Ma 107

Self-Organizing Maps for Signal and Symbol Sequences

PANU SOMERVUO

Helsinki University of Technology
Neural Networks Research Centre
P.O. Box 5400
FIN-02015 HUT
Finland

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Auditorium T1 at Helsinki University of Technology (Espoo, Finland) on the 17th of November, 2000, at 12 o'clock noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science

ESPOO 2000

Somervuo, P., **Self-Organizing Maps for Signal and Symbol Sequences**. Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 107, Espoo 2000, 45 pp. Published by the Finnish Academy of Technology. ISBN 951-666-550-0. ISSN 1456-9418. UDC 681.327.12:534.75

Keywords: neural networks, Self-Organizing Map, sequence processing, speech recognition

ABSTRACT

The topic of this thesis is the application of Kohonen's Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ) algorithms to the processing of data sequences with numeric or symbolic elements. Usually the models of the data are fixed-dimensional feature vectors in these algorithms. In this work also other models have been experimented like hidden Markov models, feature vector sequences, and symbol sequences. The goal has been to utilize the principles of the unsupervised learning of the SOM and the error corrective learning of the LVQ while also taking into account the sequential nature of the data.

The developed methods have been applied to unsupervised segmentation of speech, generating prototype templates for variable-length feature vector sequences, improving the performance of a speech recognizer based on hidden Markov models, constructing a data-driven pronunciation dictionary for speech recognition, and clustering of all currently known protein sequences.

PREFACE

This work has been carried out in the Neural Networks Research Centre, Helsinki University of Technology, during 1996-2000. I am obliged to Academian Teuvo Kohonen for his valuable contributions to the thesis, guidance to the world of Self-Organizing Maps, and for providing excellent research facilities in the laboratory.

I wish to thank all the personnel of the Neural Networks Research Centre and the Laboratory of Computer and Information Science for sharing daily conversations and creating a friendly and inspiring working atmosphere. Dr. Adrian Flanagan deserves special thanks for checking the language of the thesis.

The financial support of Emil Aaltonen Foundation and Nokia Foundation is gratefully acknowledged.

Espoo, May 2000

Panu Somervuo

CONTENTS

LIST OF PUBLICATIONS

The thesis consists of an introduction and the following publications:

P1. Panu Somervuo (1999). Redundant Hash Addressing of Feature Sequences Using the Self-Organizing Map. *Neural Processing Letters*, 10(1): 25–34.

P2. Panu Somervuo and Teuvo Kohonen (1999). Self-Organizing Maps and Learning Vector Quantization for Feature Sequences. *Neural Processing Letters*, 10(2): 151–159.

P3. Panu Somervuo (2000). Competing Hidden Markov Models on the Self-Organizing Map. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 169–174, Como, Italy.

P4. Panu Somervuo (1999). Time Topology for the Self-Organizing Map. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 1900–1905, Washington D.C.

P5. Panu Somervuo (2000). Speech Recognition Using Temporally Connected Kernels in Mixture Density Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3434–3437, Istanbul, Turkey.

P6. Teuvo Kohonen and Panu Somervuo (1998). Self-Organizing Maps of Symbol Strings. *Neurocomputing*, 21:19–30.

P7. Panu Somervuo and Teuvo Kohonen (2000). Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map. Technical Report A58, Helsinki University of Technology, Laboratory of Computer and Information Science, Finland, 10 pages. Also accepted to *Proceedings of Discovery Science 2000*, Kyoto, Japan.

In **Publication 1** the SOM was used as a codebook for transforming feature vector sequences into the sequences of best-matching unit indices. A fast associative method, Redundant Hash Addressing, was used for classifying the index sequences. The method was experimented in music retrieval and isolated word recognition. The topology-preserving property of the SOM was utilized when the feature vectors were converted into coordinate vectors of the best-matching units on the SOM grid.

In **Publication 2** the SOM algorithm was used for clustering variable-length feature vector sequences instead of traditionally used single feature vectors. The idea came from the thesis author motivated by the work with the SOM of symbol strings introduced by Kohonen in 1996. The method was experimented in the clustering of word utterances. LVQ was used for fine tuning the prototype sequences in a speaker-independent word recognition task.

In **Publication 3** an unsupervised segmentation method was presented. The models associated with the nodes of the SOM were single- and multi-state hidden Markov models. The segment models derived in an unsupervised manner were utilized for constructing a pronunciation dictionary for speech recognition.

In **Publication 4** the regular map grid was replaced by the node connections learned from temporal input data. This resulted in the network where the node connections represent the most frequent temporal signal paths between the local models of the input domain.

In **Publication 5** the idea of Publication 4 was utilized for speeding up the performance of the phonemic speech recognizer based on mixture density hidden Markov models.

In **Publication 6** the SOM and LVQ algorithms were extended for symbol strings. The idea was introduced by Kohonen in 1996. The thesis author was responsible for implementing the algorithms and conducting the practical experiments. The method was demonstrated by constructing a pronunciation dictionary for a given speech recognizer. The SOM was used for clustering phoneme sequences and LVQ was used for fine tuning the resulting prototype sequences. The solution for solving the ties in the best-matching unit search was proposed by the thesis author.

In **Publication 7** the SOM of symbol strings was applied to the clustering of all currently known protein sequences. The thesis author was responsible for the idea and the implementation of the application.

LIST OF ABBREVIATIONS AND SYMBOLS

ANN	Artificial neural network
ASR	Automatic speech recognition
BMU	Best-matching unit
DNA	Deoxyribonucleic acid
DP	Dynamic programming
DTW	Dynamic time warping
EM	Expectation maximization
GCS	Growing Cell Structure
GPD	Generalized Probabilistic Descent
HMM	Hidden Markov model
LVQ	Learning Vector Quantization
MAP	Maximum a posteriori probability
ML	Maximum likelihood
MLP	Multi-layer perceptron
PDF	Probability density function
RBF	Radial basis function
RHA	Redundant Hash Addressing
SOM	Self-Organizing Map
TDNN	Time Delay Neural Network
VQ	Vector quantization

α	learning gain
a_{ij}	transition probability from state i to state j
$b_i(\cdot)$	probability model of state i
$c(x)$	index of the best-matching unit for input x
$\delta(\cdot)$	Kronecker delta
$d(\cdot,\cdot)$	distance measure
$h_{i,j}$	neighborhood function
m_i	general model associated with map unit i
$\mathbf{m}_i$	model vector associated with map unit i
n	number of N-grams
$p(x)$	probability density of x
$\boldsymbol{\pi}$	vector of initial state probabilities
$\mathbf{q}$	state sequence
t	time index
x	input
$\mathbf{x}$	input vector
E	average expected distortion measure
$\mathbf{O}_t$	observation vector at time t
$P(\cdot\|\cdot)$	conditional probability
V_i	Voronoi cell of map unit i

1 INTRODUCTION

Representations of events and objects captured by the sensory systems are abstractions of the real phenomena. How the representations emerge is an issue related to modeling and learning. Modeling in general means making constructions and descriptions of the world. The system can use the models for representing the world, making predictions, and controlling its behavior. Learning can be defined as a process where the system adapts its current representations based on new sensory information. The ability to interact with the world, get observations, and make inferences about the observations constitutes the basis of learning. This holds for both biological and artificial systems.

The learning and modeling problem is traditionally divided into two parts: problem specification and parameter estimation. Specification involves the determination of the parametric form of the unknown underlying distribution of the problem. The structure of the model must be defined. Estimation is the determination of the values of the resulting parameters. When solving engineering problems, the specification part of the modeling process usually requires a considerable amount of application-dependent knowledge.

Since biological neural systems are extremely capable of performing a variety of pattern recognition and learning tasks - human beings are a living proof of that - it is worth investigating whether the underlying principles of these systems could also be utilized when developing new tools for engineering problems. Although there are many unanswered questions like how the information is exactly represented and how learning really takes place in the biological systems, all details need not be necessarily copied in order to make a well-performing engineering application. Artificial neural networks (ANNs) try to imitate some aspects of the information processing mechanisms found in nature. ANNs can be loosely defined as systems consisting of a large number of simple adaptive interconnected computing elements. Depending on how these elements are connected and trained leads to different kinds of ANN architectures and algorithms. ANNs are discussed in many textbooks and reviews, see e.g. (Haykin, 1994; Bishop, 1995; Kohonen, 1995; Cherkassky and Mulier, 1998; Kohonen, 2000).

The topic of this thesis is processing of sequences, by means of a specific artificial neural network, the Self-Organizing Map (SOM). The SOM algorithm was proposed by Professor Teuvo Kohonen in the early 1980's (Kohonen, 1981; Kohonen, 1982). Many cortical areas in the brain are topographically ordered meaning that the representations of similar inputs are located close each other. These kinds of representations can be effectively generated by using the SOM algorithm. The SOM is an interesting mathematical construction as such but it has also many practical engineering applications.

Instead of being a single algorithm, the SOM can be viewed as a more general framework for various information processing tasks. SOM-based models can be constructed for both static objects and dynamic processes. Since things in nature are seldom static, isolated objects, but instead, they evolve and change in time, the development of methods and tools for investigating sequential processes is a well motivated task with a wide range of applications. In this thesis, the SOM is used as a framework when constructing models for various sequence processing tasks. Choosing the best parametrization and structure of the model in each application has been left outside the scope of the thesis. Instead, the approach has been to use *the SOM principle* when estimating the parameters of the predefined models.

Speech recognition has been the main application and test bench of the algorithms during the work. However, the developed methods are not restricted to this application only. Many pattern recognition problems share common subtasks. For example, segmentation algorithms are needed in several applications. In the segmentation process, the stream of observations is divided into a chain of separate chunks. The SOM can be used for finding a set of models each of which will be tuned and specialized for a particular type of event without using any supervisor. Such an algorithm is presented in the thesis.

Usually the SOM has been applied to single vectorial inputs and models. The main result of this thesis is that the SOM can be used as a clustering and visualization tool also for variable-length data sequences. A recent extension of the SOM proposed by Kohonen and experimented in this thesis allows a SOM to be constructed for nonvectorial data sets, too. The data may then consist of e.g. symbol sequences.

Because the SOM gives an ordered representation of the input data in a compact form, it is especially well suited for the clustering and visualization of large data sets. Molecular biology provides potential and challenging applications. The sizes of databases consisting of DNA sequences, protein sequences, and molecule structures are continually growing. Improved software tools are needed for processing the vast amount of information. In the thesis work the new SOM method, the SOM of symbol strings, was applied to the organization and clustering of all currently known protein sequences.

2 THE SELF-ORGANIZING MAP

The Self-Organizing Map (SOM) is an artificial neural network which consists of nodes, usually located on a low-dimensional regular map grid (Kohonen, 1982; Kohonen, 1990; Kohonen, 1995; Kohonen et al., 1996). Each node is associated with a model of the local input domain. The learning is based on the competition of the models. Even if the models have random initializations, through an unsupervised learning process they organize and become selective to the input patterns. The result of the learning is a set of prototypes representing the input in an organized and ordered manner on the map grid. The SOM algorithm in its current form was presented by Professor Teuvo Kohonen in 1982 (Kohonen, 1982). The algorithm is based on repeating the following two steps for input items randomly picked from the training data:

1. Find the best-matching unit (BMU) of the map for the input item by using the chosen similarity measure.

2. Update the model of this unit and the models of its neighboring units.

The algorithm can be implemented as an online version or as a batch process. For vectorial input data the SOM can be viewed as a smoothed vector quantization (VQ) method. VQ (Linde et al., 1980; Gray, 1984; Makhoul et al., 1985) is a method for representing vector spaces by a set of codebook vectors, see Figure (1). The main difference between the SOM and other VQ algorithms, e.g. the K-means algorithm, is the ordering of the codebook which results from the use of the neighborhood when adapting the codebook vectors.

The mapping is defined from the input domain to the set of nodes in the SOM grid. The index of the best-matching unit for input item x is defined as

$$c(x) = \arg\min_i \{d(x, m_i)\}, \tag{1}$$

where m_i is the model associated with the map unit i and $d(x, m_i)$ is a distance measure defined between the input and models.

The basic iterative learning algorithm of the SOM for vectors is:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{c(\mathbf{x}_t),i}(t)[\mathbf{x}_t - \mathbf{m}_i(t)] \ , \tag{2}$$

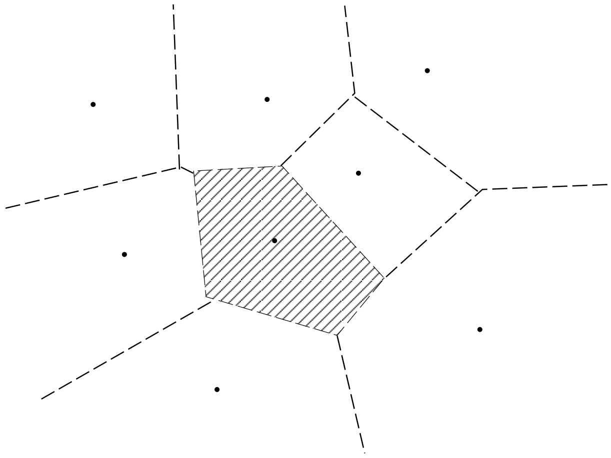


Figure 1: Vector quantization. Points represent codebook vectors and dashed lines indicate the decision borders according to the nearest neighbor classification rule. The decision borders divide the input space into Voronoi cells, one of which has been shaded with lines.

where $h_{c(\mathbf{x}_t),i}(t)$ is the neighborhood function and t is the time index. The training is continued several times, picking the input item $\mathbf{x}_t$ at each time t randomly or cyclically from the training data. The neighborhood function controls the learning gains of the local models in the network. It is usually symmetric around the best-matching unit decreasing as a function of the distance between the nodes on the map grid. The result of using the neighborhood is that the models of adjacent map units tend to be similar. The effective width and magnitude of the neighborhood function may decrease during the training process. Learning gains and neighborhood functions have been considered in (Ritter and Schulten, 1988a; Erwin et al., 1992b; Mulier and Cherkassky, 1994; Kohonen, 1995).

The final distribution of the local models after training depends both on the neighborhood function and the distribution of the input data. The size of the receptive field of each map unit, the Voronoi cell, reflects the local density of the input. Point densities of the model vectors have been analyzed in (Kohonen, 1999).

Using a fixed map structure, the number of the nodes in addition to the width of the effective neighborhood affects most the quality of the mapping. This is like in kernel density estimation, where the number of bins and the shapes of kernels affect the resulting histogram. The sizes of the Voronoi cells decrease as the total number of the nodes on the map increases.

2.1 Models, similarity measures, and learning

Matching of input and models can be defined in several ways. Different kinds of similarity and dissimilarity measures can be used. For vectorial input, the Euclidean distance or inner product are commonly used. Probability measures can also be used. Not all dissimilarity measures satisfy the properties of true mathematical metrics, e.g., the weighted Levenshtein distance (Tanaka and Kasai, 1976) used for comparing symbol strings is generally not symmetric. But for practical purposes it may be just adequate that there is a measure, even based on any heuristics, which can be used for comparing and sorting the models, and which is relevant to the specific problem.

Learning means the adaptation of the models. According to the chosen similarity or dissimilarity measure the model when adapted should better represent the input. The model adaptation can be based on some cost function, e.g. the error of representing the data. In the steepest descent method, the parameters of the model are changed in the direction of the negative gradient of the error. In online-learning, the adaptation of the models is based on the instantaneous error which leads to the stochastic approximation (Robbins and Monro, 1951). The error can also be collected from a longer training period which will usually result in smoother learning. One problem in all gradient methods, however, is the choice of the learning gain. The gradient of the error gives only the direction of the desired adaptation. The learning gain control can be completely eliminated by using fixed-point, or contractive mapping, methods. The parameter values are then determined by solving the equilibrium equation of the learning process. The Batch Map algorithm (Kohonen, 1992) is based on this idea. Using the notations of Equation (2), the new value of the model vector is

$$\mathbf{m}_i = \frac{\sum_j h_{c(\mathbf{x}_j),i}(t)\mathbf{x}_j}{\sum_j h_{c(\mathbf{x}_j),i}(t)}. \tag{3}$$

A completely different approach for adapting the models is based on evolutionary learning or genetic algorithms. In evolutionary learning, the minimization of the cost function, or maximization of the *fitness function*, can be based on any heuristics. The new model will replace the old one only if its fitness is better. In genetic algorithms (GA) (Holland, 1975; Tang et al., 1996), there are basic operations such as crossing over and mutation of the models defined for the model population in order to create new model candidates.

2.2 Extensions of the basic SOM

The SOM has its roots in biological neural systems, but not all applications require an adherence to this. In the original SOM (Kohonen, 1982), the models associated with the nodes were weight vectors. The purpose was to present a simplified and practical algorithm for constructing a topographic mapping for static feature vectors. However, the models need not be restricted to vectors. Furthermore, although in the first demonstrations of the SOM the map units were located at the nodes of a regular two-dimensional grid, also the map structure can be defined differently. For instance, due to the specific application in (Bradburn, 1989), an eight-dimensional Hamming cube with 256 nodes was used as a SOM grid.

The principles of the SOM are in competitive learning and the use of a neighborhood when adapting the models representing the local input domain. The choice for the feature extraction method, model, and (dis)similarity measure for comparing the models and data can be made in several ways. The resulting SOM then reflects the properties of these choices.

The SOM has been applied to a diversity of problems which has led to various modifications of the basic algorithm. An overview of these modifications is presented in (Kohonen, 1995).

A special case is an extension of the basic SOM for dynamic phenomena. The SOM algorithm can be used in the traditional way if the time information and the context of short-time feature vectors is simply encoded into a new feature vector. Another approach is to modify the learning algorithm itself. In (Kangas, 1990), two SOMs were used in sequence. The first SOM was used for transforming the input feature vector into an activation image of the map units. This activation pattern was then leaky-integrated and used as an input to the second SOM. In (Chappel and Taylor, 1993) and (Koskela et al., 1998), the leaky-integration of the neurons was directly embedded in the SOM algorithm.

The models associated with the nodes can be filters for dynamic signals. In the Operator Map (Kohonen, 1993; Lampinen and Oja, 1989), autoregressive (AR) models were associated with the nodes of the SOM and the competition of the models was based on the prediction errors. The adaptive-subspace SOM (ASSOM) was developed for detecting invariant features from the input (Kohonen, 1995; Kohonen et al., 1997).

A recent extension of the SOM proposed by Kohonen (Kohonen, 1996) allows the construction of the SOM for any data set, for which a similarity or distance measure between its elements can be defined. The SOM can then be applied also to nonvectorial data items. This is based on using the batch-version of the SOM with *medians* as averaged input items. The median of the data set $\mathcal{S}$ is defined as an item m which has the smallest sum of distances to other data items x_j in that set:

$$\sum_{x_j \in \mathcal{S}} d(x_j, m) = \min! \; . \tag{4}$$

The resulting SOM algorithm, *the Median SOM*, has been applied to the clustering and organization of symbol sequences, see Publication 6.

Characteristic of the SOM is the use of the winner-take-all principle (Kohonen, 1995); only one best-matching unit is selected for each input item in turn and the learning gains of other units are controlled by the neighborhood function of this unit. In the Generative Topographic Mapping algorithm (GTM) (Bishop et al., 1998), the winner-take-all principle is replaced by the expectation-maximization (EM) algorithm (Dempster et al., 1977; Moon, 1996). In the EM algorithm, each local model is adapted according to its probability to produce the data. The GTM can be considered as a smooth SOM algorithm its main advantages being the easier mathematical analysis and the use of a probabilistic framework.

2.3 Map grid and neighborhood

The SOM performs a mapping from the input space to the set of nodes in the output space. If the output space is a vector space, each map unit can be provided with a position, or a coordinate vector, there. Depending on these positions, the grid is regular or irregular. Although the word grid may imply some kind of regularity, here it is used for denoting any kind of node arrangement.

Besides defining the output domain in a vector space, another possibility is to consider only connections between the map units. The SOM can be described as an undirected graph, where vertices denote the map units and edges denote the adjacency between them. In this case the familiar Euclidean distance, like any other vector-space distance, can not be used for measuring the relative positions of the nodes in the network. Instead, the graph distance can be used, which also satisfies the properties of a true metric.

The choice for the node connections, and thus the neighborhood, affects highly the capability of the SOM to preserve the topology of the input in the mapping. Each single node contains a model representing only the local input space. The topology preservation is a demand for the network where both the local models and their neighborhood connections are taken into account. Input items which are close in the data manifold should be mapped to adjacent (or identical) map nodes.

The main dichotomy between different map grids can be made depending on whether the structure of the map grid has been predefined or whether it has been constructed by using the properties of the input data.

2.3.1 Regular map grid

Rectangular and hexagonal map grids are the two most commonly used SOM structures. In these, the node positions form a regular pattern in the output space. This is useful e.g. for visualizing the data. Investigating adjacent reference models on the map is equivalent to sampling the input space with specific intervals, determined by the distribution of the input data and the value of the neighborhood function. For a convenient visualization, the grid dimension should be sufficiently small. Higher-dimensional grids can be useful for other purposes, e.g., for finding a specific nonlinear coordinate system for the input data. In many cases, especially when processing natural data, the dimension of the actual data manifold is considerably lower than the dimension of the input feature space. The SOM can be used for performing a nonlinear dimension reduction.

2.3.2 Irregular map grid

An irregular map grid may provide a better approximation of an arbitrarily shaped input distribution compared to the predefined fixed regular grid. If the data distribution contains empty regions, gaps and holes inside the data manifold, the regular map neighborhood is likely to fill them. In (Kangas et al., 1990) the minimal spanning tree (MST) algorithm was used for constructing the map topology. The MST topology is constructed by using the distances between model vectors in the input space.

In the Neural Gas network (Martinetz and Schulten, 1991; Martinetz et al., 1993), the topology of the network is formed by creating a connection between two nearest map units of each input item. In this algorithm, the connections are not used directly for controlling the learning gains of the neighboring models. Instead, the local models are adapted according to their ranked closeness to the input items. In (Martinetz and Schulten, 1994) the correspondence between the node connections of the Neural Gas network and the Delaunay triangulation for achieving perfect topology preservation was pointed out.

2.3.3 Growing map grid

The number of the SOM units may be increased during training. The insertion and interpolation of new nodes can be applied to both regular and irregular grids. One possibility for expanding the map grid is to insert new nodes between all existing nodes thus preserving the structure of the old grid. Another approach is to add new nodes only to specific parts of the old network according to some performance criterion.

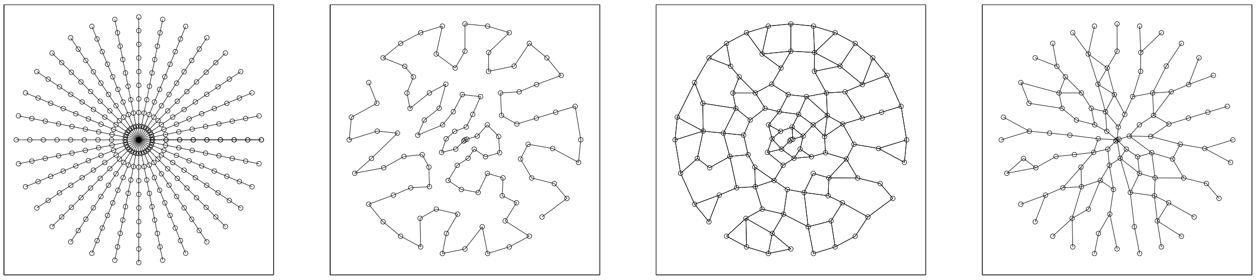
The algorithm for growing hypercubical SOMs has been presented in (Bauer and Villmann, 1997). The algorithm for the Growing Cell Structure (GCS) which forms two-dimensional (irregular) map grids has been proposed in (Fritzke, 1991). The basic structure of this network consists of three map units connected together in order to form a triangle. During the map expansion, old node connections are removed and new node connections are created so that the new inserted node will always be a part of some triangle. The generalization of this for forming arbitrary-dimensional grids has been presented in (Fritzke, 1994). In GCS, the neighborhood of the nodes is used when adapting the reference models. Growing Neural Gas has been proposed in (Fritzke, 1995b) and GCS which preserves the structure of a rectangular map grid has been presented in (Fritzke, 1995a).

Koikkalainen has proposed a tree-structured SOM which aims at speeding up the BMU search (Koikkalainen and Oja, 1990; Koikkalainen, 1994). In this method, multiple SOMs are constructed one by one and stored as the layers of the SOM tree so that each new SOM is an expansion of its ancestor.

2.3.4 Node connections representing time

Instead of creating the neighborhood connections between those two nodes which are the two best-matching units for each input item as in the Neural Gas network, the neighborhood connection can be created between those two units which are the best-matching units for two consecutive input items in time. This was experimented in Publication 4. The idea is illustrated in Figure 2.

The reference models associated with the SOM nodes are first trained in the usual way, treating the input samples as static vectors and defining the neighborhood of the map nodes on the regular map grid. Once the map has been trained, old node connections are removed and new connections are created according to the time information of the input. The SOM training can then be continued by using the new node connections as a neighborhood when adapting the reference models. The result is a network where node connections represent temporal signal paths in the input space. Since any two nodes which are the best-matching units for two consecutive input items in time can be connected independently of their Euclidean distance on the regular map lattice, the new connections may provide "worm-holes" to the original map lattice space. The neighborhood connections follow the trajectory of the BMUs.



(a) (b) (c) (d)

Figure 2: Codebooks with different kinds of neighborhood connections. Input data consist of sequences of two-dimensional feature vectors proceeding from the origin to the unit circle (a). Input items are depicted by dots and successive input items in time are connected by lines. A one-dimensional SOM with 100 nodes was constructed using this data (b). The model vectors are depicted by dots and the neighborhood connections are depicted by line segments. Figure (c) shows the connections created between two nearest nodes in the input space for each input item. Model vectors are the same as in (b). Figure (d) represents the connections created between the best-matching units of two successive input items in time. The model vectors are the same as in (b). The original input data does not consist of separate feature vectors only, but sequences of them with time information. Network (d) resembles thus clearly best the original input data (a).

2.4 Analysis of the SOM

It is interesting that although the SOM algorithm is very simple and straightforward to implement, it is surprisingly difficult to analyze. The convergence of the SOM algorithm to the ordered state has been proved only for the case where both the input and the map grid are one-dimensional. Merely the definition of order is difficult in higher dimensions. A collection of references related to the analysis of the SOM can be found in (Kangas, 1994; Cottrell et al., 1997). Mathematical problems of the SOM concentrate on the vector quantization properties, derivation of the learning algorithm from some energy function, ordering, and convergence of the model vectors.

It has been shown that in general case there does not exist an energy function for the SOM (Erwin et al., 1992a). However, if the input set consists of a finite number of items, which in practice is always the case, an energy function can be defined (Ritter and Schulten, 1988b; Kohonen, 1991). Some approaches modify the determination of the BMU in order to find the energy function: e.g. if the quantization error of the input item is replaced by the SOM distortion measure in the BMU search, it can be shown that there exists an energy function for the SOM (Heskes and Kappen, 1993). The SOM distortion measure is a sum of the quantization errors of all map units to the given input item weighted by the neighborhood function of the given node, see Equation (5).

The convergence of the one-dimensional Batch SOM in which the distortion measure was used for the BMU search has been proved in (Cheng, 1997). Conditions for the convergence of the SOM have been analyzed in (Flanagan, 1999). Conditions for the convergence of the SOM with quantized inputs and model vectors have been studied in (Thiran and Hasler, 1994).

Besides analyzing the convergence of the SOM algorithm, different measures have been proposed for quantifying the quality of the mapping. Because each measure has its own target, it is important that the concepts to be used are first defined. Order, topology preservation, and quantization error are the main concepts to be investigated. It should be emphasized that the SOM performs a mapping from the input manifold to the set of nodes in the map grid trying to preserve the topological correspondence between these two domains. It does not try to preserve the distance relations between input items which is usually the objective of multidimensional scaling (MDS) methods (Sammon Jr., 1969; Kruskal and Wish, 1978).

2.4.1 Quantization error

Quantization error measures the accuracy of the codebook in representing individual input items. It does not address the question of topology preservation nor order. Quantization error is computed as a distance from the input item to the nearest reference model and it is averaged over all input items. The minimum is achieved when the neighborhood is decreased to zero at the end of the map training. This may, however, cause discontinuities on the map, and the orderliness of the map may be lost, especially when the dimension of the data manifold is higher than the map dimension.

Adding the neighborhood function to the quantization error leads to the definition of the SOM distortion measure. Let L denote the set of indices of all map units, V_i the Voronoi cell and m_i the model associated with the map unit i, respectively, and $p(x)$ the probability density of the input x. The average expected distortion measure is then defined as (Kohonen, 1995)

$$E = \sum_{i \in L} \int_{V_i} \sum_{k \in L} h_{i,k} d(x, m_k) p(x) dx. \tag{5}$$

One way of defining the SOM is to define it as the set of m_i that globally minimizes E (Kohonen, 1995). This SOM, however, is different from that resulting from Equations (1) and (2) (Kohonen, 1999).

2.4.2 Measures for orderliness and topological correspondence

Measures suggested for quantifying the quality of the mapping when taking the neighborhood and the structure of the map grid into account are described in the following. For the orderliness it is required that the map grid does not intersect itself in the input space unless the data manifold does so. Topology preservation, instead, requires that the map grid follows the data manifold correctly. Topology preservation requires thus more than plain orderliness. In case of an N-dimensional hypercubical data manifold, the dimension of the map grid should thus be N. If the dimension of the map grid is lower, the topology of the input can not be preserved. However, using the definition of the orderliness above, the map can be ordered. As an example, let us consider a one-dimensional SOM grid (a chain of nodes) on a two-dimensional square defining the input domain. The one-dimensional map can be considered ordered, if it does not form loops in the input space. A typical map with the model vectors forming the shape of the Peano curve satisfies this.

Bauer and Pawelzik have used *the topographic product* for the evaluation of the topology preservation of the SOM (Bauer and Pawelzik, 1992). This measure makes use of k nearest map units in the input space and k nearest map units in the output space found for each map unit. How much these two sets differ is computed by means of the distances in the input space and output space, computing both distances for both sets. The information about the order of the k nearest items is not used. Index k takes all values from one to the number of map nodes minus one and the measure is averaged over all map units. Since this measure is based only on the prototype vectors of the SOM, it is clear that the topological correspondence between input and output domains can not be revealed. This measure quantifies only the orderliness of the map.

Bezdek and Pal have investigated the preservation of the relative positions of input items in the mapping (Bezdek and Pal, 1995). In essence, their *index of topological preservation* measures how well the distance order is preserved. Spearman's coefficient is used to compare two rank ordered vectors. The first vector contains the rank orders of the pairwise distances of data items in the input space and the second vector contains the rank orders of the distances in the output space.

Zrehen has considered an orderliness measure for the SOM (Zrehen, 1993). According to this measure, the map is ordered if all points in a straight line segment between the models of two neighboring map units are closer to these than any other models. Map units whose Voronoi cells intersect the hypersphere between the two neighboring units are called intruders and the measure counts the number of them.

The topographic function, proposed by Villmann et al. (Villmann et al., 1994), is somehow similar to Zrehen's measure. But in this measure, the neighborhood of the map units is utilized together with the data manifold. Hence, the topographic function is a measure for the topology preservation. The receptive field of a map unit is defined to be an intersection of its Voronoi cell and the data manifold. The neighboring map units should have adjacent receptive fields. If other map units have adjacent receptive fields, perfect topology preservation is lost. The topographic function is computed as a function of one variable which acts as a threshold for ignoring the violations of the topology preservation. In that respect this measure resembles the topographic product. The major difference is, however, in making use of the data manifold. The problem of accessing the true data manifold was tackled in (Villmann et al., 1994) by seeking two nearest map units for each input sample and then defining the area between these two map units to belong to the data manifold.

Minamimoto et al. have measured the goodness of the SOM by combining the representation of the input, the topology preservation of the map and the number of the map units (Minamimoto et al., 1995). Kaski and Lagus have also combined the representation of the input and the topology preservation (Kaski and Lagus, 1996; Kaski, 1997). In (Minamimoto et al., 1995) several commensurable quantities were combined with individual weights, whereas in (Kaski and Lagus, 1996) the representation of the data and the topology preservation of the map were both measured in terms of the quantization error in the input space. This is advantageous, because finding appropriate weights for different types of measures to be combined may be difficult. The measure for the topology preservation in (Kaski and Lagus, 1996) was based on computing the sum of distances between the reference vectors of the adjacent map units along the continuous path on the map grid between the best-matching unit and the second-best-matching unit of each input item. This part of the measure is similar to the distance proposed by Kraaijveld et al. in (Kraaijveld et al., 1995). In (Kaski and Lagus, 1996) it was used together with the quantization error of the input.

Unfortunately, none of the aforementioned measures can be used for describing the degree of topological correspondence reliably in practice. Although the definition of the topographic function for the topology preservation is valid, in practice it is difficult to construct the input manifold from the samples, which is needed for this measure. In order to use the topographic function successfully, the prototype vectors should be dense in the input manifold so that a triangle formed by any input item and its best- and second-best-matching units would lie completely in the input manifold (Villmann et al., 1997). With a limited number of map nodes, however, this can not be guaranteed. An example of a difficult case is a two-dimensional data manifold forming a knot-like structure in a three- or higher-dimensional input space which a two-dimensional map structure follows accurately. It is very difficult to detect or measure the topological correspondence between the input and output domains even though it is perfect in this case.

Some of the abovementioned measures have been compared using several data sets in (Herrmann et al., 1997).

2.5 Benefits of ordered representations

Ordered representations can be utilized in several applications. The main application of the SOM is the visualization of large data sets (Kohonen, 1995; Kaski, 1997). The SOM can give insights to the data structure.

Another application, where the topological ordering can be utilized, is speeding up the database search. A tree-structured SOM (Koikkalainen and Oja, 1990; Koikkalainen, 1994) can be used for this purpose.

In (Bradburn, 1989), the SOM grid was constructed so that every single bit error shifted the mapping of the input vector into a neighboring node. The method was experimented in speech coding. The same idea has been applied also to image coding (Kangas, 1995).

Even if the ordered representation of the local models is not needed, the SOM provides a good method for initializing the models. The use of the neighborhood helps the local models to move to those regions of space where most of the data is. All models are taken already from the beginning of the training into use and the whole set of them is effectively utilized. The K-means or LVQ algorithm can then be used for fine tuning the models in order to increase the representation or recognition accuracy. The SOM has been used e.g. in speech recognition for initializing the Gaussian mean vectors of the mixture density hidden Markov models (Kurimo, 1997).

2.6 Learning Vector Quantization

The local models on the SOM adapt to the density of the input vectors at the same time trying to preserve also the topological relations of the input. The density approximation, however, is not necessarily the best method for discriminating data items from different classes. For that purpose another method, the Learning Vector Quantization (LVQ) algorithm, has been developed (Kohonen, 1986; Kohonen, 1995). While the basic SOM algorithm is based on unsupervised learning, LVQ is based on supervised learning. This means that the class identifiers of the training data items must be known before the training. While the LVQ has originally been developed for an effective numerical algorithm, the principle underlying it can be shown to be a special case of the Generalized Probabilistic Descent (GPD) algorithm (Katagiri et al., 1991).

In the LVQ method, the model vectors are used for directly defining the class borders according to the nearest-neighbor classification rule.

The learning rule of the basic LVQ (LVQ1) is

$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) + \alpha(t)[\mathbf{x}_t - \mathbf{m}_c(t)] \,, \tag{6}$$

if $\mathbf{x}_t$ and $\mathbf{m}_c$ belong to the same class, and

$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) - \alpha(t)[\mathbf{x}_t - \mathbf{m}_c(t)] \,, \tag{7}$$

if $\mathbf{x}_t$ and $\mathbf{m}_c$ belong to different classes. $\mathbf{m}_c$ is the nearest model vector to the given input vector. In LVQ1 no other models are adapted. In LVQ2 and LVQ3, *two* nearest model vectors are adapted according to their class identifiers using Equations (6) and (7) (Kohonen, 1995).

3 SEQUENCE PROCESSING

Sequences, both numeric and symbolic, are encountered in various applications and fields of science like time series analysis, molecular biology, geology, computer science, music, speech, language, and communication in general. A sequence can be defined as a continuous or connected series of events or objects.

Temporal sequences arise from various kinds of sources in the nature. Sensory elements transform the events into measurements and corresponding feature vectors. Methods for comparing sequences are needed in retrieval, error correction, and recognition tasks. Due to the durational differences in the feature sequences and the variation and noise in the feature vectors, both temporal and spatial fluctuations must be tolerated in the comparison. An introduction to the sequence comparison can be found in (Sankoff and Kruskal, 1983).

3.1 Comparison of sequences

The comparison of two sequences is in general based on the measure for comparing the two instantaneous values, i.e., the elements of the sequences. A simple, naive way is to use the Euclidean or Hamming distance between the sequences. However, this can be used only when the lengths of the sequences are equal. If the lengths differ, the sequences can be converted into fixed-sized histogram vectors which then can be compared using the Euclidean distance or inner product. Neither of these approaches, however, are applicable in general. The lengths of the sequences may vary and the order of the elements usually does contain information. A better, more fundamental comparison method takes both of these sequence properties into account. The differences of the sequences can be characterized by the compressions, expansions, deletions, insertions, and substitutions of the elements.

The mathematical background of the comparison of two sequences when taking the order of the elements into account can be traced to the *calculus of variations*. In this branch of mathematics the objective is to find a function for which the value of a certain integral is either the largest or the smallest possible. Dynamic programming (DP) is a nonlinear optimization method related to this (Bellman, 1957). It is based on a stage-wise decision process where the global optimum is obtained by a chain of actions, in each stage utilizing the results of the previous decisions and ignoring the effect of the future ones. Dynamic programming has been rediscovered many times by engineers, biologists, and computer scientists. It was first applied to speech recognition and molecular biology around 1970 (Vinstyuk, 1968; Needleman and Wunsch, 1970; Sakoe and Chiba, 1978). DP can be used for finding optimal paths in a graph and computing time- or length-normalized distances between numeric or symbolic sequences efficiently. The sequences can be matched using their full lengths, or finding the best-matching subsequences.

In dynamic time warping (DTW), see e.g. (Sakoe and Chiba, 1978), the objective is to find a *warping function* which optimally matches two (or more) feature vector sequences. Timing differences of the sequence elements are eliminated by warping the time axes of the sequences nonlinearly so that the maximum coincidence between the sequences is attained. A time-normalized distance between the variable-length feature sequences can then be calculated as a sum of distances between sequence elements along the warping function. Its counterpart for symbol sequences is the Levenshtein distance, also called the edit distance (Levenshtein, 1966). It is defined as the smallest number of insertions, deletions, and substitutions of symbols required for converting one sequence to another. Different edit operations can also have different costs leading to the weighted Levenshtein distance (Tanaka and Kasai, 1976). The Viterbi algorithm (Viterbi, 1967; Forney, 1973) is a DP-based method for finding the maximum probability path in a state network.

Significant speedups can be gained, if instead of seeking a global optimum, the search space is reduced by using some heuristics. The optimal solution can be sought around some initial solution. For example, DTW can be restricted to the narrow band around the linear time warping path. The same holds also for the Levenshtein distance. In the Viterbi beam search (Lowerre, 1976) those candidates whose probability scores do not exceed a given threshold are not investigated further. The Viterbi algorithm is based on the breadth-first search. Another approach is stack decoding (Jelinek, 1969; Bahl et al., 1983) which is based on the depth-first search and the A* algorithm (Nilsson, 1971). Like the Viterbi beam search, stack decoding can be used when the exhaustive search of the candidate sequences due to the combinatorial growth simply cannot be performed. This is the case e.g. in the large vocabulary continuous speech recognition (LVCSR). Search issues related to the LVSCR are discussed in (Jelinek, 1997; Ney and Ortmanns, 1999; Deshmukh et al., 1999).

Comparison and alignment of sequences can be speeded up also by using lookup tables. When the sequences are symbol strings, the lookup tables consist of the alphabet of the strings. Using feature vector sequences, the elements of the sequences must first be vector-quantized, see e.g. Publication 1.

3.2 Language processing

Natural language processing can be approached from many perspectives. Language can be defined as a method for transferring the information of an idea into a serial form. In the spoken language, acoustic events follow each other in one-dimensional time. In the written language, symbols follow each other position by position again in one dimension. In both cases, the data consist of sequences.

When constructing pattern recognition systems for signal-to-symbol transformation tasks, a bottom-up approach is a natural choice. In higher levels of language processing, however, the case is different. It seems easier to approach the structure of language using top-down methods. Nevertheless, it is tempting to try to grasp the abstractions and structure of the language using also the bottom-up approaches.

Some connectionist methods have been experimented in the field of language processing. Recurrent networks have been used for extracting the structure of linguistic sequences (Elman, 1990). In (Gorin et al., 1991) the goal was to learn the meaning of the input message during a dialogue by creating feedback-controlled associations between unrestricted textual input and output actions. The SOM has been used for extracting semantic categories from textual data (Ritter and Kohonen, 1989). The SOM has also been used for constructing prototypes of symbol strings, see Publication 6.

3.3 Speech recognition

Speech recognition can be defined as the mapping from the continuous acoustic signal to the set of discrete symbols. The general goal is to understand normal human speech and then to be able to perform some task based on this understanding.

The interaction between the speaker and listener is described in Figure 3. First an idea which has arisen in the mind of the speaker is put into words. These words are then converted into acoustic signals using the articulatory system of the speaker. Acoustic signals consist of pressure changes which propagate in the air. When these pressure waves reach the ears of the listener, they activate first the auditory nerves. After that the activated parts of the brain cause the listener to associate the perception to some image or idea in her mind. The information content has thus been extracted from the air pressure waves. In the automatic speech recognition (ASR) system, the listener is replaced by a machine.

The smallest meaningful unit in the spoken language is a *phoneme*. It does not have any meaning alone, but it does allow different words to be discriminated. A realization of a phoneme, i.e., the physical sound produced when a phoneme is uttered, is called a *phone*.

There are several reasons for the difficulties in speech recognition. Natural speech has variations in many levels. In addition to that different speakers have different voices, there is also considerable variation in the voice of a single speaker. In normal conversation, some parts of the words may be emphasized more than others depending on the context. The loudness and the pitch of the voice may change and also the speaking rate may vary. Even if the speaker tries to use as steady a voice as possible, no two uttered sounds are generally equal. Speech can be characterized as a nonstationary signal where the transients may carry more information than the stationary parts.

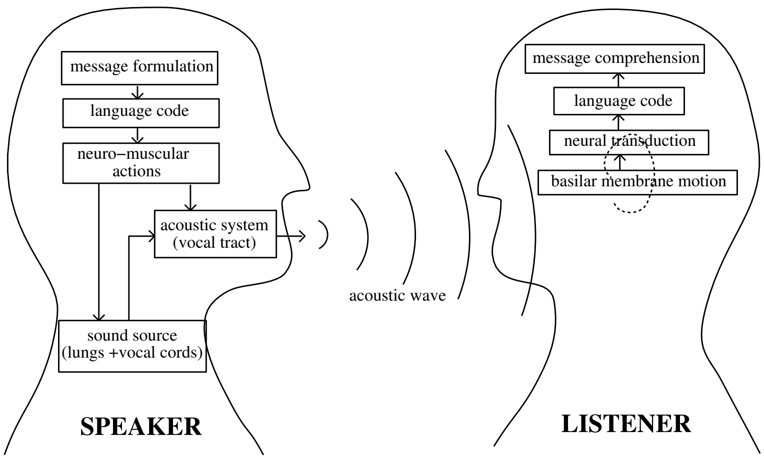


Figure 3: Communication by speech.

Figure 4 represents the block diagram of a typical speech recognition system. First stages of this kind of pattern recognition system involve more signal to symbol conversions. In the later stages of the recognition, the need for symbol processing becomes more dominant. The two first blocks (labeled as preprocessing and classification) perform the signal to symbol transformation. The ensuing block (postprocessing) stands for selecting the final word string. The language processing element may be used for generating hypotheses which are fed back to the classification block for being evaluated.

In the signal-to-symbol transformation level, the whole state space can be exhaustively searched using dynamic programming. In the higher levels, however, the combinatorial growth of the recognition result alternatives forces the recognition to be limited to testing only some feasible number of hypotheses. The quality of the final recognition result is then determined by the quality of the hypothesis generation mechanism.

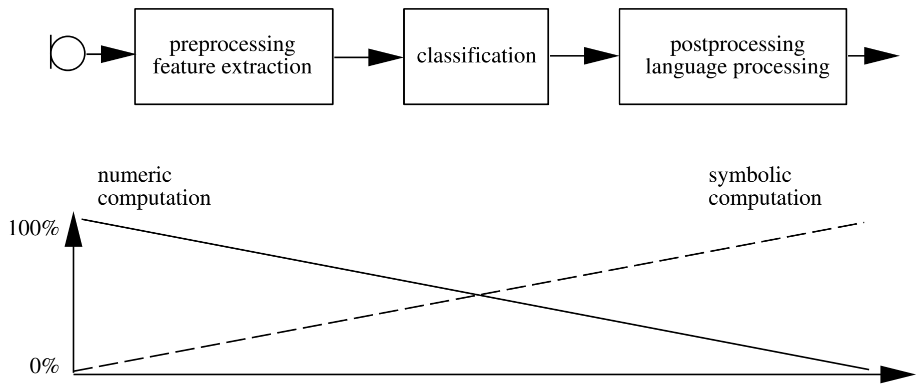


Figure 4: A block diagram of a speech recognition system. The borders of the blocks need not be fixed and possibly a considerable amount of feedback between the blocks is required in order to make a flexible and well-performing system. The main idea of this simplified block diagram is to show that speech recognition involves signal processing, pattern matching, and language processing.

In the speech recognition system, some limitations are usually made concerning the nature of the speech to be recognized. These limitations may include the number of the speakers, the size of the vocabulary, the amount of the noise in the speech, and the assumption that the input will always be speech. Whether the speech recognizer has been trained for a single speaker, for a known speaker group, or for a totally unknown speaker, affects the performance due to the increasing variation in the input signal. Background noise must be taken into account if the speech recognizer is to be used outside laboratory conditions. In some systems, all these aspects have purposely not been taken into account. The development and research may have been concentrated only on a specific subtask. For example, the speech recognition may have been used only as a test bench for comparing different pattern recognition algorithms or signal-to-symbol transformations.

Humans are clearly the best benchmark for comparing speech recognition systems. We can easily tolerate erroneous speech and replace missing parts of it by using our knowledge about the language, the speaker, the context of the speech and the meaning of the words. Therefore it may be useful to try to understand the mechanisms of the human speech recognition system. An interesting question is how much one should copy from biological neural networks in order to build a good speech recognizer. The tonotopical representations of speech sounds constitute one existing part of the human speech recognition system. These kinds of representations can be constructed by the SOM algorithm. However, tonotopical representations alone are not enough, if the goal is to construct an entire speech recognition system. The system should also have knowledge about the structure of the language.

3.3.1 History and current status of speech recognition methodology

The first task when making a speech recognition system is to define the speech units to be recognized. These can be linguistic units such as phones, diphones, triphones, syllables, words, or sentences. Also data-driven segment units can be used. Larger speech units can be constructed by means of lower-level units. For example, word recognition can be based on recognizing phonemes. An additional pronunciation dictionary is then needed in order to map the phoneme sequences into words and vice versa.

The speaking style divides speech recognition into isolated or continuous speech recognition. Word dictation can be considered as an intermediate form of these. The speaker holds then pauses between the words. Keyword spotting represents also one category of the recognition. In that, the task of the recognizer is to detect predefined words from the stream of arbitrary speech.

Most artificial speech recognition systems are based on computing *feature vectors* from the overlapping time windows of speech (Picone, 1993). The speech frames are usually extracted at constant time intervals. The role of the feature extraction method in the pattern recognition system is very important. Although the bottleneck of the current speech recognizers is not necessarily the preprocessing of the speech signal, more biologically plausible preprocessing methods may be beneficial in the future. However, the most important thing is to achieve a balance between different parts of the recognition system. Optimizing only one part or copying it from the existing biological system does not guarantee the improvement of the whole system.

The most generally used feature vector consists of cepstral coefficients computed from the mel-scaled power spectrum of a speech frame (Davis and Mermelstein, 1980). First- and second-order time derivatives can also be concatenated to this feature vector. Some feature extraction methods try to mimic the human auditory system (Ghitza, 1994; Morgan and Scofield, 1991). However, in current speech recognition systems they have not clearly outperformed the standard mel-cepstrum unless the environment is very noisy (Jankowski Jr. et al., 1995).

The speech signal converted into a feature vector sequence can be further processed in various ways. In template matching, the feature sequences are matched against reference sequences. If distance measures are used, the recognition method is usually DTW which computes the time-normalized distance between two feature vector sequences by means of dynamic programming. If the reference sequences are state sequences with transition probabilities associated between states and probability densities associated with the states, the dynamic programming approach can be used for computing the probability of each reference sequence. The reference sequences need not be represented separately, but they can form a single network allowing the recognition of continuous speech. There are various methods for constraining the DP search of these kinds of *pronunciation networks* by means of thresholds for cutting off unlikely paths, and grammars for generating hypotheses (Jelinek, 1997).

DP algorithms utilize the information and order of the whole sequence in recognition. Another approach is to divide the input sequence into parts and recognize the resulting segments separately. The joining and post-processing of these separate recognition results can then be carried out at the higher level of the recognition system. DP can be used for decoding the corresponding sequence of segments (Zue et al., 1990; Leung et al., 1992). Redundant Hash Addressing method (Kohonen and Reuhkala, 1978) has been used as a fast alternative for DP in this task (Kohonen et al., 1984).

DTW-based methods represented the mainstream of the ASR in the early 1970's. Hidden Markov models (HMMs) (Rabiner, 1989) replaced them during the 1970's and early 1980's. This was mainly because of the existing training algorithms for the HMMs and the whole recognition process could be represented in a single probabilistic setting; the acoustic and linguistic subtasks could be combined in a reasonable way. Currently, HMMs are used in almost all large-scale ASR systems. It is worth noting, however, that the HMMs give only a framework for the speech recognition. The combination of improved signal representations, structures of HMMs, learning algorithms, and subword models can reduce the error rate of a continuous speech recognizer by an order of magnitude (Lee, 1989). On the other hand this also means that the plain use of HMMs does not guarantee the good performance of the ASR. The models must be also properly chosen and trained.

An example of a well-performing ASR system is the SPHINX speech recognizer (Lee, 1989). This system gave the best result in a specific speaker-independent continuous word recognition task in its time. The results were even better than those of the previously best systems operating in a speaker-dependent mode in the same task.

In 1994, several dictation-style large-vocabulary recognizers were evaluated, the participating systems including those of AT&T Bell Laboratories, BBN, Boston University, the CUED ABBOT group, the CUED HTK group, IBM, LIMSI-CNRS, Philips, and SRI International. The lowest word-error rate (by the HTK recognizer) was 7 % using a 65,000-word vocabulary (Young, 1996).

There is active research concentrated on developing better language models and methods for dialogue systems. However, this does not mean that any subtasks of the current ASR systems before language processing would have been solved optimally yet. Current ASR systems perform well only in restricted application domains and environments. But as the technology improves, more and more demanding tasks can be tackled. Some recent work has been done for developing methods for transcribing broadcast news into a written form. In the 1998 DARPA-sponsored evaluation, the lowest word-error for a broadcast news task was 14 % (Pallett et al., 1999).

3.3.2 Hidden Markov models

A hidden Markov model (HMM) is a stochastic signal model. The landmark paper was published by Baum in 1972 (Baum, 1972). The strength of the HMM method lies in its mathematical framework (theory) and implementational structure (practice). For an overview of HMMs in speech recognition, see (Rabiner, 1989; Juang and Rabiner, 1991; Furui and Sondhi, 1992). Besides speech recognition, HMMs have been applied also to natural handwriting recognition, text analysis, coding theory, ecology, and molecular biology.

A Markov process or chain is a system, which is in one of N states at any discrete time instant t. The state variable is denoted by q_t. Generally, a full description of the system would require knowledge about the entire history of the state sequence, but in a first-order Markov process the current state of the system depends only on the previous state. When the state transition probabilities are time independent, the process can be described by a state transition probability matrix $A = [a_{ij}]$, where

$$a_{ij} = P(q_t = j | q_{t-1} = i) \quad , \quad i, j = 1, \dots, N \ . \tag{8}$$

The initial state of the system is defined by a probability vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_N]$. For any state sequence $\mathbf{q} = (q_0, q_1, \dots, q_T)$, the probability of being generated by the Markov process is then

$$P(\mathbf{q}|A, \boldsymbol{\pi}) = \pi_{q_0} a_{q_0 q_1} \cdots a_{q_{T-1} q_T} \ . \tag{9}$$

A Markov process is called hidden, if the state sequence $\mathbf{q}$ cannot be directly observed. A one-to-one correspondence between an observation and a state does not necessarily exist, but each state can be considered to have produced the observation by a certain probability. Because of this stochastic nature of the outputs, the observations can be used only for computing the probabilities of different state sequences.

In the speech recognition task, the observations are the extracted feature vectors of the speech and each HMM state is associated with some speech unit. Speech recognition then means finding the hidden state sequence by means of the observations. An observation vector at time instant t is denoted by $\mathbf{O}_t$. The probability of $\mathbf{O}_t$ being generated by the state i is denoted by

$$b_i(\mathbf{O}_t) = P(\mathbf{O}_t | q_t = i) \ . \tag{10}$$

The probability model of each state for generating the observations can be a discrete or continuous pdf. The set of probability models $b_i(\mathbf{O}_t)$ of all states is denoted by $B = \{b_i(\mathbf{O}_t)\}_{i=1}^{N}$. The triple $\lambda = (\boldsymbol{\pi}, A, B)$ defines the hidden Markov model.

If the observations produced by the states are independent, the probability of the observation sequence $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_T)$ produced by the state sequence $\mathbf{q}$ is

$$P(\mathbf{O}|\mathbf{q}, B) = b_{q_1}(\mathbf{O}_1) b_{q_2}(\mathbf{O}_2) \cdots b_{q_T}(\mathbf{O}_T) \ . \tag{11}$$

The joint probability of $\mathbf{O}$ and $\mathbf{q}$ being produced by the model is then the product of (9) and (11):

$$P(\mathbf{O}, \mathbf{q}|\boldsymbol{\pi}, A, B) = \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{O}_t) \ . \tag{12}$$

The Viterbi algorithm can be used for computing the most likely state sequence $\hat{\mathbf{q}}$ given the HMMs and the stream of T observation vectors:

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} \boldsymbol{\pi}_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{O}_t). \tag{13}$$

HMMs can be trained by using maximum likelihood (ML) estimation. The EM-algorithm-based Baum-Welch algorithm (Baum and Petrie, 1966) and the Viterbi-search-based segmental K-means algorithm (Rabiner et al., 1986) are the two commonly used training methods. However, ML estimation does not directly aim at minimizing the recognition error, which in pattern recognition is usually of interest. Maximum mutual information (MMI) training (Bahl et al., 1986) is a more discriminative training method. Like MAP training (Gauvain and Lee, 1994), it aims at maximizing the posterior probabilities of the training sequences. Error corrective training (Bahl et al., 1988) aims at directly minimizing the recognition error of the training set. LVQ and GPD algorithms have also been used for this task (Kurimo, 1997; Chou et al., 1992).

Two assumptions in the standard HMM modeling, the independence of observations and the first-order Markov process property, are generally not valid for speech. Successive speech frames computed at short time intervals are not independent. This has led to the modifications of the basic HMM, such as the segmental HMMs (Russell, 1993; Gales and Young, 1993; Holmes and Russell, 1995; Ostendorf et al., 1996). However, even in their basic form, HMMs have been found to be very useful in practice. They are flexible in modeling sequential data.

3.3.3 Artificial neural networks in automatic speech recognition

Many ANN methods are better suited for processing separate feature vectors than variable-length feature sequences. In these static pattern recognition systems, the time information can be represented as an additional dimension of the feature vector. However, several attempts have been made to extend ANN methods to dynamic tasks. Some of the methods are just neural interpretations or architectures of the existing algorithms, like the Viterbi network (Lippmann et al., 1987) which implements the Viterbi algorithm or the HMM network (Niles and Silverman, 1990) which is a recurrent neural network implementing both the structure and formulation of an HMM.

Some methods use time warping as a preprocessing method for reducing the durational differences of the input, like the Dynamic Programming Neural Network (DPNN) (Sakoe et al., 1989). In (Katagiri and Lee, 1993), HMMs were used for time warping the input sequences, and the classification of the resulting fixed-length word representations was based on LVQ.

Multi-layer perception (MLP) networks are commonly used in pattern recognition and prediction tasks. MLPs have been shown to be universal approximators of nonlinear functions (Funahashi, 1989; Hornik et al., 1989). In the Time Delay Neural Network (TDNN) (Waibel et al., 1989), a predefined number of concatenated feature vectors is used as an input of the MLP. The performance of the TDNN is based on recognizing the temporal structure of the fixed-sized input window. Despite the lack of time warping, TDNNs have been reported to give excellent results. The drawback of the method is the computationally intensive training. Concatenated feature vectors have been experimented also with the SOM (Kangas et al., 1990) and LVQ (McDermott and Katagiri, 1989; Mäntysalo et al., 1994). The use of the context of short-time feature vectors has improved the recognition rates considerably.

Recurrent networks (Jordan, 1986; Elman, 1990) can be characterized by having both feed-forward and feedback connections. The activity of the network is integrated over time by means of feedback connections. The previous activation of the network and the entire history of the sequence has effect on the current activation of the network. It has been shown that the recurrent networks are universal approximators of dynamic systems (Funahashi and Nakamura, 1993). Recurrent networks have been applied to ASR in (Watrous and Shastri, 1987; Kangas et al., 1990; Robinson and Fallside, 1991).

HMMs have been very successful in ASR. However, the HMM methodology gives only a framework and it does not address the practical modeling of the observation pdfs of the states. In this respect ANNs have been very useful. Besides estimating pdfs without using any traditional parametric models, ANNs have also been used for directly estimating the posterior probabilities of the states (Bourlard and Wellekins, 1990; Renals et al., 1994). Several HMM/ANN combinations have been experimented like HMM/MLP (Bourlard and Wellekins, 1990; Franzini et al., 1990), HMM/RBF (Renals et al., 1991; Singer and Lippmann, 1992), HMM/TDNN (Devillers and Dugast, 1991; Dugast et al., 1994), HMM/SOM (Kurimo, 1997), and HMM/LVQ (Kurimo, 1997). The SOM has been used in the continuous density HMM framework for initializing the kernel mean vectors of Gaussian mixtures and LVQ has been used for their error corrective tuning.

In some earlier works the TDNN (Ma and Compernolle, 1990), SOM (Torkkola et al., 1991) and LVQ (Iwamida et al., 1990; Torkkola et al., 1991) have been used as frame labelers for HMM. In that case HMM decoding can be considered as a postprocessing method for discrete, symbolic data.

Reviews on the use of ANNs in ASR can be found in (Lippmann, 1989; Morgan and Scofield, 1991; Furui and Sondhi, 1992; Morgan and Bourlard, 1995).

4 CASE STUDIES ON SEQUENCE PROCESSING USING THE SELF-ORGANIZING MAP AND LEARNING VECTOR QUANTIZATION

In the following case studies, the SOM has been used for constructing models of sequential data. In order to gain a view of different representations, levels of abstraction, and sequences for which a SOM can be constructed, see Figure 5. First the SOM has been used as a vector quantizer of speech without taking the order of the speech frames into account. The time information of the input has been utilized by connecting the local models on the SOM according to the trajectories of the input. The SOM was first trained using a predefined regular grid and then using data-driven node connections. The network then represented the input sequences by means of the local models and their temporal connections. The sequential nature of the input has been taken into account also when temporal segment models have been trained on the SOM. Dynamic time warping and the Viterbi algorithm were then used for determining the best-matching units. Finally, the clustering of higher-level abstractions has been demonstrated by constructing the SOM for symbol strings.

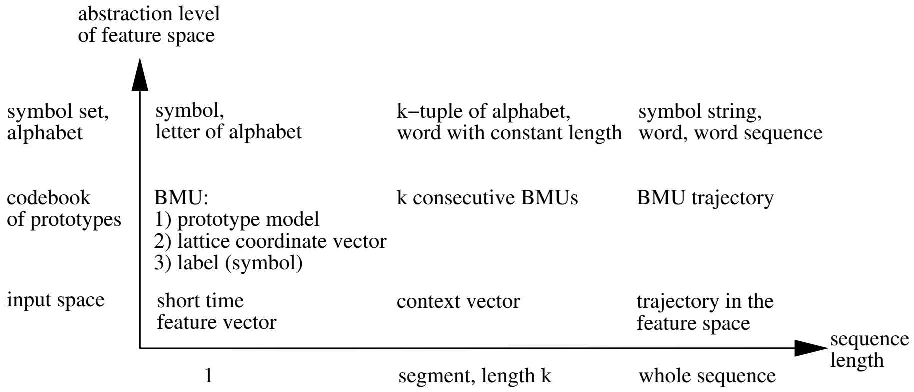


Figure 5: SOMs can be constructed for different abstraction levels of the input. This diagram illustrates what kind of models can be associated with the SOM nodes.

4.1 SOM as a vector quantizer

In Publication 1, the SOM was used as a codebook for quantizing the stream of continuous-valued feature vectors into the sequence of codebook symbols. Each speech frame was encoded by the index of its best-matching unit. The recognition of the speech signal sequence was then based on processing the corresponding map unit sequence. This work was motivated by the experiments with the existing classification method, the Redundant Hash Addressing (RHA) method (Kohonen and Reuhkala, 1978). RHA has been used mainly for error-correction of symbol strings. The central idea in the RHA is to extract multiple features from the same input item. The comparison of the input item against the reference items is based on the common features shared by the two items. In case of character strings, segments of N consecutive letters (N-grams) have been used. The RHA system consists of the N-gram table and the dictionary, see Figure 6. Multiple features (N-grams) are extracted from the input string and each extracted N-gram associates the input string with the dictionary items. The closest dictionary item can be defined according to the feature distance (Kohonen and Reuhkala, 1978; Kohonen, 1995):

$$FD(A, B) = \max(n_A, n_B) - n_d \ , \tag{14}$$

where n_A and n_B are the numbers of N-grams in the sequences A and B, respectively, and n_d is the number of matching N-grams. A match can be defined to occur if the positions of the N-grams shared by two sequences differ no more than d symbol positions.

The RHA method is fast, because it does not compare the input items directly against all reference items in the dictionary. Comparison can be restricted to only those items which share common features. The redundancy of the extracted N-grams enables the error tolerance in retrieval. It suffices that only some of the N-grams in the input sequence remain free from defects.

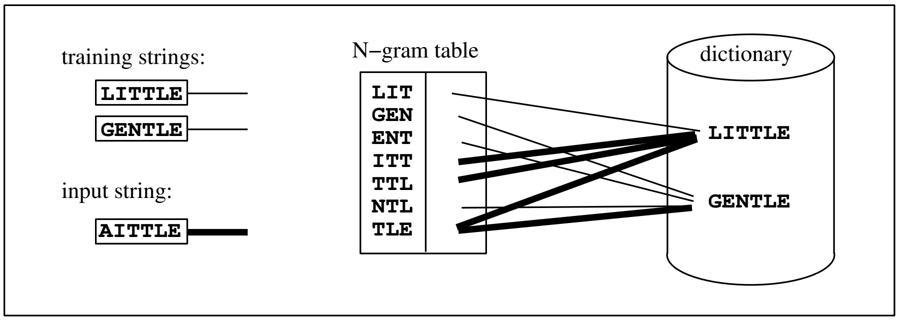


Figure 6: The RHA principle applied to character strings. The N-gram table is constructed by extracting the N-grams from training strings. Here trigrams are used ($N = 3$). From each item in the N-gram table there are associations (pointers) to the dictionary items. Associations activated by the erroneous input string 'AITTLE' are depicted by thick lines. In the SOM/RHA hybrid, the N-grams consist of the indices of the SOM nodes.

The speech material used in the experiments consisted of 1760 utterances collected from 20 speakers (5 female speakers and 15 male speakers). The vocabulary was 22 Finnish command words. 10-dimensional cepstrum vectors were computed from the spoken words at 8 ms time intervals using 16 ms speech frames. The SOM was trained and then the N-gram table was constructed by using the training set. RHA-based word recognition was then conducted using the test set.

Besides using only the index of the best-matching unit and the RHA method, also other transformations were experimented. Coordinate vectors and symbol probability vectors of the BMUs were used as representations of the input. Dynamic time warping was used for matching these representations against reference sequences. The benefit of the RHA method was the superior speed compared with the DP-based methods, see Table 1. Although the recognition accuracy of the RHA method was not as high as the accuracy using 10-dimensional cepstrum vectors and DTW, the recognition time was an order of a magnitude smaller. The BMU index sequences were matched against reference templates also by using the Levenshtein distance. The result showed that the recognition accuracy using BMU index sequences can be as good as using cepstrum vectors and DTW if multiple reference templates are used for each class, however, this increases the recognition time considerably. A remarkable property of the RHA method is that adding new sequence templates to the recognition system does not slow the performance distinctly.

The recognition system using the SOM with the RHA method is based on collecting the histograms of short BMU sequences from the sliding time windows of the input sequence and

Table 1: Multi-speaker speech recognition experiment with 1760 utterances from the vocabulary of 22 Finnish command words. Time is the average recognition time for one sequence using a 5×5 SOM.

feature, recognition method	error, per cent		time/ms
one reference sequence per class:			
1. 10-dim unquantized, DTW	3.4		78
	5×5 SOM	10×5 SOM	
2. 10-dim prototype vector of BMU, DTW	5.3	4.5	76
3. 2-dim lattice coordinate vector of BMU, DTW	6.6	5.0	55
4. 19-dim phoneme probability vector of BMU, DTW	4.8	3.2	98
multiple reference sequences (60) per class:			
5. BMU index, Levenshtein	3.4	5.1	1060
6. BMU index, RHA N=2, d=150ms	6.6	7.7	5

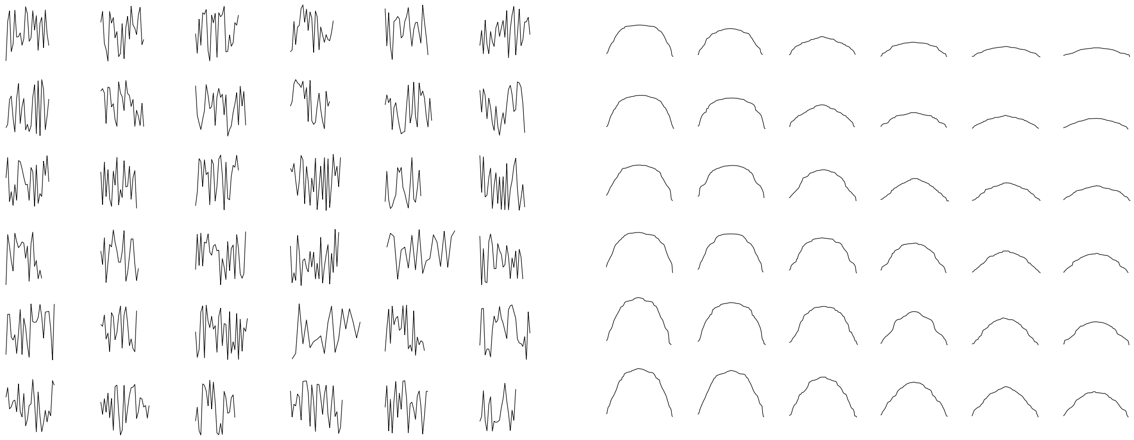
comparing these with the corresponding histograms of the training sequences. No segmentation of the input signal into phonemes or other predefined classes is required. The SOM is used in a purely unsupervised mode.

Classification of feature sequences extracted from pieces of music was also experimented using the same principle. Mel-scaled spectrum vectors were computed from pieces of Bach's piano music "Well-Tempered Clavier". A 10-by-10-unit SOM was trained and the RHA table was constructed using 24 training sequences with durations of 10 seconds each. The test material consisted of 24 subsequences of the training sequences with a duration of one second. The beginnings of the subsequences were randomly chosen. As a result, for N=1, there were 2 erroneous retrievals, 4 completely correct retrievals, and 18 retrievals where the correct music piece shared a tie with an incorrect retrieval. For N=2, all retrievals were completely correct. Also for the values N=3, N=4, and N=5, all retrievals were completely correct.

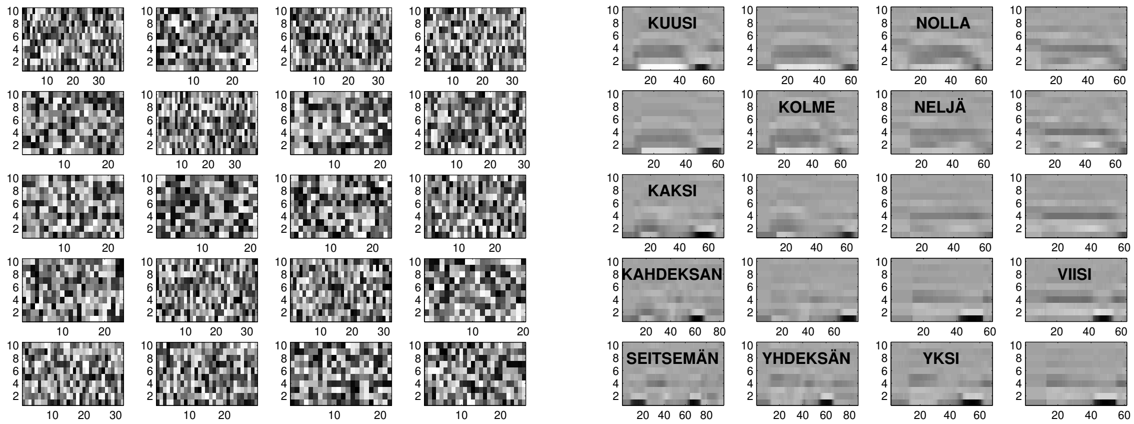
In the SOM, the number of nodes can be considered as the amount of available resources. For exploratory data analysis and pure visualization purposes, the choice of this number is not so crucial. However, when the indices of the SOM units are used as symbols in pattern recognition, the number of nodes is an important parameter. The Voronoi cells of single prototype vectors are convex polyhedra. In order to construct regions with arbitrary shapes, several prototype vectors should be grouped together. In supervised learning, certain amount of resources or prototype vectors can be allocated a priori to each class, which determines the representation accuracy of the class decision borders. In unsupervised learning, however, the number of classes is not known or there are no such things as classes. If higher-level representations and abstractions are desired by means of unsupervised learning methods, there should be some kind of a mechanism for connecting similar events and patterns together. A cluster of prototype vectors could then behave as an entity instead of single prototype vectors.

4.2 SOM and feature sequence models

In competitive learning, the models may compete against each other on the basis of how well they represent the input. The models on the SOM grid can represent the dynamic nature of the input better if they are extended from single feature vectors into feature vector sequences. A straightforward way to implement this is to concatenate a fixed number of successive feature vectors into a new feature vector. However, a more flexible approach is if the number of the concatenated feature vectors is not defined a priori and the lengths of the prototype sequences may vary during the learning process. Feature vector sequences with different lengths can be compared by using the dynamic time warping algorithm, see e.g. (Sakoe and Chiba, 1978; Sankoff and Kruskal, 1983). Prototype sequences can be constructed on the SOM grid using unsupervised learning and LVQ can be used for fine tuning them. In Publication 2 the SOM and LVQ algorithms were constructed for variable-length feature sequences.



(a) One-dimensional input sequences



(b) Ten-dimensional input sequences

Figure 7: Prototype sequences on the SOM for variable-length input sequences. The inital map is shown on the left and the map after training is shown on the right.

Examples of SOM-based prototype sequences are shown in Figure 7. In the first example, see Figure 7(a), feature vectors are one-dimensional. Prototype sequences of the SOM were initialized by white noise. Input data consisted of arcs having different shapes, amplitudes and lengths. Arithmetic averaging of sequences (Sankoff and Kruskal, 1983) was used and the prototype sequences were adapted after each input sample sequence. The shapes of the input varied between triangular, Gaussian and almost box-like forms, but because DTW is insensitive to durational differences, the slopes of the arcs were ignored and amplitudes of the sequences determined the ordering. The Gaussian shape was the average form of the input shapes.

In the second example, see Figure 7(b), feature sequences were computed from natural speech. 10-dimensional cepstrum vectors were used as feature vectors and input data consisted

Table 2: Speaker-independent word-recognition experiment with 1760 utterances from the vocabulary of 22 Finnish command words. The centermost sequence (median) was defined according to Equation (4).

reference templates	error, per cent
one randomly picked sequence from each class	18.5
one centermost sequence from each class	3.1
one LVQ fine tuned sequence for each class	1.5

of one hundred utterances of numbers from zero to nine in Finnish. The SOM was initialized by 10-dimensional white noise vector sequences. After training, the map was labeled according to majority voting. Phonetically similar words were located near each other on the SOM.

Speaker-independent speech recognition was then experimented using one reference template for each of the 22 Finnish command words in the vocabulary (this was the same data set as used in Publication 1). Tests were repeated 20 times, each time having a different speaker in the test set and the remaining 19 speakers in the training set. There were altogether 1760 sequences in the test sets. 10-dimensional cepstrum vectors were used as features. Basic DTW was used with no slope constraints. The slope constraint introduced in (Sakoe and Chiba, 1978) would have limited the slope of the warping path between 0.5 and 2 and the length of the longer sequence could not have exceeded two times the length of the shorter sequence if the start and end points of the sequences are fixed in the matching. However, for some words in the current data set, the duration of the longest utterance was three times the duration of the shortest utterance. The results of the speech recognition experiments in Table 2 show the superiority of the reference sequences based on LVQ.

Besides basing the matching of prototypes and input sequences on the use of time-normalized distances, also probability measures can be used. This was experimented in Publication 3 where HMMs were used as the models associated with the map nodes and the Viterbi algorithm was used in the BMU search. An unsupervised segmentation method was developed where the HMMs associated with the nodes of the SOM were segment models of speech. The emission probability densities of the states were modeled by Gaussian mixtures. The training was based on repeating the following two steps for each input sequence in the training set:

1. *BMU-search.* Segment the input sequence to the map units by using the Viterbi algorithm.

2. *Model adaptation.* Use each data segment for updating the model of its BMU and the models of the neighboring nodes.

The BMU-search, which means the same as the segmentation in case of feature vector sequences, is illustrated in Figure 8. Due to the Viterbi algorithm, the individual feature vectors in the BMU-search are not matched against map nodes as separate items but as a part of an entire feature sequence. After unsupervised learning, the models on the SOM grid represented whole phonemes or finer details like transition parts. An example of the resulting segmentation is shown in Figure 9.

In order to evaluate the quality of the segment models and investigate whether they are relevant and consistent for the speech recognition task, a word recognition experiment was conducted.

The reference word models were obtained by computing the best state sequence for each word utterance in the training set by using the Viterbi algorithm. These state sequences

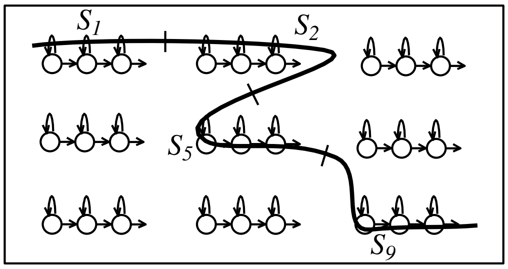
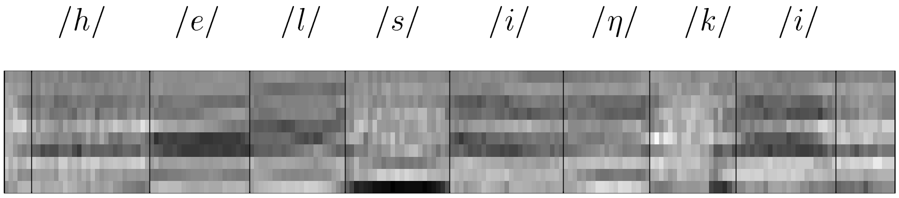
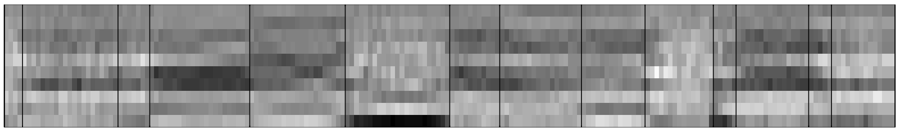


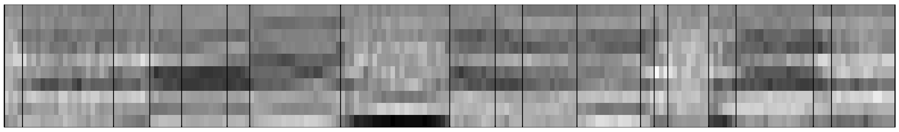
Figure 8: Competitive-learning segment models on the SOM. Each map node is associated with an HMM having one or more states (a three-state left-to-right model in this example). The thick line represents the segmentation of an input sequence. The data segments are denoted by S_i, where i is a BMU index. The models of BMUs and neighboring units are then updated by the corresponding data segments.



(a) Phoneme borders



(b) Segment borders using the SOM with 24 units



(c) Segment borders using the SOM with 216 units

Figure 9: Segmentation of the word utterance 'HELSINKI' using SOM-based segment models. 10-dimensional feature vectors are represented as column vectors on the horizontal time-axis. Segment borders according to linguistic speech units, phonemes, are shown on the top for comparison.

constituted the basis of the word models and they were stored in the pronunciation dictionary. When forming the word models, the durations of the states were compressed so that the same state was not allowed to follow itself more than a limited number of times. This parameter is denoted by 'max segment length in word model' in Table 3. The word recognition was then performed using the test data. Comparing the results of the three maps, the best time resolution of the largest SOM (18×12 competing models) gave the best word recognition result.

For comparison, the word recognition was experimented also using linguistic speech units. Three-state left-to-right HMMs were trained for 20 phonemes. One additional model was trained for silence. The reference word models were constructed by concatenating the phoneme models according to the correct pronunciation of the word. The best recognition result was

Table 3: Speaker-dependent word recognition using segment models of the SOM. Each segment model was a three-state HMM. The vocabulary consisted of 340 Finnish words. Number of correct words per cent.

number of HMMs	max segment length in word model 1	2	3	4	5
6×4	97.7	97.4	96.6	97.7	97.4
9×6	97.1	97.7	98.3	98.6	98.0
18×12	98.0	99.1	99.7	100.0	100.0

then 99.7 per cent. This is very close to the results obtained by using segment models derived in an unsupervised manner.

In the previous works, the neighborhood on the SOM grid was utilized for initializing and smoothing the models rather than representing the temporal relations of the models. In Publication 4 the node connections were created for representing the temporal trajectories in the input feature space. The SOM was first trained with the regular map grid for initializing the models after which the node connections on the regular map grid were removed and the new node connections were created according to the time information of the input.

The connection was created between those two nodes which were the best-matching units for two successive input vectors in time. The strength of the connection between nodes i and j was defined as

$$a'_{ij} = \frac{\sum_{t=2}^{T} \delta(i - c(\mathbf{x}_{t-1}))\delta(j - c(\mathbf{x}_t))}{\sum_{t=2}^{T} \sum_{j'} \delta(i - c(\mathbf{x}_{t-1}))\delta(j' - c(\mathbf{x}_t))}, \tag{15}$$

where

$$\delta(l) = \begin{cases} 1 & \text{if } l = 0, \\ 0 & \text{otherwise} \end{cases}, \tag{16}$$

and $c(\mathbf{x}_{t-1})$ and $c(\mathbf{x}_t)$ are the best-matching units for successive input items $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, respectively.

This method was applied in Publication 5 for speeding up the performance of a speech recognizer based on mixture density HMMs. When modeling HMM states by means of a large number of Gaussians, usually only a few of them dominate when computing the emission probabilities of the states for a given input vector. Significant speedup can be gained if the dominating Gaussians are found without investigating the whole mixture. In this work the time information of the input was utilized when reducing the search space for finding the dominating Gaussians. Each map unit was provided with the list of nodes sorted according to Equation (15). This list was utilized in the reduced search of K-best Gaussians in the recognition phase. In the reduced search only those kernels are investigated which have connections from the previous best-matching units in each state.

The speech data for phoneme recognition experiments was collected from six male speakers and four female speakers. Each speaker had uttered four times the vocabulary of 350 Finnish words. The speaker-dependent recognition system was based on phoneme-wise HMMs. Each phoneme was modeled by a three-state left-to-right HMM. A separate 12-by-8-unit SOM was trained for each phoneme and a spherical Gaussian kernel was attached to each SOM node as in

(Kurimo, 1997). A fixed kernel width was used as a smoothing parameter of the final pdf. After initializing the state codebooks by the SOMs, the HMMs were trained by the segmental K-means algorithm. Only state transition probabilities inside phoneme models were re-estimated. The probabilities for phoneme transitions had been estimated from the larger Finnish text corpus. The phoneme recognition error was then computed using the fourth speech set. In the baseline system the full codebook search was performed at every speech frame resulting in the phoneme error rate of 5.7%. The influence of the time interval between the full codebook searches on the recognition result was then tested. The recognition results and the corresponding amount of distance computations are shown in Figure 10. 75% of the distance computations could be reduced when keeping the error rate below that of the baseline system. The reason for the good performance was that the node connections modeled well the temporal trajectories of the speech which were followed during the recognition.

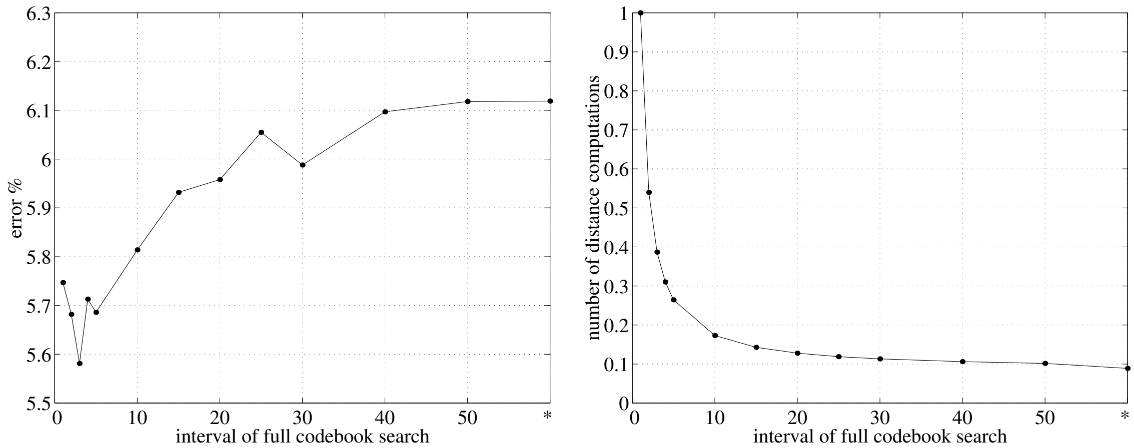


Figure 10: Speech recognition using temporally connected kernels in mixture density HMMs. Average phoneme errors are shown on the left and the corresponding number of distance computations between feature vectors and kernels are shown on the right. The rightmost error rate (*) is for the case where the full codebook search was performed only once in the beginning of each word utterance. The average length of a word utterance was 103 frames. The number of distance computations is scaled by that of the baseline system where the full search was performed at every speech frame.

4.3 SOM of symbol strings

An extension of the SOM, proposed by Kohonen in 1996, allows the construction of the SOM also for nonvectorial data sets (Kohonen, 1996). The first demonstration and application of this new method was the construction of the pronunciation dictionary for a given speech recognizer. Publication 6 is based on these results.

1760 word utterances from the vocabulary of 22 Finnish command words were collected from 20 speakers (the same data as used in Publication 1) and converted into phoneme sequences by using an HMM-based phoneme recognizer. The speech recognizer had no grammatical restrictions to its output. The SOM of symbol strings was then trained using these phoneme sequences as input data. The algorithm was based on repeating the two steps:

1. For each map unit i, collect a list of those sample strings to whom the reference string of unit i is the nearest reference string.

2. For each map unit i, take for the new reference string the mean or median over the union of the lists that belong to the topological neighborhood set of unit i.

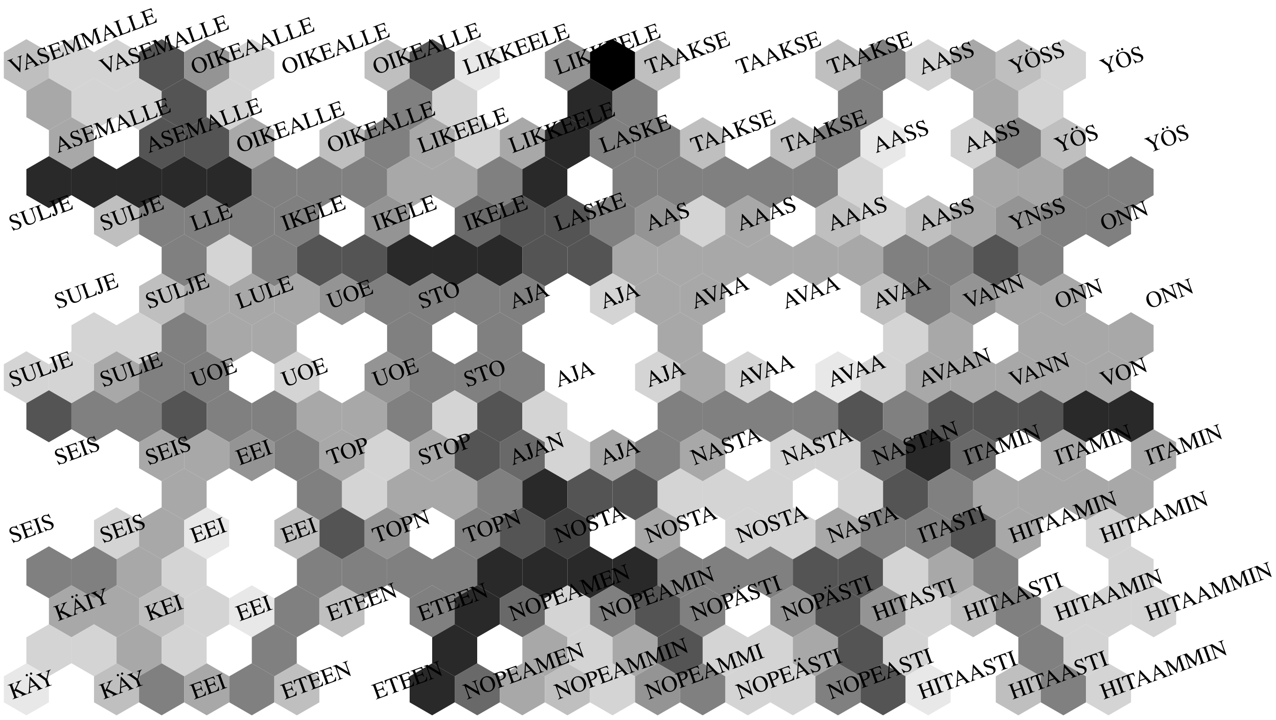


Figure 11: A 13-by-9-unit string-mean SOM. The shades of gray represent distances between neighboring reference vectors; dark means large distance, white small distance, respectively.

The DP-based weighted Levenshtein distance was used for computing the distance between strings. An example of the resulting SOM is shown in Figure 11. The sequence mean was defined according to Equation (4) but using squared pairwise distances. According to (Kohonen, 1996), if m is restricted to be an exact replica of an item in the training set, it is called a *set median/mean*. The median/mean which minimizes the right hand side of Equation (4) can then be found by systematically varying the symbol positions of the set median/mean using insertions, deletions, and substitutions of symbols (Kohonen, 1985).

Multi-speaker word recognition experiments were carried out by using 9-by-9 unit SOMs. Phoneme strings were divided into four sets, each containing 440 strings. The recognition experiments were repeated four times, each time using different combination of training and test sets. The SOM was trained by using the phoneme sequences of the training set and the resulting string averages at the nodes of the SOM were then stored in the pronunciation dictionary. The word class of each prototype sequence was determined by majority voting. The final word recognition was carried out by matching the phoneme sequences of the test set against the reference sequences of the pronunciation dictionary. The recognition accuracy was further improved by fine tuning the prototype sequences with the special LVQ algorithm for nonvectorial items (Kohonen, 1996). The results are shown in Table 4. For comparison, when the conventional (linguistic) phonemic transcriptions were used as reference strings, the error percentage was considerably higher: 5.9 per cent for the test set.

The SOM of symbol strings was also applied to the clustering of all currently known protein sequences, see Publication 7. The SWISS-PROT database release 37 (Bairoch and Apweiler, 1999) consists of 77,977 sequences with 28,268,293 amino acid residues. The FASTA method (Pearson and Lipman, 1988) was used for computing the pairwise sequence similarities. A collection of true representative sequences that approximated the contents of the database in a compact way was then found by means of the SOM. The structure of the clustering was visualized by indicating the degree of the similarity of the adjacent model sequences by shades of gray, see Figure 12.

Table 4: Word error rates in speech recognition using SOM- and LVQ-based pronunciation dictionaries.

Mean strings

	training set	test set
SOM only, mean	4.1	4.4
SOM only, set mean	4.0	4.0
SOM + set LVQ1	3.5	3.8
SOM + LVQ1	2.6	2.7

Median strings

	training set	test set
SOM only, median	4.3	4.5
SOM only, set median	3.7	3.7
SOM + set LVQ1	3.4	3.5
SOM + LVQ1	3.2	3.3

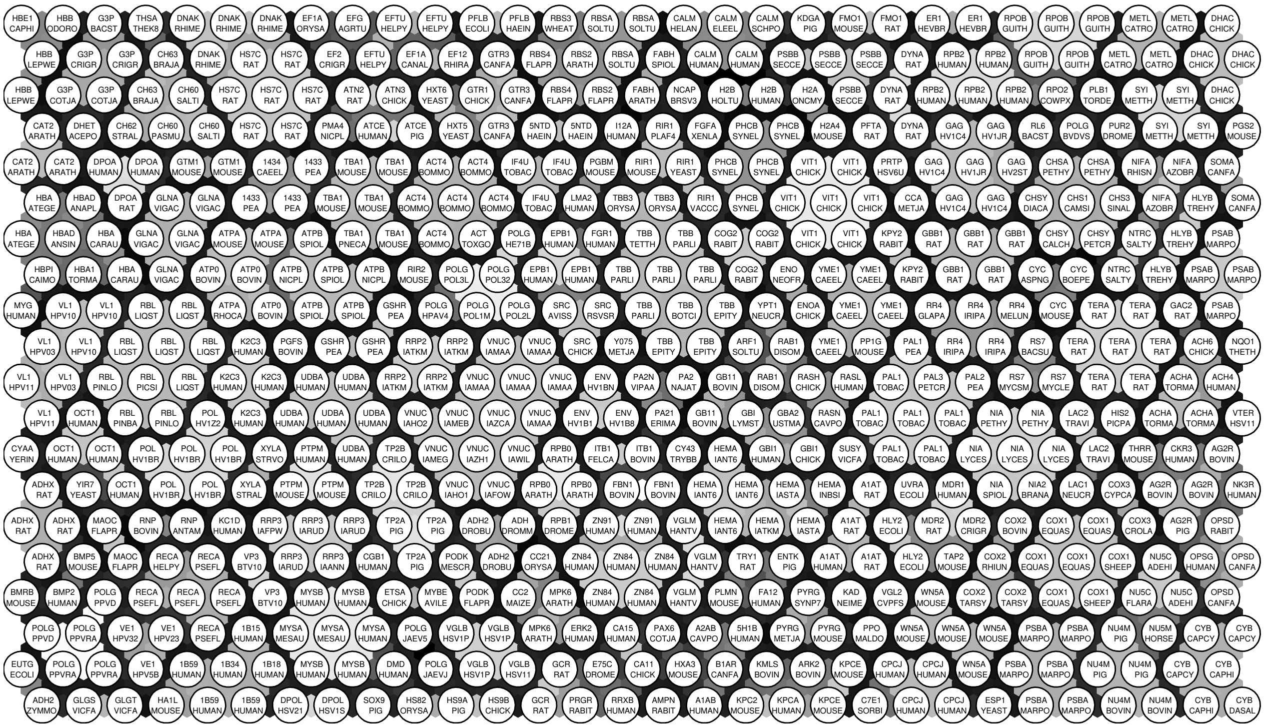


Figure 12: A 30-by-20-unit SOM for protein sequences. The map was constructed using all the 77,977 protein sequences of the SWISS-PROT database. Each node contains a prototype sequence and a list of data sequences. The labels on the map nodes are the SWISS-PROT identifiers of the prototype sequences. The upper label in each map node is the mnemonic of the protein name and the lower label is the mnemonic of the species name. The similarities of the neighboring prototype sequences are indicated by shades of gray. The light shades indicate a high degree of similarity, and the dark shades a low degree of similarity, respectively. Light areas on the map reveal large clusters of similar sequences.

5 CONCLUSION

In this thesis, the SOM has been used as a framework when constructing models for sequence processing. Traditionally, fixed-dimensional feature vectors have been used as models associated with the nodes of the SOM. In this work also other kinds of models were experimented like segments of the sequences, whole feature vector sequences, and symbol sequences. The goal was to modify the basic SOM to better take into account the sequential nature of the input.

SOM-based models were experimented in the thesis work mainly in speech recognition, but the developed methods are not restricted to this application only. For example, the SOM of symbol strings was first applied to the construction of a pronunciation dictionary according to the outputs of a phonemic speech recognizer. The same algorithm was then applied to the organization of all currently known protein sequences.

A great advantage of the SOM is that the basic form of the algorithm is very simple and straightforward to implement. It is therefore easy to apply the SOM to various tasks. By means of an unsupervised learning process the SOM gives an organized representation of input using a limited number of local models. Various kinds of feature extraction methods and (dis)similarity measures for comparing the models and data items can be used. The resulting SOM reflects the properties of these choices.

The SOM can be used as a data mining and visualization tool for any data set, for which a similarity or distance measure between its elements can be defined. Besides visualization, the SOM can be used for initializing the models for pattern recognition tasks.

REFERENCES

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 49–52, Tokyo, Japan.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1988). A new algorithm for the estimation of hidden Markov model parameters. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 493–496, New York.

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190.

Bairoch, A. and Apweiler, R. (1999). The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Research*, 27:49–54.

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

Bauer, H.-U. and Villmann, T. (1997). Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8(2):218–226.

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov chains. *Inequalities*, 3:1–8.

Baum, L. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, New Jersey.

Bezdek, J. and Pal, N. (1995). An index of topological preservation for feature extraction. *Pattern Recognition*, 28(3):381–391.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

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

Bourlard, H. and Wellekins, C. J. (1990). Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Bradburn, D. (1989). Reducing transmission error effects using a self-organizing network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume II, pages 531–537, Washington, D.C.

Chappel, G. J. and Taylor, J. G. (1993). The temporal Kohonen map. *Neural Networks*, 6(3):441–445.

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

Cherkassky, V. and Mulier, F. (1998). *Learning from Data: Concepts, Theory, and Models.* John Wiley & Sons, NY.

Chou, W., Juang, B. J., and Lee, C. H. (1992). Segmental GPD training of HMM based speech recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume I, pages 473–476, San Francisco, California.

Cottrell, M., Fort, J. C., and Pagès, G. (1997). Theoretical aspects of the SOM algorithm. In *Proceedings of the Workshop on Self-Organizing Maps (WSOM)*, pages 246–267, Espoo, Finland.

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Deshmukh, N., Ganapathiraju, A., and Picone, J. (1999). Hierarchical search for large-vocabulary conversational speech recognition. *IEEE Signal Processing Magazine*, 16(5):84–107.

Devillers, L. and Dugast, C. (1991). Comparison of continuous mixture densities and TDNN in a Viterbi-framework: Experiment on speaker dependent DARPA RM1. In *Proceedings of the 2nd European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 2, pages 991–994, Genova, Italy.

Dugast, C., Devillers, L., and Aubert, X. (1994). Combining TDNN and HMM in a hybrid system for improved continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):217–223.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Erwin, E., Obermayer, K., and Schulten, K. (1992a). Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55.

Erwin, E., Obermayer, K., and Schulten, K. (1992b). Self-organizing maps: stationary states, metastability and convergence rate. *Biological Cybernetics*, 67(1):35–45.

Flanagan, J. A. (1999). Self-organization in the SOM with a decreasing neighborhood function of any width. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 156–161, Edinburgh, UK.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Franzini, M. A., Lee, K. F., and Waibel, A. (1990). Connectionist Viterbi training: A new hybrid method for continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 425–428, Alburquerque, New Mexico.

Fritzke, B. (1991). Let it grow - self-organizing feature maps with problem dependent cell structure. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume I, pages 397–402. Elsevier, Amsterdam.

Fritzke, B. (1994). Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460.

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

Fritzke, B. (1995b). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 625–632. The MIT Press, Cambridge, Massachusetts.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 3(4):183–191.

Funahashi, K. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806.

Furui, S. and Sondhi, M. M. (1992). *Advances in Speech Signal Processing*. Marcel Dekker, Inc., New York.

Gales, M. and Young, S. (1993). Segmental hidden Markov models. In *Proceedings of the 3rd European Conference on Speech Communication and Technology (EUROSPEECH)*, volume 3, pages 1579–1582, Berlin, Germany.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298.

Ghitza, O. (1994). Auditory models and human performance in tasks related to speech coding and speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):115–132.

Gorin, A., Levinson, S., Gertner, A., and Goldman, E. (1991). Adaptive acquistion of language. *Computer Speech and Language*, 5(2):101–132.

Gray, R. (1984). Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan, New York.

Herrmann, M., Bauer, H.-U., and Villmann, T. (1997). Measuring topology preservation in maps of real-world data. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pages 205–210, Bruges, Belgium.

Heskes, T. M. and Kappen, B. (1993). Error potentials for self-organization. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume III, pages 1219–1223, San Francisco, California.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

Holmes, W. and Russell, M. (1995). Experimental evaluation of segmental HMMs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 536–539, Detroit.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayered feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Iwamida, H., Katagiri, S., McDermott, E., and Tohkura, Y. (1990). A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume I, pages 489–492, Alburquerque, New Mexico.

Jankowski Jr., C., Vo, H., and Lippman, R. (1995). A comparison of signal processing front ends for automatic word recognition. *IEEE Transactions on Speech and Signal Processing*, 3(4):286–292.

Jelinek, F. (1969). A fast sequential decoding algorithm using a stack. *IBM Journal of Research Development*, 13:675–685.

Jelinek, F. (1997). *Statistical methods for speech recognition.* The MIT Press.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546. Erlbaum, Hillsdale, New Jersey.

Juang, B.-H. and Rabiner, L. R. (1991). Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272.

Kangas, J. (1990). Time-delayed self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume II, pages 331–336, San Diego, California.

Kangas, J. (1994). *On the Analysis of Pattern Sequences by Self-Organizing Maps.* PhD thesis, Helsinki University of Technology, Finland.

Kangas, J. (1995). Using self-organizing map in error tolerant transmission of vector quantized images. In *Proceedings of the World Congress on Neural Networks (WCNN)*, pages 517–522, Washington D.C.

Kangas, J., Kohonen, T., and Laaksonen, J. (1990). Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1(1):93–99.

Kaski, S. (1997). *Data Exploration Using Self-Organizing Maps.* PhD thesis, Helsinki University of Technology, Finland.

Kaski, S. and Lagus, K. (1996). Comparing self-organizing maps. In von der Malsburg, C., Vorbrüggen, J. C., von Seelen, W., and Sendhoff, B., editors, *Artificial Neural Networks - ICANN 96*, volume 1112 of *Lecture Notes in Computer Science*, pages 809–814, Berlin. Springer.

Katagiri, S. and Lee, C.-H. (1993). A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization. *IEEE Transactions on Speech and Audio Processing*, 1(4):421–430.

Katagiri, S., Lee, C.-H., and Juang, B.-H. (1991). New discriminative training algorithms based on the generalized probabilistic descent method. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 298–308, Princeton, New Jersey.

Kohonen, T. (1981). Automatic formation of topological maps of patterns in a self-organizing system. In *Proceedings of The Second Scandinavian Conference on Image Analysis*, pages 214–220, Helsinki, Finland.

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

Kohonen, T. (1985). Median strings. *Pattern Recognition Letters*, 3:309–313.

Kohonen, T. (1986). Learning vector quantization for pattern recognition. Technical Report A601, Helsinki University of Techonology, Laboratory of Computer and Information Science, Finland.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Kohonen, T. (1991). Self-organizing maps: optimization approaches. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume II, pages 981–990. Elsevier, Amsterdam.

Kohonen, T. (1992). New developments of learning vector quantization and the self-organizing map. In *Symposium on Neural Networks; Alliances and Perspectives in Senri (SYNAPSE)*, Osaka, Japan.

Kohonen, T. (1993). Generalizations of the self-organizing map. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume I, pages 457–462, Nagoya, Japan.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin (2nd ed. 1997).

Kohonen, T. (1996). Self-organizing maps of symbol strings. Technical Report A42, Helsinki University of Technology, Laboratory of Computer and Information Science, Finland.

Kohonen, T. (1999). Comparison of SOM point densities based on different criteria. *Neural Computation*, 11(8):2081–2095.

Kohonen, T. (2000). Perspectives on the research of artificial neural networks. *International Journal of Computer Research*.

Kohonen, T., Kaski, S., and Lappalainen, H. (1997). Self-organized formation of various invariant-feature filters in the adaptice subspace SOM. *Neural Computation*, 9(6):1321–1344.

Kohonen, T., Oja, E., Simula, O., Visa, A., and Kangas, J. (1996). Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384.

Kohonen, T. and Reuhkala, E. (1978). A very fast associative method for the recognition and correction of misspelt words, based on redundant hash addressing. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, pages 807–809, Kyoto, Japan.

Kohonen, T., Riittinen, H., Reuhkala, E., and Haltsonen, S. (1984). On-line recognition of spoken words from a large vocabulary. *INFORMATION SCIENCES*, 33:3–30.

Koikkalainen, P. (1994). Progress with the tree-structured self-organizing map. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI)*, pages 211–215, Amsterdam, The Netherlands.

Koikkalainen, P. and Oja, E. (1990). Self-organizing hierarchical feature maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume II, pages 279–284, San Diego, California.

Koskela, T., Varsta, M., Heikkonen, J., and Kaski, K. (1998). Time series prediction using recurrent SOM with local linear models. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 2(1):60–68.

Kraaijveld, M., Mao, J., and Jain, A. (1995). A non-linear projection method based on Kohonen's topology preserving maps. *IEEE Transactions on Neural Networks*, 6(3):548–559.

Kruskal, J. and Wish, M. (1978). *Multidimensional Scaling.* Number 07-11 in Sage University Paper Series on Quantitative Applications in the Social Sciences. Sage Publications, Newbury Park, CA.

Kurimo, M. (1997). *Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models.* PhD thesis, Helsinki University of Technology, Finland.

Lampinen, J. and Oja, E. (1989). Self-organizing maps for spatial and temporal AR models. In *Proceedings of the 6th Scandinavian Conference on Image Analysis*, pages 120–127, Oulu, Finland.

Lee, K.-F. (1989). *Automatic Speech Recognition - The Development of the SPHINX System.* Kluwer Academic Publishers, Boston.

Leung, H. C., Hetherington, I. L., and Zue, V. (1992). Speech recognition using stochastic segment neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 613–616, San Francisco, California.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10(8):707–710.

Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95.

Lippmann, R. (1989). Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38.

Lippmann, R. P., Martin, E. A., and Paul, D. B. (1987). Neural-net classifiers useful for speech recognition. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume IV, pages 417–425, San Diego, California.

Lowerre, B. T. (1976). *The Harpy Speech Recognition System.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Ma, W. and Compernolle, D. V. (1990). TDNN labeling for a HMM recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 421–424, Alburquerque, New Mexico.

Makhoul, J., Roucos, S., and Gish, H. (1985). Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):1551–1588.

Mäntysalo, J., Torkkola, K., and Kohonen, T. (1994). Mapping context dependent acoustic information into context independent form by LVQ. *Speech Communication*, 14(2):119–130.

Martinetz, T., Berkovich, S., and Schulten, K. (1993). Neural-gas network for vector quantization and its application to time series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569.

Martinetz, T. and Schulten, K. (1991). A neural-gas network learns topologies. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume I, pages 397–402. Elsevier, Amsterdam.

Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7(3):507–522.

McDermott, E. and Katagiri, S. (1989). Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 81–84, Glasgow, Scotland.

Minamimoto, K., Ikeda, K., and Nakayama, K. (1995). Topology analysis of data space using self-organizing feature map. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume 2, pages 789–794, Pearth, Australia.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60.

Morgan, D. P. and Scofield, C. L. (1991). *Neural Networks and Speech Processing*. Kluwer Academic Publishers, Boston Dodrecht London.

Morgan, N. and Bourlard, H. (1995). Neural networks for statistical recognition of continuous speech. *Proceedings of the IEEE*, 83(5):742–770.

Mulier, F. and Cherkassky, V. (1994). Learning rate schedules for self-organizing maps. In *12th IAPR International Conference on Pattern Recognition*, volume II, pages 224–228, Jerusalem, Israel.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino-acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453.

Ney, H. and Ortmanns, S. (1999). Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83.

Niles, L. T. and Silverman, H. F. (1990). Combining hidden Markov models and neural network classifiers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 417–420, Alburquerque, New Mexico.

Nilsson, N. J. (1971). *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, New York.

Ostendorf, M., Diagalakis, V. V., and Kimball, O. A. (1996). From HMM's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378.

Pallett, D., Fiscus, J., Garofolo, J., Martin, A., and Przybocki, M. (1999). 1998 broadcast news benchmark test results. In *Proceedings of the DARPA Broadcast News Workshop*, Herndon, Virginia.

Pearson, W. and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Science USA*, 85:2444–2448.

Picone, J. (1993). Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–1247.

Rabiner, L., Wilpon, J., and Juang, B. (1986). A segmental K-means training procedure for connected word recognition. *AT&T Technical Journal*, 64:21–40.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Renals, S., McKelvie, D., and McInnes, F. (1991). A comparative study of continuous speech recognition using neural networks and hidden Markov models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 369–372, Toronto, Canada.

Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. (1994). Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174.

Ritter, H. and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 61:241–254.

Ritter, H. and Schulten, K. (1988a). Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. *Biological Cybernetics*, 60(1):59–71.

Ritter, H. and Schulten, K. (1988b). Kohonen's self-organizing maps: exploring their computational capabilities. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume I, pages 109–116, San Diego, California.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Robinson, T. and Fallside, F. (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5(3):259–274.

Russell, M. (1993). A segmental HMM for speech pattern modelling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume II, pages 499–502, Minneapolis.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Sakoe, H., Isotani, R., Yoshida, K., Iso, K., and Watanabe, T. (1989). Speaker independent word recognition using dynamic programming neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 29–32, Glasgow, Scotland.

Sammon Jr., J. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409.

Sankoff, D. and Kruskal, J. B. (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley.

Singer, E. and Lippmann, R. P. (1992). A speech recognizer using radial basis function neural networks in an HMM framework. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 629–632, San Francisco, California.

Tanaka, E. and Kasai, T. (1976). Syncronization and substitution error-correcting codes for the Levenshtein metric. *IEEE Transactions on Information Theory*, 22(2):156–162.

Tang, K. S., Man, K. F., Kwong, S., and He, Q. (1996). Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13(6):22–37.

Thiran, P. and Hasler, M. (1994). Self-organization of a one-dimensional Kohonen network with quantized weights and inputs. *Neural Networks*, 7(9):1427–1439.

Torkkola, K., Kangas, J., Utela, P., Kaski, S., Kokkonen, M., Kurimo, M., and Kohonen, T. (1991). Status report of the Finnish phonetic typewriter project. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume I, pages 771–776. Elsevier, Amsterdam.

Villmann, T., Der, R., Herrmann, M., and Martinetz, T. (1997). Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266.

Villmann, T., Der, R., and Martinetz, T. (1994). A new quantitative measure of topology preservation in Kohonen's feature maps. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, volume 2, pages 645–648, Orlando, Florida.

Vinstyuk, T. K. (1968). Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Watrous, R. and Shastri, L. (1987). Learning phonetic features using connectionist networks: An experiment in speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume IV, pages 381–388, San Diego, California.

Young, S. (1996). A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, 13(5):45–57.

Zrehen, S. (1993). Analyzing Kohonen maps with geometry. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 609–612, Amsterdam, The Netherlands.

Zue, V., Glass, J., Goodine, D., Phillips, M., and Seneff, S. (1990). The SUMMIT speech recognition system: Phonological modelling and lexical access. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 49–52, Alburquerque, New Mexico.