SOMSD neuron grid is not needed, MNG helps to reduce the additional parameters for the description of the target grid topology. Without this specification, only the number of neurons m, the learning rate γ, the metric balance parameter for context and weight α, their merging parameter β, and some canonic initializations are required. These few choices make the selection of a good quantization model quite easy.

Reber grammar

Once more, the Reber grammar automaton, already used in Section 7.4 and depicted in Figure 7.7 on page 89, is studied for MNG. Again, a number of $3 * 10^6$ and 10^6 six-dimensional input vectors have been used for training and testing, respectively. For comparison with the successful hyperbolic SOMSD, 617 MNG neurons are taken. The merge parameter is $\beta = 0.5$, the context influence is initialized by $\alpha = 0$, the starting neighborhood size is $\sigma = 617$, and the context vector is initialized to $\mathbf{0} \in \mathbb{R}^6$ which is the center of gravity of the embedded symbols. The learning rate is $\gamma = 0.03$; after training, the adjusted parameters are $\sigma = 0.5$ and $\alpha = 0.43$.

Finally, the context information stored by the ensemble of neurons has been analyzed. The average length of Reber strings from the test sequence leading to unambiguous winner selection is 8.902, whereby 428 neurons develop a distinct specialization on Reber words. The reference results of the hyperbolic SOMSD are an average string length of 7.23, and a number of 338 active neurons.

In addition to this external statistics based on the test sequence, a network-internal backtracking has been performed like this: $\langle 1 \rangle$ visit for each neuron all other neurons, $\langle 2 \rangle$ calculate their merging vectors for context and weight, $\langle 3 \rangle$ sort the vector distances to the current neuron's context increasingly, $\langle 4 \rangle$ determine the symbols associated with the neurons in the sorted list. With this context backtracking, on average 67.9 neurons, corresponding to the best matching contexts, represent the same symbol, before a different symbol is encountered. This large number is a very strong support for a high consistency of the context and for a proper precedence learning.

Therefore, further backtracking has been performed to collect the string associated with each neuron, strings composed of symbols represented by the recursively visited best matching predecessors. The string assembly stops at the first revisit of a neuron: words with an average length of 13.78 are produced, most of them with valid Reber grammar. The longest word **TVPXTTVVEBTSXXTVPSEBPVPXTVVEBPVVEB** corresponds almost perfectly to the data-driven specialization **TVPXTTVVEBTSXXTVPSEBPVPXTVVE** that has been determined by keeping track of the winners for the training set. A similarly high correspondence of shorter strings could be observed for most neurons.

For comparison with SOMSD on page 89ff., the best matching backtracked words of length 3 have been counted. Only valid Reber strings were found, of which the frequencies are given in Figure 8.7. Interestingly, MSOM does not overemphasize the **VPX** and **SSS** cycles as strong as SOMSD, and for **TTT** even an under-representation can be found. The correlation of the probabilities of Reber trigrams and the reconstructed frequencies from the map is $r^2 = 0.77$, which is even higher than $r^2 = 0.75$ for SOMSD. This is another indication of an adequate neural representation of the Reber characteristic.

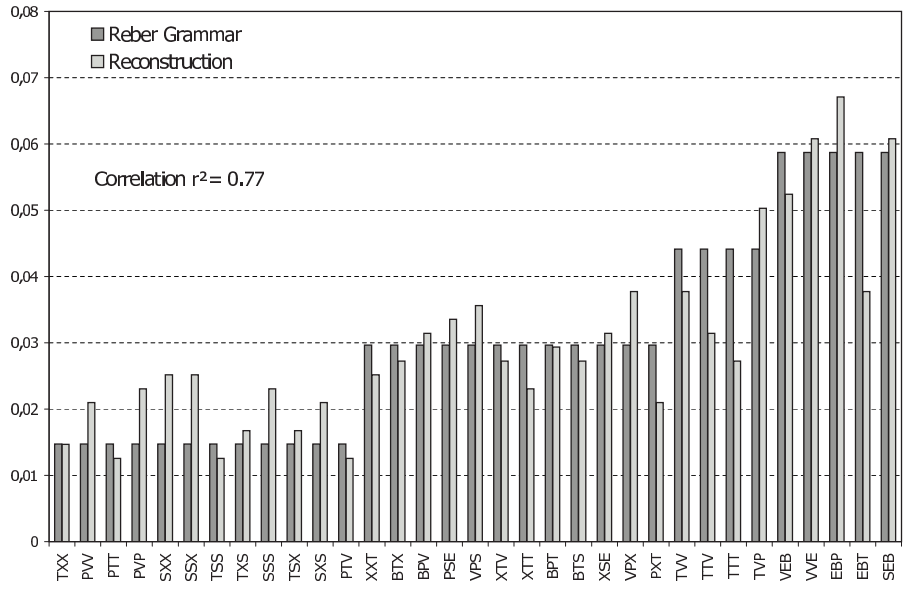


Figure 8.7: MSOM frequency reconstruction of trigrams of the Reber grammar.

Physiological 3-variate series (`B-1991`)

A real-life data series has been taken from `physionet.org` containing recordings of a patient suffering from apnea during sleep. The first column contains the heart rate, the second the chest volume, and the third is the blood oxygen concentration. Linear trends, caused by parameter drifts in the gage, have been removed from the data by calculating the first derivative of a smooth order 4 spline interpolation. Mean subtraction has been carried out, and a logarithmic transform $\tilde{x} = \mathrm{sgn}(x) \cdot \log(|x| + 1)$ has been computed for each column to account for the sharp peaks in the otherwise moderate physiological dynamic. Data preprocessing has been concluded by a z-score transform. The final `B-1991` set contains 64,701 samples of temporally dependent 3D-vectors.

For the obtained series, training has been conducted with a 20-fold presentation of its elements. It is assumed that the singular wrap-around events at the end of the sequence to its beginning do not disturb the training significantly. Figure 8.8 shows the quantization errors for three methods, each of them possessing a number of 617 neurons: original NG, hyperbolic SOMSD with seven neighbors per neuron, and MNG. The error is the average of the three attributes' variances that have been obtained separately for each column the same way as for the Mackey-Glass series. As expected, NG without context is the worst temporal quantizer, MNG works pretty well up to four steps in the past, and it is then outperformed by SOMSD. Since the temporal data vectors form, in phase space, a curve of concatenated spatial loops around the origin $\mathbf{0}$ with a roughly unimodal density distribution, a long-term integration for the MNG context falls into the indistinguishable center of gravity of the dynamic. Thus, distinct context clusters cannot easily emerge for histories longer than four steps, and MNG results asymptotically in the standard NG

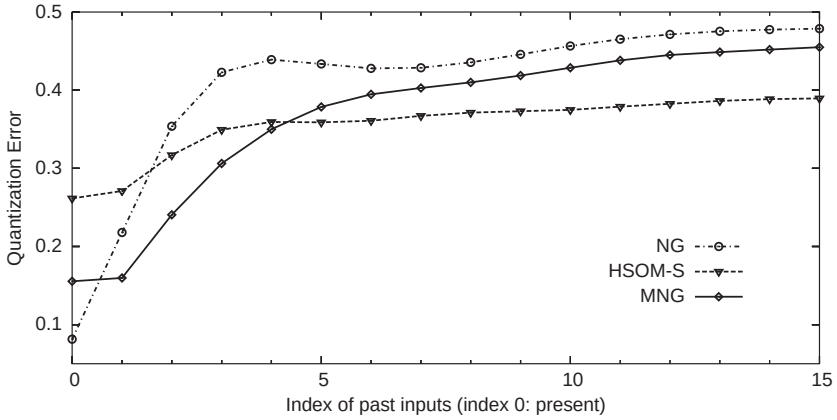


Figure 8.8: Temporal quantization errors for the preprocessed `B-1991` data.

quantization performance. SOMSD does not suffer from this problem, and it remains on a smaller error level; this is because the SOMSD context representation does not require a folding of historic information into the weight domain, but it uses the more independent space of neuron indices instead.

Speaker identification by a posteriori MNG labeling

This experiment revisits the UCI speaker data discussed in Section 5.4 on page 51 [11]. In contrast to the SRNG-specific data resampling explained in that section, MNG does not require this resampling operation; only mean subtraction has been applied to the original data. However, care must be taken during the data presentation: the temporal structure of each articulation is represented by a variable number of successive 12-dimensional cepstrum vectors, but between different utterances there is no such temporal connection. Therefore, a special neuron is added to represent the default state $\boldsymbol{w} = \boldsymbol{c} = \mathbf{0}$ when no context is available at an utterance start.

After the unsupervised MNG training that does not account for the speaker identity, each of the 150 neurons used is assigned a 9-bin histogram containing activation frequencies for the speakers from the training set. For each articulation sequence from the test set, the accumulated majority vote over the bins of activated neurons is calculated to identify the speaker. The resulting histograms are very specific for the speakers. Figure 8.9 supports a good clusterability of the trained weight/context product space: the 2×12-dimensional prototypes have been projected to two dimensions by means of the linear discriminant analysis (LDA) technique [46]; the parameters of the 24×2-projection matrix have been adapted by projection pursuit to yield a maximum Gaussian separation of the prototypes in the two dimensional target space [42]. Hence in the unprojected original space, a good separability can be expected, which is confirmed by the experiment.

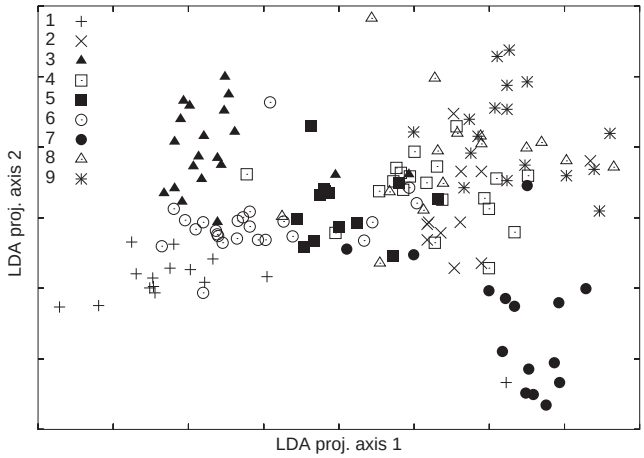


Figure 8.9: LDA-projection of the weight × context-space of speaker data.

When, finally, the obtained a posteriori labels are checked against the data, there is no error on the training set and an error of only 2.7% on the test set. This accuracy is much better than the reference errors of 5.9% and 3.8% accompanying the original data set. Surprisingly, the result is even slightly better than the SRNG error of 2.76%, although the number of free prototype parameters are $240 \times 18 = 4320$ for SRNG and only $24 \times 150 = 3600$ for MNG, given by the product of prototype dimension and the number of prototypes. For MNG training with 1,000 neurons, a number of 21 neurons fall into idle state during context formation, and the error decreases to only 1.6%. This interesting result illustrates that good-natured interpolation emerges from the dynamic, if more neurons than training examples are given.

Part IV

Discussion and Outlook

Dave, my mind is going, I can feel it . . . I can feel it.

HAL 9000

Summary

Sequence processing has been studied with self-organizing neural models for supervised and unsupervised processing of multi-dimensional discrete and continuous data. Two widely established methods, LVQ and the SOM, both being simple and powerful prototype-based Hebbian learners, have served as starting points for the extensions to sequence data presented in this work.

GRLVQ/SRNG Basically, time is integrated into LVQ by means of high-dimensional time window vectors for which the curse of dimensionality is reduced by adaptive metrics, and potential multi-modality is tackled by neural neighborhood cooperation which leads to the supervised relevance neural gas (SRNG) algorithm. The backbone of the presented LVQ extensions is the formulation in terms of a cost function minimization from which robust update rules are derived for the local prototype positions and for the global metric parameters. Due to the high modularity of SRNG, customized metrics can be easily plugged-in for specific learning tasks; for example, DNA processing highly profits from the locality improved bi-directional sequence metric LIK. In the case of GRLVQ with weighted Euclidean distance, the converged prototypes and dimension weighting factors can be used to turn a network into a BB-tree of classification rules. Experiments on sequence processing have been conducted for the prediction of word forms of linguistic data, for the attractor reconstruction of the Lorenz attractor by means of the proposed relevance embedding, for a speaker identification problem, and for DNA splice site recognition. In comparison with other methods, the presented LVQ variants produce excellent results by compact and intuitive data models with prototypes and relevance profiles that can be easily interpreted.

SOMSD/MSOM In the extended SOM, sequential inputs are represented by a recursive back-reference to the previous winner neuron: in case of the SOM for structured data (SOMSD), each neuron keeps track of a compact location index for addressing the most recent winner in the grid of neurons; in case of the merge SOM (MSOM), the grid index is replaced by a grid-independent context which combines the previously represented pattern and the previously active context.

The modifications of SOMSD presented in this work focus on sequence processing and introduce a graph-based alternative target lattice of neurons. This way, the potentially exponential context growth of sequences can be mapped to neuron neighborhoods with hyperbolic connectivity. Experiments with the Reber grammar have shown the struc-

tural difference of the map ordering for Euclidean and hyperbolic target grids. This and other experiments point out the good quality of context representation, expressed by low temporal quantization errors and by a high correspondence between training data and context. Context representations of trained maps have been obtained by backtracking neuron activation transitions and by data-driven calculations of the temporal receptive fields.

Good temporal quantization properties also apply to the new MSOM technique. Not only the experiments but also the analysis of the dynamic yield compact context encoding by fractal codes which turn out to be stable fixed points of the training. Particularly, multi-modal data can be efficiently represented this way.

Limitations

Despite the very good classification results of SRNG, a principal drawback remains for temporal data: although adaptive metrics reduce the curse of dimensionality, the time window training is computationally still demanding. For multi-dimensional sequence entries, traditional measures like the autocorrelation function fail for the determination of the time window dimension; alternatively, the multi-dimensional histogram calculation for a reliable identification of the time shift corresponding to the first minimum of the mutual information requires too many data points. Therefore, the combination of the MSOM context with SRNG would be desirable to make sure that the available number of neurons is optimally used for representing the most prominent data histories. This combination has indeed been studied, but a yet unsolved problem has been encountered: SRNG adapts correct and wrong prototypes towards the presented pattern and away from it. Now, how to deal with the context? If the prototype is correct, the MNG adaptation can be applied, but for the wrong prototype neither adaptation towards the current context descriptor nor away from it can be justified. The paradox is: either the context vector is made more similar, although the represented class is wrong, or the context is made more dissimilar but is then representing a wrong history. Both variants and other scenarios have been extensively tested, but without producing results superior to the simple presentation of the compound pattern $(\mathbf{x}^{t-1}, \mathbf{x}^t, y^t)$ to standard SRNG. Therefore, a mathematical derivation in the framework of supervised recursive cost function minimization would be worthwhile.

For the given experiments with SOMSD and MSOM, data sets are mostly good-natured in two senses: the data are stationary by definition or quasi-stationary by multi-epoch training, and the context of the data sets is well connected. This coherence is important, because in a counter-example, where $\mathbf{x}^t$ and $\mathbf{x}^{t-2}$ are correlated, but neither is correlated to $\mathbf{x}^{t-1}$, an overhead of all occurring transitions of $\mathbf{x}^{t-2}$ to $\mathbf{x}^{t-1}$ must be learned before the subsequence $(\mathbf{x}^{t-2}, \mathbf{x}^{t-1}, \mathbf{x}^t)$ can be reasonably represented. Hochreiter and Schmidhuber [71] and Hochreiter and Mozer [70] propose architectures for the modeling of long short-term memory LSTM to circumvent this problem. However, these models are rather designed for discrete sequences than for noisy data, and larger networks induce severe computational problems for the training [48].

Interestingly, the back-reference models for SOMSD and MSOM are not implementing just a simple fading memory for which events in the further past contribute less to the definition of the current state. As the experiments have demonstrated, an ordered activation dynamic leads to disjoint receptive fields. As a matter of fact, a past event has major influence on the pathway of activations. The length of such a historic chain, the temporal scope, depends on the data modes, the data dimension, the number of neurons in the network, on the quality of training, and, in case of MSOM, also on the computing accuracy.

Future work

Hebbian learning turns out to be a suitable method for the presented models, because a large number of training epochs can be realized. However, limitations occur for representing dynamics that take place on different time scales, or at its extreme, for one-shot learning problems. The question is how long it takes to identify and represent a given data example by a class or a specialized cluster of its own. In other words, the plasticity-stability dilemma of memory adaptivity to new patterns is addressed.

For still maintaining the Hebbian learning principle, this would require to know beforehand the importance of the currently presented stimulus, because the prototype adaptation rate should be greater for very prominent data and smaller for unsurprising examples. In addition to the definition of data similarity, the dissimilarity — which needs not be the inverse distance measure — could be used for assigning the learning rate associated with the novelty of a pattern.

In this respect, three ideas should be studied for future learning architectures: dynamic network sizes, context-specific metrics, and interactive network visualization.

1. Dynamic network growing and shrinking

A promising contribution to one-shot learning is to let new inputs create new representations by adding more neurons on demand. This is weakly related to learning by heart, using a dynamically growing table of unknown size. Therefore, a valuable extension of the presented classifiers would be an implementation of a data-driven dynamic network growing and age-based shrinking, such as proposed by Fritzke [44, 45]. He describes the dynamic growing for grid and data topologies for the supervised and the unsupervised case. This way, also nonstationary online data can be dealt with. For convergence and stability of training, the annealing of learning rates and the neighborhood radius is only suitable for the assumption of stationary data; for the ability to adopt to instationary changes, the dynamic network growth must ensure a basic level of activity, at least for the recently added neurons. An instationary scenario may be a comforting reason for not insisting on a gradient-based minimization technique which would be applied to a cost function that is subject to continuous changes.

2. Context-specific metrics

According to the well-known experience that different situations require different ratings even for the same input stimulus, a context-specific application of a particular metric and a weighting of several competing metrics are appealing domains for future research. More specifically, a multi-metric approach with experience-based metric selection, realized by Hebbian training, could capture different facets of the data. For example, one metric could measure the norm of a difference vector, another return the vector variance or the amount of the most prominent attribute. Since all vector norms in real-valued spaces are conceptually equivalent, possibly 'orthogonal' aspects of the data might be better captured by functions $\mathbb{R}^d \mapsto \mathbb{R}$ with non-metric properties. Then, of course, the notion of neural winner selection must be redefined. For biologically plausible learning and for perception binding, a double LVQ architecture with self-supervised reinforcement of input pairs from different data channels has been proposed by deSa [34]; this could be used as a basis for context-specific metric adaptation. For sequence processing, metrics sensitive to specific time scales could be realized by integrating the presented data over a number of observations with or without exponential weighting. Although the evaluator for multiple adaptive metrics is a natural extension to the generalized adaptive metrics as discussed for SRNG, a fundamental question of interest is whether the high number of free parameters can be reasonably adapted, or if parameter regularization can be realized. For unsupervised learning, metric adaptation must rely on implicit targets such as the maximization of the clusterability, of the information transfer, or of the topology preservation in grid architectures [15].

From a rather visionary point of view, context-specific evaluation refers to an implementation of artificial emotions, for which the cost function to be optimized depends on the current state of the classifier. As discussed e.g. in Aretz, this issue is of great interest for the robotics community with respect to the design of robust autonomous agents and with respect to creating well-accepted machine-human interfaces [3].

3. Interactive visualization

In combination with the SOM, a major concern of future work should be adequate visualization. For Euclidean SOMSD lattices, a proper display on a screen or on a sheet of paper is no problem in principle, but the development of long range back-reference connections to the temporal context shatters preceding winner nodes on large areas on the neural map. This specialization according to the ordering of the most recent symbol elements is wanted for the diversification of the representations. For sequence analysis, though, which is the focus of the presented SOMSD, a complementary view must answer the question of temporal backtracking: which are the most likely context states belonging to a neuron? Due to Hebbian learning, these temporal states are not given by the grid neighbors, and their determination would require an online temporal backtracking of the context of a neuron picked by the user.

Another interactive component must be provided for hyperbolic grid topologies, because a static plot can only focus on a fish-eye magnification of local data projections; thus, an overall data visualization can no longer be obtained, and visual inspection must be extended by an interactive component to browse a trained map. For example, Ontrup and Ritter have implemented a browser for navigation in a movie data base [114]. If interactive visualization is agreed on, also the beneficial neural gas data topology could be locally projected into plane in real time.

So, for a wide acceptance of HSOMSD and NG type networks, visualization and navigation should be a crucial point of further investigations. The remaining benefit of HSOMSD is its update efficiency which takes a computing time complexity of only $\mathcal{O}(m)$ for both the winner search and the neighborhood update, in contrast to the sorting complexity of $\mathcal{O}(m \cdot \log(m))$ for the NG prototype ranking. Trivial speedups can be obtained for SOMSD by defining a threshold for the neighborhood contribution below which more distant graph nodes are excluded from further visitation. The ranking complexity NG can be reduced by restricting the search to only k of m nearest prototypes, or by sorting distances on parallel hardware, for example with the bitonic merge sort.

Mission statement

This work has investigated aspects of temporal self-organization in Hebbian learning models. The author hopes that his contribution will be a source of ideas for future SOM designs. He thinks that these model will profit to a great extent from adaptive metrics, variable model topologies, context integration, and visualization. Additionally, as indicated by the back-reference models, self-introspection is useful for taking into account the current internal state for complementing the externally given stimulus. As in real life, such a further reflection is generally a good strategy before a decision is taken, and before a subsequent action is about to take influence on the future.

Bibliography

[1] N. Allinson, H. Yin, L. Allinson, and J. Slack, editors. *Advances in Self-Organizing Maps*. Springer, London, 1st edition, 2001.

[2] R. Andonie and A. Cataron. An informational energy LVQ approach for feature ranking. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 471–476. D-facto Publications, 2004.

[3] M. Aretz. Emotionen in der Künstlichen Intelligenz. Diplomarbeit, Fachbereich Mathematik, Naturwissenschaft und Informatik, Fachhochschule Gießen-Friedberg, 2004.

[4] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Computer Graphics Forum*, 15(3):387–396, 1996.

[5] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

[6] G. Barreto and A. Araújo. Time in self-organizing maps: An overview of models. *International Journal of Computer Research*, 10(2):139–179, 2001.

[7] G. Barreto, A. Araújo, and S. Kremer. A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case. *Neural Computation*, 15(6):1255–1320, 2003.

[8] H.-U. Bauer and T. Villmann. Growing a Hypercubical Output Space in a Self-Organizing Feature Map. *IEEE Transactions on Neural Networks*, 8(2):218–226, 1997.

[9] J. Baxter. The canonical distortion measure for vector quantization and function approximation. In *Proceedings of the 14th International Conference on Machine Learning*, pages 39–47. Morgan Kaufmann Publishers, 1997.

[10] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[11] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[12] S. Blohm. Data mining on sequences with recursive self-organizing maps. Bachelor thesis, University of Osnabrück, 2003.

[13] T. Bojer, B. Hammer, and C. Koers. Monitoring technical systems with prototype based clustering. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 433–439. D-side Publications, 2003.

[14] T. Bojer, B. Hammer, D. Schunk, and K. Tluk von Toschanowitz. Relevance determination in learning vector quantization. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 271–276. D-facto Publications, 2001.

[15] T. Bojer, B. Hammer, M. Strickert, and T. Villmann. Determining relevant input dimensions for the self-organizing map. In L. Rutkowski and J. Kacprzyk, editors, *Neural Networks and Soft Computing (Proc. ICNNSC 2002)*, Advances in Soft Computing, pages 388–393. Physica-Verlag, 2003.

[16] P. Bosch, K. Dalinghaus, B. Hammer, J.-P. Reuter, B. Schrader, T. Steffens, and C. Umbach. Cognitive architectures: The integration of rules and patterns. Research focus, Institute of Cognitive Science, University of Osnabrück, 2003.

[17] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, 1984.

[18] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. Technical report, Technical University of Denmark, Lyngby, Denmark, 1991.

[19] J. Buhmann. Data Clustering and Learning. In M. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 308–312. MIT Press, Cambridge, MA, 2nd edition, 2002.

[20] F. Camastra and A. Vinciarelli. Combining neural gas and learning vector quantization for cursive character recognition. *Neurocomputing*, 51:147–159, 2003.

[21] M. Casdagli. Recurrence plots revisited. *Physica D*, 108:12–44, 1997.

[22] R. Chang, W. Kuo, D. Chen, Y. Huang, J. Lee, and Y. Chou. Computer-aided diagnosis for surgical office-based breast ultrasound. *Archives of Surgery*, 135:696–699, 2000.

[23] G. Chappell and J. Taylor. The temporal Kohonen map. *Neural Networks*, 6:441–445, 1993.

[24] V. Cherkassky. Fuzzy inference systems: A critical review. In O. Kayak and L. Zadeh, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, pages 177–197. Springer, 1998.

[25] V. Cherkassky, D. Gehring, and F. Mulier. Comparison of adaptive methods for function estimation from samples. *IEEE Transactions on Neural Networks*, 7:969–984, 1996.

[26] V. Cherkassky and F. Mulier. *Learning from Data*. John Wiley & Sons, New York, 1998.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[28] M. Cottrell and J.-C. Fort. Etude d'un processus d'auto-organisation. *Annales de l'Institut Henri Poincaré*, 23(1):1–20, 1987.

[29] M. Cottrell, J.-C. Fort, and G. Pagès. Theoretical aspects of the SOM algorithm. *Neurocomputing*, 21(1–3):119–138, 1998.

[30] T. Cover and P. Hart. Nearest neighbor pattern classifcation. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[31] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley & Sons, New York, 1991.

[32] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin Analysis of the LVQ Algorithm. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS)*, volume 15, pages 462–469, Cambridge, MA, 2003. MIT Press.

[33] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. TiMBL: Tilburg memory-based learner (version 5.0) reference guide. Report ILK 03-10, CNTS - Language Technology Group, University of Antwerp, URL: http://ilk.uvt.nl, 2003.

[34] V. de Sa. Combining uni-modal classifiers to improve learning. In H. Ritter, H. Cruse, and J. Dean, editors, *Prerational Intelligence: Adaptive Behavior and Intelligent Systems without Symbols and Logic*, volume 2, pages 707–722. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[35] G. Dorffner. Neural networks for time series processing. *Neural Network World*, 6(4):447–468, 1996.

[36] W. Duch, R. Adamczak, and K. Grabczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.

[37] P. Duda, P. Hart, and D. Stork. *Pattern Classification.* John Wiley & Sons, New York, 2001.

[38] W. Ebeling, L. Molgedey, J. Kurths, and U. Schwarz. Entropy, complexity, predictability and data analysis of time series and letter sequences, 1999.

[39] J. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[40] D. Ford. Analysis of LVQ in the Context of Spontaneous EEG Signal Classification. Master's thesis, Department of Computer Science, Colorado State University, Fort Collins, CO 80523, 1996.

[41] A. Fraser and H. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33:1134–1140, 1986.

[42] J. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[43] B. Fritzke. Growing Grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[44] B. Fritzke. *Vektorbasierte Neuronale Netze.* Shaker Verlag, Aachen, 1998.

[45] B. Fritzke. Growing self-organizing networks - history, status quo, and perspectives. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 131–144. Elsevier Science, Amsterdam, 1st edition, 1999.

[46] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd edition, 1990.

[47] I. Gath and A. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–780, 1989.

[48] F. Gers, N. Schraudolph, and J. Schmidhuber. Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[49] A. Geva. Feature extraction and state recognition in biomedical signals with hierarchical unsupervised fuzzy clustering methods. *Medical and Biological Engineering and Computing*, 36(5):668–614, 1998.

[50] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[51] S. Gottschalk, M. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30:171–180, 1996.

[52] T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: generalizations and new optimization techniques. *Neurocomputing*, 21:173–190, 1998.

[53] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings IEEE Conference on Decision and Control (CDC)*, pages 761–766, San Diego, USA, 1979.

[54] M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, 2003.

[55] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35, 2004.

[56] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. Recursive self-organizing network models. *Neural Networks*, to appear, 2004.

[57] B. Hammer, A. Rechtien, M. Strickert, and T. Villmann. Rule extraction from self-organizing maps. In J. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 370–375, Madrid, Spain, 2002. Springer.

[58] B. Hammer, A. Rechtien, M. Strickert, and T. Villmann. Vector Quantization with Rule Extraction for Mixed Domain Data. Internal report, Institut für Informatik, Universität Osnabrück, 2003.

[59] B. Hammer, M. Strickert, and T. Villmann. Learning vector quantization for multimodal data. In J. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 370–375, Madrid, Spain, 2002. Springer.

[60] B. Hammer, M. Strickert, and T. Villmann. On the generalization ability of GRLVQ-networks. Report P-249, Institut für Informatik, Fachbereich Mathematik/Informatik, Universität Osnabrück, 2003.

[61] B. Hammer, M. Strickert, and T. Villmann. Supervised neural gas with general similarity measure. *Neural Processing Letters*, to appear, 2003.

[62] B. Hammer, M. Strickert, and T. Villmann. Relevance LVQ versus SVM. *Proceedings of ICAISC 2004, Lecture Notes in Computer Science*, 3070:592–597, 2004.

[63] B. Hammer, M. Strickert, and T. Villmann. Prototype based recognition of splice sites. In U. Seiffert, L. Jain, and P. Schweitzer, editors, *Bioinformatics using computational intelligence paradigms*. Springer, in press.

[64] B. Hammer and T. Villmann. Estimating relevant input dimensions for self-organizing algorithms. In N. Allinson, H. Yin, L. Allinson, and J. Slack, editors, *Advances in Self-Organizing Maps*, pages 173–180, London, 2001. Springer.

[65] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15:1059–1068, 2002.

[66] D. Hebb. *The Organization of behaviour: a neurophysiological theory*. John Wiley & Sons, 1949.

[67] M. Herrmann, H. Bauer, and R. Der. The perceptual magnet effect: A model based on self-organizing feature maps. In *Proceedings of the 3rd Neural Computation and Psychology Workshop*, pages 107–116, Stirling, Scotland, 1994.

[68] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing, Redwood City, CA, 1991.

[69] T. Heskes. Energy functions for self-organizing maps. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 303–316. Elsevier, Amsterdam, 1999.

[70] S. Hochreiter and M. Mozer. A discrete probabilistic memory model for discovering dependencies in time. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 661–668. Springer, 2001.

[71] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[72] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[73] A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

[74] D. James and R. Miikkulainen. SARDNET: a self-organizing feature map for sequences. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS)*, volume 7, pages 577–84, Cambridge, MA, 1995. MIT Press.

[75] M. Jordan. Serial Order: A Parallel Distributed Processing Approach. In J. Elman and D. Rumelhart, editors, *Advances in Connectionist Theory*, San Diego, La Jolla, CA, 1989. Institute for Cognitive Science, University of California.

[76] H. Kantz and T. Schreiber. *Nonlinear time series analysis.* Cambridge University Press, 1997.

[77] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'98)*, volume 1, pages 413–418. IEEE Service Center, Piscataway, NJ, 1998.

[78] S. Kaski. Convergence of a stochastic semisupervised clustering algorithm. Technical Report Publications in Computer and Information Science: A62, Helsinki University of Technology, Espoo, Finland, 2000.

[79] S. Kaski. Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947, 2001.

[80] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[81] T. Kohonen. Learning Vector Quantization for Pattern Recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.

[82] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, 3rd edition, 2001.

[83] T. Kohonen. Learning Vector Quantization. In M. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 631–635. MIT Press, Cambridge, MA, 2nd edition, 2002.

[84] T. Kohonen, S. Kaski, H. Lappalainen, and J. Saljrvi. The Adaptive-Subspace Self-Organizing Map (ASSOM). In *Proceedings of the Workshop on Self-Organizing Maps (WSOM)*, pages 191–196. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[85] J. Kolen and S. Kremer, editors. *A Field Guide to Dynamical Recurrent Networks.* IEEE Press, 2001.

[86] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Recurrent SOM with local linear models in time series prediction. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 167–172. D-facto Publications, 1998.

[87] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski. Temporal sequence processing using recurrent som. In *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Engineering Systems*, volume 1, pages 290–297, Adelaide, Australia, 1998.

[88] J. Kruskal. An overview of sequence comparison. In D. Sanko and J. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 1–44. Addison-Wesley Publishing, Massachusetts, USA, 1983.

[89] M. Kudo, J. Toyama, and M. Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11–13):1103–1111, 1999.

[90] S. Kwek. *Geometric Concept Learning and Related Topics.* PhD thesis, University of Illinois, 1996.

[91] K. Lagus. *Text Mining with the WEBSOM.* PhD thesis, Helsinki University of Technology, 2000.

[92] D. LaLoudouanna and M. Tarare. Data set selection - Winner of the most original NIPS-2002 contribution award. *Journal of Machine Learning Gossip*, 1:11–19, 2003.

[93] H. Lange. *Charakterisierung ökosystemarer Zeitreihen mit nichtlinearen Methoden.* Bayreuther Forum Ökologie vol. 70, University of Bayreuth / BITÖK, 1999.

[94] P. Lehtimäki. Self-Organizing Operator Maps in Complex System Analysis. Master's thesis, Helsinki University of Technology, Espoo, Finland, 2002.

[95] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[96] W. Loh and Y. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

[97] J. MacDonald, editor. *Alternatives for landmine detection.* RAND Corporation, Santa Monica, CA, 2003.

[98] D. MacKay. *Information Theory, Inference and Learning Algorithms.* MIT Press, Cambridge, MA, online edition, 2003.

[99] 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.

[100] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 3(7):507–522, 1993.

[101] N. Marwan. *Encounters With Neighbours - Current Developments Of Concepts Based On Recurrence Plots And Their Applications.* PhD thesis, University of Potsdam, 2003.

[102] J. McClelland, D. Rumelhart, and G. Hinton. The Appeal of Parallel Distributed Processing. In D. Rumelhart, J. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 3–44. MIT Press, Cambridge, 1987.

[103] W. McCulloch and W. Pitts. A logical calculus of the ideas immannent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(115), 1943.

[104] K. McGarry, J. Tait, S. Wermter, and J. MacIntyre. Rule extraction from radial basis function networks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 613–618. IEE Society, 1999.

[105] P. Meinicke and H. Ritter. Quantizing density estimators. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS)*, volume 14, pages 825–832, Cambridge, MA, 2002. MIT Press.

[106] E. Merényi and A. Jain. Forbidden Magnification? In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 51–62. D-facto Publications, 2004.

[107] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[108] R. Miikkulainen. *Subsymbolic Natural Language Processing: An integrated Model OF Scripts, Lexicon, and Memory.* MIT Press, Cambridge, MA, 1993.

[109] M. Mozer. Neural network architectures for temporal pattern processing. In A. Weigend and N. Gershenfeld, editors, *Time series prediction: Forecasting the future and understanding the past*, pages 243–264. Addison-Wesley Publishing, Redwood City, CA, 1993.

[110] S. Muggleton. Inductive Logic Programming. In *The MIT Encyclopedia of the Cognitive Sciences (MITECS).* MIT Press, 1999.

[111] H. Núñez, C. Angulo, and A. Català. Rule extraction from support vector machines. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 107–112. D-facto Publications, 2002.

[112] L. Ohno-Machado and M. Musen. Hierarchical neural networks for partial diagnosis in medicine. In *Proceedings of the World Congress on Neural Networks*, pages 291–296, San Diego, CA, 1994.

[113] E. Oja and S. Kaski, editors. *Kohonen Maps.* Elsevier Science, Amsterdam, 1st edition, 1999.

[114] J. Ontrup and H. Ritter. Hyperbolic self-organizing maps for semantic navigation. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS)*, volume 14, pages 1417–1424, Cambridge, MA, 2001. MIT Press.

[115] B. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989.

[116] J. Peltonen, A. Klami, and S. Kaski. Learning more accurate metrics for Self-Organizing Maps. In J. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 999–1004, Madrid, Spain, 2002. Springer.

[117] M. Pertea, X. Lin, and S. Salzberg. Genesplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

[118] G. Pfurtscheller and M. Pregenzer. LVQ and single trial EEG classification. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 317–327. Elsevier Science, Amsterdam, 1st edition, 1999.

[119] M. Pregenzer. *Distinction Sensitive Learning Vector Quantization (DSLVQ).* PhD thesis, Technische Universität Graz, 1996.

[120] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1992.

[121] J. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann Publishers, 1993.

[122] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, 2002.

[123] A. Reber. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6:855–863, 1967.

[124] H. Ritter. Asymptotic level density for a class of vector quantization processes. *IEEE Transactions on Neural Networks*, 2:173–175, 1991.

[125] H. Ritter. Self-organizing Maps on non-euclidean Spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–110. Elsevier, Amsterdam, 1999.

[126] H. Ritter. Self-Organizing Feature Maps. In M. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 1005–1010. MIT Press, Cambridge, MA, 2nd edition, 2002.

[127] H. Ritter and K. Schulten. Kohonen's self-organizing maps: exploring their computational capabilities. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'88)*, pages 109–116, San Diego, USA, 1988. IEEE Service Center.

[128] T. Sanger. Optimal unsupervised learning in a singlelayer linear feed forward neural network. *Neural Networks*, 2:459–473, 1989.

[129] A. Sato and K. Yamada. Generalized Learning Vector Quantization. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS)*, volume 7, pages 423–429. MIT Press, 1995.

[130] A. Sato and K. Yamada. An analysis of convergence in generalized LVQ. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 172–176. Springer, 1998.

[131] A. Savitzky and M. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[132] W. Schiffmann, M. Joost, and R. Werner. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, University of Koblenz, Koblenz, Germany, 1992.

[133] B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS)*, volume 10, pages 640–646, Cambridge, MA, 1998. MIT Press.

[134] B. Schölkopf and A. Smola, editors. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[135] U. Seiffert and L. Jain. *Self-Organizing Neural Networks – Recent Advances and Applications*, volume 78 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, 2002.

[136] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2003.

[137] J. Sinkkonen and S. Kaski. Clustering based on conditional distribution in an auxiliary space. *Neural Computation*, 14:217–239, 2002.

[138] S. Slade. Case-based reasoning: A research paradigm. *AI Magazine*, 12:42–55, 1991.

[139] P. Somervuo. *Self-Organizing Maps for Signal and Symbol Sequences*. PhD thesis, Helsinki University of Technology, 2000.

[140] P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999.

[141] S. Sonnenburg. New Methods for Splice Site Recognition. Diplom thesis, Institut für Informatik, Humboldt-Universität zu Berlin, 2002.

[142] S. Sonnenburg, G. Rätsch, A. Jagota, and K. Müller. New Methods for Splice Site Recognition. *Lecture Notes in Computer Science*, 2415:329–336, 2002.

[143] M. Strickert. Treatment of Time Series from Ecosystems: Analysis and Modelling by the Example of Daily Rainfall and Runoff Data Recorded at St. Arnold, Germany. Diplom thesis, Institute of Environmental Systems Research, University of Osnabrück, 2000.

[144] M. Strickert, T. Bojer, and B. Hammer. Generalized relevance LVQ for time series. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 677–683. Springer, 2001.

[145] M. Strickert and B. Hammer. Neural Gas for Sequences. In T. Yamakawa, editor, *Proceedings of the Workshop on Self-Organizing Networks (WSOM)*, pages 53–58, Kyushu Institute of Technology, 2003.

[146] M. Strickert and B. Hammer. Unsupervised recursive sequence processing. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 27–32. D-side Publications, 2003.

[147] M. Strickert and B. Hammer. Self-organizing context learning. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 39–44. D-facto Publications, 2004.

[148] A. Tickle, R. Andrews, M. Golea, and J. Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*, 9(6):1057–1068, 1998.

[149] P. Tino and G. Dorffner. Predicting the future of discrete sequences from fractal representations of the past. *Machine Learning*, 45(2):187–217, 2001.

[150] G. Towell and J. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, 1993.

[151] G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1–2):119–165, 1994.

[152] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K. Müller. A new discriminative kernel from probabilistic models. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS)*, volume 14, pages 977–984, Cambridge, MA, 2002. MIT Press.

[153] A. Ultsch. Knowledge Extraction from Self-Organizing Neural Networks. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification*, pages 301–306, London, UK, 1993. Springer.

[154] A. Ultsch. Maps for the Visualization of high-dimensional Data Spaces. In T. Yamakawa, editor, *Intelligent Systems and Innovational Computing*, pages 225–230. Kyushu Institute of Technology, 2003.

[155] M. van Hulle. *Faithful Representations and Topographic Maps*. John Wiley & Sons, New York, 2000.

[156] M. Vannucci and V. Colla. Meaningful discretization of continuous features for association rules mining by means of a som. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 489–494. D-facto Publications, 2004.

[157] M. Varsta, J. Heikkonen, and J. del R. Millan. Context learning with the self organizing map. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM)*, pages 197–202. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.

[158] M. Varsta, J. Heikkonen, and J. Lampinen. Analytical comparison of the temporal kohonen map and the recurrent self organizing map. In M. Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 273–280. D-facto Publications, 2000.

[159] T. Villmann. *Topologieerhaltung in selbstorganisierenden neuronalen Merkmalskarten*. PhD thesis, Universität Leipzig, 1996.

[160] T. Villmann and J. Claussen. Investigation of magnification control in self-organizing maps and neural gas. In T. Yamakawa, editor, *Proceedings of the Workshop on Self-Organizing Networks (WSOM)*, pages 59–65, Kyushu Institute of Technology, 2003.

[161] 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.

[162] T. Villmann, B. Hammer, and M. Strickert. Supervised neural gas for learning vector quantization. In D. Polani, J. Kim, and T. Martinetz, editors, *Proceedings od the Fifth German Workshop on Artificial Life*, pages 9–18. IOS Press, 2002.

[163] T. Villmann, E. Merenyi, and B. Hammer. Neural maps in remote sensing image analysis. *Neural Networks*, 16(3–4):389–403, 2003.

[164] T. Villmann, F. Schleif, and B. Hammer. Supervised Neural Gas and Relevance Learning in Learning Vector Quantization. In T. Yamakawa, editor, *Proceedings of the Workshop on Self-Organizing Networks (WSOM)*, pages 47–52, Kyushu Institute of Technology, 2003.

[165] F. Virili and B. Freisleben. Nonstationarity and data preprocessing for neural network predictions of an economic time series. In *Proceedings of the IEEE 2000 International Joint Conference on Neural Networks*. IEEE Press, 2000.

[166] T. Voegtlin. *Neural Networks and Self-Reference*. PhD thesis, Université Lyon 2, 2002.

[167] T. Voegtlin and P. Dominey. Recursive self-organizing maps. *Neural Networks*, 15(8–9):979–991, 2002.

[168] S. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 781–787, Detroit, MI, 1989. Morgan Kaufmann Publishers.

[169] D. Wettschereck, D. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):273–314, 1997.

[170] R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. Report 8805, Institute for Cognitive Science, University of California, San Diego, La Jolla, CA, 1988.

[171] C. Wu, H. Chen, and S. Chen. Counter-Propagation Neural Networks for Molecular Sequence Classification: Supervised LVQ and Dynamic Node Allocation. *Applied Intelligence*, 7(1):27–38, 1997.

[172] F. Wysotzki, W. Müller, and B. Schulmeister. Automatic construction of decision trees and neural nets for classification using statistical considerations. In G. Della Riccia, H. Len, and R. Kruse, editors, *Learning, Networks and Statistics – CISM Courses and Lectures No. 382*. Springer, New York, 1997.

[173] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS)*, volume 15, pages 505–512. MIT Press, Cambridge, MA, 2003.

[174] P. Zador. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, 28:139–149, 1982.

[175] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K. Müller. Engineering Support Vector Machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.

Index