On the Analysis of Pattern Sequences by Self-Organizing Maps

Jari Kangas

Helsinki University of Technology
Laboratory of Computer and Information Science
Rakentajanaukio 2 C, SF-02150, FINLAND

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in the Auditorium F1 of the Helsinki University of Technology on 6th of May 1994, at 12 o'clock noon.

Abstract

This thesis is organized in three parts. In the first part, the Self-Organizing Map algorithm is introduced. The discussion focuses on the analysis of the Self-Organizing Map algorithm. It is shown that the nonlinear nature of the algorithm makes it difficult to analyze the algorithm except in some trivial cases.

In the second part the Self-Organizing Map algorithm is applied to several patterns sequence analysis tasks. The first application is a voice quality analysis system. It is shown that the Self-Organizing Map algorithm can be applied to voice analysis by providing the visualization of certain deviations. The key point in the applicability of Self-Organizing Map algorithm is the topological nature of the mapping; similar voice samples are mapped to nearby locations in the map.

The second application is a speech recognition system. Through several experiments it is demonstrated that by collecting some time dependent features and using them in conjunction with the basic Self-Organizing Map algorithm one can improve the speech recognition accuracy considerably.

The applications explained in the second part of the thesis were rather straightforward works where the sequential signal itself was transformed for the analysis. In the third part of the thesis it is demonstrated that the Self-Organizing Map algorithm itself could be extended by identifying each Map unit with an arbitrary operator with capabilities for pattern sequence processing. It is shown that the operator maps are applicable for example to speech signal (waveform) categorization.

On the Analysis of Pattern Sequences by Self-Organizing Maps

Jari Kangas

Helsinki University of Technology
Laboratory of Computer and Information Science
Rakentajanaukio 2 C, SF-02150, FINLAND

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in the Auditorium F1 of the Helsinki University of Technology on 6th of May 1994, at 12 o'clock noon.

Dedication

To my wife Päivi and our little son Miika.

Acknowledgements

The work presented in this thesis has been carried out in the Laboratory of Computer and Information Science in the Faculty of Information Technology of the Helsinki University of Technology.

I wish to thank Professor Teuvo Kohonen for initially directing me to work with the Self-Organizing Map algorithm and for the discussions and the encouragement to continue the research on SOM during the ensuing years. I also wish to thank Professor Kohonen for providing the excellent facilities and support for this research.

I would like to thank all the personnel of the Laboratory of Computer and Information Science for creating an inspiring and comfortable atmosphere. The informal discussions with the research group members have always been useful.

Professor Teuvo Kohonen and Professor Erkki Oja commented in detail on the manuscript during the final revisioning which is gratefully acknowledged. Mr. Paul Schurman deserves many thanks for checking the language of the thesis.

The financial support of the Academy of Finland is acknowledged.

Espoo, April 15, 1994

Jari Kangas

Contents

Introduction **7**

Contents of the Thesis and Contributions of the Author 8

I Theoretical Motivation of the Self-Organizing Map **11**

1 Self-Organizing Maps **12**

1.1 Self-Organizing Maps 13

1.2 Topographic Maps in Brain 14

1.3 Physiological Models of the Self-Organizing Map 14

1.4 Self-Organizing Map Algorithm 14

1.5 Formal Analysis of the Self-Organizing Process 16

1.5.1 Vector Quantization 16

1.5.2 Energy Functions 19

1.5.3 Ordering proofs 21

1.5.4 Convergence theorems 23

1.6 Other Analysis Papers of Self-Organizing Process 28

II Analysis of Pattern Sequences by the Self-Organizing Map **30**

2 Visualization of Pattern Sequences by Self-Organizing Maps **31**

2.1 Detection of Dysphonic Voices 31

2.1.1 Self-Organizing Map in Visualization of Speech 32

2.1.2 Speech Data Used 33

2.1.3 Acoustic Maps 34

2.1.4 Results 35

2.1.5 Discussion 36

2.2 Analyses on Articulation 38

2.2.1 Recognition of /s/ Misarticulation 38

2.2.2 Detection of Co-articulation Effects 39

2.3 Process Monitoring and Control by Self-Organizing Maps 41

2.3.1 Process State Monitoring 41

3 Utilizing Contextual Information with the Self-Organizing Map **43**

3.1 Neural Networks and Analysis of Pattern Sequences . . . 43
3.1.1 Time-Delay Neural Network . . . 44
3.1.2 Recurrent Error Back-Propagation . . . 45
3.1.3 Other Models . . . 45
3.2 Speech Recognition . . . 46
3.2.1 Preprocessing of the Speech Signal . . . 47
3.2.2 Labeling by Phonotopic Map Method . . . 47
3.2.3 Segmentation of a Quasiphoneme String . . . 48
3.2.4 Later Improvements . . . 48
3.3 Recognition of Non-Stationary Phonemes by Auxiliary Maps . . . 49
3.3.1 Experiments . . . 50
3.4 Time-Delay Self-Organizing Map . . . 51
3.4.1 Time-Delay SOM . . . 51
3.4.2 Experiments . . . 52
3.5 Self-Organizing Maps with Leaky Integration of Signals . . . 53
3.5.1 SOM Responses . . . 54
3.5.2 Storage of the Signal in the Integrators . . . 54
3.5.3 Experiments with Speech Data . . . 55
3.6 Hypermap Structures . . . 55
3.6.1 Experiments . . . 56
3.7 Other Studies and Applications . . . 57
3.7.1 Analysis of Protein Sequences . . . 57
3.7.2 Analysis of Syntactic and Semantic Knowledge using Self-Organizing Maps . . . 58
3.7.3 Image Analysis by Self-Organizing Maps . . . 59
3.7.4 Analysis of Medical Signals and Other Sequences . . . 60

III Future Directions **61**

4 Direct Representation of Pattern Sequences by Self-Organizing Operator Maps **62**
4.1 Operators in a Self-Organizing Map . . . 62
4.2 Experiments with SOM and Operators . . . 63
4.3 Speech Data . . . 64
4.4 Genetic Training . . . 65
4.4.1 Results . . . 65
4.5 Gradient Training . . . 68
4.5.1 Results . . . 69
4.6 Discussion . . . 70

5 Discussion **72**

List of Notations

The main symbols and abbreviations used in this thesis are collected below.

c	index of best matching unit
i	unit index
t	time variable
$m_i(t), m_i$	(model) weight vector of unit i
$x(t), x$	input vector
r_i	coordinate vector of unit i
$\alpha(t), \alpha$	scalar adaptation gain
$h(i, c, t), h(r, t)$	neighborhood function

ANN	Artificial Neural Network
DEC	Dynamically Expanding Context
HMM	Hidden Markov Models
LPC	Linear Prediction Coding
LVQ	Learning Vector Quantization
SOM	Self-Organizing Map
TDNN	Time Delay Neural Network

Introduction

Artificial neural networks, connectionists models, have become a popular field of research in a number of scientific fields. Two different approaches are used. In the first approach, the artificial neural networks are modeled after true biological neural networks. The aim is to get as good as possible approximations of the true neural networks to be utilized in the research work.

The second approach which comprises most research on the artificial neural networks, however, concentrates on applying the principles of natural neural systems to some specific applications. In these applications the highly parallel and adaptive properties of true neural systems are simulated. The requirements of application efficiency set strict constraints to the system models, thus preventing the usage of truly representative biological models, and as a result the applied models are quite simple computationally, when the parallelism and adaptivity of the models have been emphasized.

There are many different competing artificial neural network models that share some common features. For the taxonomy and systematic treatment of some popular neural network algorithms, see for example [Lippmann 1987]. The following definition applies to most of the artificial neural network models [Kohonen 1988a]:

> *'Artificial neural networks' are massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of the world in the same way as biological nervous systems do.*

This work concentrates on one specific model, the Self-Organizing Map (SOM) algorithm proposed by professor Teuvo Kohonen (see [Kohonen 1982c] [Kohonen 1990]). Alongside with the theoretical studies on the SOM itself, the emphasis on this work has been on sequential patterns and analysis of them using the Self-Organizing Map algorithm.

The Self-Organizing Map algorithm is an unsupervised learning algorithm that creates topological mappings between the input data and map units: If two input patterns are similar, then the most active units responding to the two input patterns are located near each other on the map. These mappings resemble findings of physicians in that some sensory processing areas of brain are also spatially ordered. An example is the auditory cortex where different frequencies have representative areas that are organized spatially.

Originally the SOM algorithm was presented as an example of a process which can create a neighborhood preserving mapping of input patterns into a layer of independent interconnected units. Since the goal was to construct *spatial* mappings, the correlations between successive input patterns were not considered; each pattern was taken independently of all the others. For example, in the first experiments with speech data, the phoneme classification task, the correlations between input patterns were neglected: Each phoneme was taken as a stationary state of the signal, and coarticulation effects were treated as noise which degraded the classification. These static mappings already have plenty of applications.

On the contrary, natural signals usually form dynamic sequences, and the requirement to analyze these pattern sequences by Self-Organizing Map algorithm could not be left unchallenged. The first attempts to analyze pattern sequences with the SOM algorithm were rather straightforward, often heuristically motivated systems where the input pattern sequences were transformed into 'static' vectors before inputting them into the SOM. The transformations were conducted, for example, using a shift register where a sequence of patterns was stored in parallel.

The original Self-Organizing Map algorithm was also applied to visualization of pattern sequences. A visualization system mapped an input pattern sequence into another sequence on the map surface, making an automatic dimensionality reduction while preserving the neighborhood relationship of the input patterns. The algorithm offers an easily understood method for visualization of any multi dimensional pattern sequence.

The above methods did not modify the Self-Organizing Map algorithm itself, the modifications arise from the manipulations of the original pattern sequences or in the interpretation of the mapping results. 'True' modeling of dynamic sequences on SOM requires that the algorithm itself be modified. For example, if the distance function, used in finding the best match among the map units were generalized as an operator, as a filter for dynamic signals, the units themselves would become adaptive detectors of dynamic phenomena. This Operator Map could then be applied as an analysis tool for pattern sequences.

Contents of the Thesis and Contributions of the Author

This work is organized in three parts. In the first part (Chapter 1) the Self-Organizing Map algorithm is introduced and it consists mostly of a review and summary of the formal analysis papers written about the Self-Organized algorithm since the first analysis [Kohonen 1981]. The author was responsible for selecting the analysis papers, reviewing them, and finding an appropriate categorization of the papers for the summary.

The second part of the work (Chapters 2 and 3) concentrates on the problem of analyzing pattern sequences with the basic Self-Organizing Map algorithm. Most of the work is in the field of analysis of speech signals, either for a quality analysis of voice or for speech recognition purposes. Other application works are also presented.

In Chapter 2 it is shown that the SOM might be used as an aid for visualization of pattern sequences. The use of the Self-Organizing Map algorithm in the analysis of voice quality was suggested by Dr. Lea Leinonen. The author has been responsible for the application of the Map algorithms and for the developing of the diagnostic programs used in the experiments with speech analysis. ([Leinonen et al. 1991], [Leinonen et al. 1992], [Leinonen et al. 1993a], [Leinonen et al. 1993b], [Mujunen et al. 1993], [Utela et al. 1992]) Visualization using the Self-Organizing Map has also been used in some other applications. The author has contributed in [Kasslin et al. 1992] where the visual display was used in monitoring the working conditions of a technical device.

Chapter 3 consists of several studies where the contextual information in the pattern sequences are utilized. In these studies the pattern sequences have been transformed to 'static' patterns suitable for the Self-Organizing Map algorithm. The input patterns for the SOM algorithm has been constructed for example by storing a sequence of patterns in a register in a parallel form. Most of the discussion in Chapter 3 consist of experiments where SOM has been applied to speech recognition. The author contributed in the development of the speech recognizer described in chapter 3.2. The first independent contribution by the author was to add the auxiliary Self-Organizing Map to the system, described in section 3.3. Author was solely responsible for the studies described in sections 3.4 and 3.5 where some time dependent

components were used in conjunction with the Self-Organizing Map. ([Kohonen et al. 1988], [Kangas et al. 1989], [Kangas 1990], [Kangas 1991a], [Kangas 1991b], [Torkkola et al. 1991]) At the end of the chapter some other application works are explained.

In the third part (Chapter 4) the Self-Organizing Map algorithm itself is modified to make it possible to analyze dynamic signals directly. It is demonstrated that the resulting Operator Maps could be used, for example, to categorize speech waveforms. The general idea of using dynamic operators as the processing elements in SOMs was suggested by prof. Teuvo Kohonen. The author has performed all of the feasibility studies, programming and the final experiments with speech data using the operator maps described in Chapter 4.

Results from the following list of papers, where author has contributed, have been used in the thesis.

[Kangas et al. 1989] Jari Kangas, Teuvo Kohonen. Transient map method in stop consonant discrimination. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech89)*, Paris, France, September 26-28 1989.

[Kangas 1990] Jari Kangas. Time-delayed self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks*, pages II 331–336, San Diego, June 1990.

[Kangas 1991a] Jari Kangas. Phoneme recognition using time-dependent versions of self-organizing maps. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 101–104, Toronto, Canada, May 14-17 1991.

[Kangas 1991b] Jari Kangas. Time-dependent self-organizing maps for speech recognition. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1591–1594. North-Holland, June 1991.

[Kasslin et al. 1992] Mika Kasslin, Jari Kangas, Olli Simula. Process state monitoring using self-organizing maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1531–1534. North-Holland, September 1992.

[Kohonen et al. 1988] Teuvo Kohonen, Kari Torkkola, Makoto Shozakai, Jari Kangas, Olli Ventä. Phonetic typewriter for Finnish and Japanese. In *Proceedings of the IEEE 1988 International Conference on Acoustics, Speech, and Signal Processing, New York, N.Y., April 11-14*, pages 607–610, 1988.

[Leinonen et al. 1991] Lea Leinonen, Jari Kangas, Kari Torkkola, Anja Juvas. Pattern recognition of hoarse and healthy voices by the self-organizing map. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1385–1388. North-Holland, June 1991.

[Leinonen et al. 1992] Lea Leinonen, Jari Kangas, Kari Torkkola, Anja Juvas. Dysphonia detected by pattern recognition of spectral composition. *Journal of Speech and Hearing Research*, 35:287–295, April 1992.

[Leinonen et al. 1993a] Lea Leinonen, Tapio Hiltunen, Kari Torkkola, Jari Kangas. Self-organized acoustic feature map in detection of fricative-vowel coarticulation. *Journal of the Acoustic Society of America*, 93(6):3468–3474, June 1993.

[Leinonen et al. 1993b] Lea Leinonen, Riitta Mujunen, Jari Kangas, Kari Torkkola. Acoustic pattern recognition of fricative-vowel coarticulation by the self-organizing map. *Folia Phoniatrica*, 45:173–181, 1993.

[Mujunen et al. 1993] Riitta Mujunen, Lea Leinonen, Jari Kangas, Kari Torkkola. Acoustic pattern recognition of /s/ misarticulation by the self-organizing map. *Folia Phoniatrica*, 45:135–144, 1993.

[Torkkola et al. 1991] Kari Torkkola, Jari Kangas, Pekka Utela, Sami Kaski, Mikko Kokkonen, Mikko Kurimo, Teuvo Kohonen. Status report of the finnish phonetic typewriter project. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–771–776. North-Holland, June 1991.

[Utela et al. 1992] Pekka Utela, Jari Kangas, Lea Leinonen. Self-organizing map in acoustic analysis and on-line visual imaging of voice and articulation. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–791–794. North-Holland, September 1992.

Part I

Theoretical Motivation of the Self-Organizing Map

Chapter 1

Self-Organizing Maps

Artificial Neural Networks (ANNs) constitute a very large and diverse field of techniques where ideas about the principles of natural neural networks are utilized. The research into ANN algorithms (which began more than 40 years ago, [Lippmann 1987]) has increased rapidly in the last ten years, due to the development of some new learning algorithms and the advent of more powerful computing machines to simulate computationally heavy algorithms. It is also believed that the massive parallelism offered by neural network implementations is essential for real-time solutions to many practical problems.

During the last few years several collection works have been published where the basic theory and explanatory applications are described. For a thorough introduction to the subject see [Anderson et al. 1988], [Sánchez-Sinencio et al. 1992], [Mehra et al. 1992] and [Lau 1992] where some basic research papers have been reprinted. To obtain a basic understanding of the artificial neural network methods, read [Lippmann 1987] which contains a short and simple introduction to the ideas behind the ANN models.

The artificial neural networks ('neural nets') consist of a great number of simple computing elements which are connected to each other via unidirectional signal channels. The computing elements operate in parallel. These models have great potential in application areas where there are many competing hypotheses about the pattern classes tested and each test requires significant amount of computing, making the usage of parallel processing a necessity (such as speech and image recognition).

The neural network models (the connection topologies, computational element capabilities and the associated learning algorithms) differ greatly. Categorization of the neural network algorithms based on their properties has been difficult; one categorization is given in [Kohonen 1990]:

- Feedforward nets which transform the input signals into output signals. The transformation is usually trained in a supervised manner.
- Feedback nets which define the initial activations of the computing elements from input signals, and execute a cycle of state transitions, the outcome of which is the output signal. The connection strengths are usually given (not trained) to the network.
- Competitive, self-organizing nets where the computing elements compete against each other in the activations. The training is done in an unsupervised manner since the network adapts the connection strengths based on the properties of the input data.

The capabilities of the computational elements differ from one artificial neural network model to another. The elements are, however, usually nonlinear, have plenty of input channels and execute rather simple computations. The parameters of the computing elements are usually adaptive, thus enabling the system to learn.

Many of the early applications of neural network models were pattern recognition systems. The models are naturally suited for 'difficult' recognition tasks where the ability of neural network models to learn the data properties can be utilized advantagely. The models provide a greater degree of robustness than the 'traditional' sequential computing models, because neural models utilize so many processing elements. Additionally the neural network models are non-parametric and make rather weak assumptions of the underlying structure of the signal source. The artificial neural network models can be more robust than the traditional pattern classification models when the signals are generated by a nonlinear processes. A set of common artificial neural network models which are used in pattern recognition applications is described in [Lippmann 1987].

1.1 Self-Organizing Maps

Self-Organizing Map, SOM, is one of the most widely used artificial neural network algorithms which uses unsupervised learning (see [Rappa et al. 1992] for research community survey). The algorithm was introduced by professor Teuvo Kohonen in [Kohonen 1981] and [Kohonen 1982c]. An extensive review of the algorithm and related subjects was published in [Kohonen 1990].

The Self-Organizing Map algorithm belongs to the group of unsupervised (competitive, self-organizing) learning algorithms. SOM is usually represented as a two dimensional neural network sheet, whose units, neural cells, become tuned to different input signal patterns. Those units which are best tuned to a given signal pattern become active and a response is concentrated in the area with the most active units in the sheet. The important difference from other neural network models is the phenomenon occurring in the Self-Organizing Map that after learning, the responses are ordered in the map. That is, two similar input patterns which are near each other in the signal space, are correspondingly located near each other in the map sheet.

The development of the algorithm was inspired by the findings of physicians that some areas of brain tissue are organized according to the input signal. The Self-Organized Map algorithm was introduced as a process which can create such an ordered mapping of input signals (as in the brain) into a layer of independent interconnected units. The ordering takes place automatically without external supervision based on only the internal relations in the structure of input signals themselves and on the coordination of the unit activities through the lateral connections between the units.

Through the (self-)organization of the responsive areas in the map, the Self-Organizing Map algorithm creates an internal representation of the incoming signal structure. The responses to different input patterns are organized in an ordered fashion, similar input patterns produce similar responses. Furthermore, the more frequently inputed patterns gain more area in the map surface to give a more detailed response.

Since the introduction of the Self-Organizing Map, the algorithm has been studied as a pure mathematical model of internal representations of pattern vectors in computational artificial neural network structures. The training algorithm in practical studies has often been simplified to attain more speed for technical applications.

The Self-Organizing Map algorithm and the resulting mappings have been used in many practical applications. The most common application areas are pattern recognition and process control. The ordered organization of pattern responses in maps provides some advantages in error tolerance and in smooth function mappings compared to some other neural network architectures.

1.2 Topographic Maps in Brain

By physiological studies it has been found out that some sensory processing areas of the brain are spatially organized according to the input stimulus. A popular example of this phenomenon is the auditory cortex where tonotopic maps can be found. Different frequencies in the auditory signal have representative areas that are organized in an ordered fashion.

Other spatially organized areas exist. In [Knudsen et al. 1987] (also in [Anderson et al. 1990]) a great number of references to different kinds of sensory signal processing maps in the brains are collected. Topographically organized 'computational' maps exist in vision areas (representations of the angle of tilt of a line stimulus, representations of movement direction), in auditory areas (representations of frequency, representations of amplitude, representations of time intervals between acoustic events) and in motor control areas (control of eye movements) of the brain.

Other more abstract topographic maps have been found in the brains. For example, a map for representation of a sound source location [Knudsen et al. 1987]. The representation depends on the interactions of multiple parameters, some of them mediated through 'lower' level maps processing e.g. the interaural differences in an acoustic signal.

1.3 Physiological Models of the Self-Organizing Map

In the early papers (e.g. [Kohonen 1981] and [Kohonen 1982c]) the Self-Organizing Map algorithm was described as a system of dynamic equations working on functionally identical units. Most of the details in the system were ignored and the functioning of the model was described qualitatively. However, it was shown that by assuming some rather general behaviour about the system components the desired phenomenon can be observed.

The self-organizing map based on dynamic equations was further improved in [Kohonen 1988c]. A full system was not described until in a series of papers [Kohonen 1992], [Kohonen 1993c] and [Kohonen 1993b] where the dynamical equations were described in more detail. Although the basic assumptions about the model were (slightly) changed from the earlier papers, the original behaviour of the SOM algorithm was still preserved: The processing units in the map were automatically adapted by the input signal so that the resulting mapping from input signal space to the map surface was ordered.

The detailed system equations were also simulated in a serial computer, and thus the proper functioning of the map algorithm was ensured. The system simulations were very time consuming, however, and the full system could only be implemented effectively using truly parallel computers where a great number of processors could be used. Such a simulation was earlier described in [Miikkulainen 1991], when a slightly simplified algorithm was implemented in a massively parallel computer.

1.4 Self-Organizing Map Algorithm

In the biologically inspired models described above, the self-organization was defined in its parallel form where each unit is functionally independent. There are no synchronized clocks or other higher level controllers coordinating unit activities. The weight vectors of neighborhood units, however, come to resemble each other because of various kind of lateral feedback connections that correlate the activity of nearby units during the best matching unit search.

Models which are faithful to their biological models, are always computationally difficult to simulate using the current serial computers. In practical applications it is not necessary to mimic the biological components to achieve the functional appearance of Self-Organizing

Maps. We can make some approximations to simplify the system while maintaining similarity to the original processes.

In the following a typical simplified version of the the Self-Organizing Map algorithm is defined. It consists of two steps that are iterated for every sample (see e.g. [Kohonen 1990]):

Step 1. *Finding the best matching units.*

For each input vector $x(t)$ used in the training of SOM the best matching cell in the map is found. The notion of 'best match' is defined by the distance function used. Usually some very simple functions are selected. In most instances the best matching unit c is the unit with the weight vector m_i closest to input vector x in Euclidean space

$$\|x(t) - m_c(t)\| = \min_i\{\|x(t) - m_i(t)\|\}. \tag{1.1}$$

A common variation is the inner product of the vectors $x^T m_i$ (which, however, requires normalization of the vector lengths).

Step 2. *Adaptation of the weights.*

The weight vector values m_i are adapted (tuned) so that the match of the modified weight vectors in the active area (defined in the neighborhood of the best matching unit) and the input vector $x(t)$ is improved. Some detailed derivations of the adaptation laws have been published (see, for example, [Kohonen 1991a], [Luttrell 1990]) where some criterion function has been optimized. The resulting adaptation laws have the same general form (if Euclidean distance function has been used):

$$m_i(t+1) = m_i(t) + \alpha(t)h(i,c,t)[x(t) - m_i(t)], \quad \forall i, \tag{1.2}$$

where $\alpha(t)$ is a scalar adaptation gain, $0 < \alpha(t) < 1$, c is the index of the best matching unit of step 1 and $h(i,c,t)$ is the neighborhood of the winning unit c (i.e. $h(i,c,t)$ determines the units i that are adapted).

The neighborhood function $h(i,c,t)$ ($0 < h(i,c,t) < 1$) defines the activity of neighboring units. Usually the value of the neighborhood function depends only on the distance between the units and time, $h(i,c,t) = h(\|r_i - r_c\|, t)$ (r_i and r_c are the coordinates of units i and c, respectively). In the 'traditional' learning algorithm (e.g. in [Kohonen 1988c]) the neighborhood function was defined as

$$h(i,c,t) = \begin{cases} 1, & \text{if } \|r_i - r_c\| < s(t), \\ 0, & \text{otherwise,} \end{cases} \tag{1.3}$$

where $s(t)$ is some decreasing function of time. The value of $s(t)$ is usually large at the beginning of learning (comparable to the radius of map) and decreases during the training.

In other papers (e.g. in [Kohonen 1990]) the neighborhood function has been given more general forms. For example, a bell shaped function

$$h(i,c,t) = h_0(t)\exp\left(\frac{-\|r_i - r_c\|^2}{\sigma(t)^2}\right), \tag{1.4}$$

where $h_0(t)$ and $\sigma(t)$ are suitable decreasing functions of time.

The neighborhood function can be defined quite freely, and despite the selection, the overall effect of smooth mapping from the input pattern space to the output space is preserved. The resulting mappings thus have the following topographic neighborhood property:

> If some input pattern vector $x(t)$ is mapped to unit i, then all of the (possible) input pattern vectors 'near' $x(t)$ in the input space are mapped also to i or to its close neighbors in the topology of map.

1.5 Formal Analysis of the Self-Organizing Process

Some mathematical analysis of the Self-Organizing Map was done in [Kohonen 1982a]. Using computational example cases some properties of the self-organization process were studied. All of these cases were one dimensional (one dimensional input and one dimensional map, i.e. chain of units).

Many research papers have been published concerning the analysis of SOM algorithm. The mathematical problems considered in those theoretical studies include:

1. *Vector quantization properties of the Self-Organizing Map.* The problem is to find out the possible advantage of using weight vectors given by the SOM algorithm in vector quantization applications instead of vector quantizations given by more traditional quantization algorithms (like *k-means*).

2. *Derivation of SOM learning algorithm from some energy function.* The problem is to find out if there exists an energy function of the weight vectors and the input pattern vectors that would be minimized by the given adaptation laws.

3. *Ordering of the units in the map.* The problem is to find out whether the weight vectors are modified by the stated adaptation laws in such a way that the ordering of the map is ensured. That is, if the weight vectors of the units, having initially random values, will be ordered in the topological order through the iteration of the above mentioned two training equations.

4. *Convergence of the weight vectors in the map.*

   (a) *Locations of the weight vectors.* The problem is to find out where the weight vectors are located after learning (apart from the ordering property), and if there are any such stable convergence locations at all.

   (b) *The magnification factor of the map if the probability density function of input patterns varies in the input space.* The problem is to find out how the unit density reflects the density of input sample vectors (i.e. how the density of weight vectors in map units varies depending on the variations in the density of input samples in the input space).

1.5.1 Vector Quantization

Although the SOM algorithm was originally based on modeling the topological mappings of the brain with an idealized mathematical model where local learning functions were used, the training algorithm could also be derived from the theory of vector quantization. In [Luttrell 1990] ([Luttrell 1989a] and [Luttrell 1989b]) the derivation of SOM algorithm from general vector quantizer equations was shown. In the following the main points of the derivation in [Luttrell 1990] are described.

The idea of vector quantization is to reduce the amount of data required to store a vector by replacing each entry in the data sets by a good representative of an entry selected from a given set of code vectors. Designing a good (representative) code vector set is the main problem in vector quantization.

The distortion error $d(x, x')$ due to quantization process is given by the Euclidean distance between the original input sample vector x and the quantized code vector x'

$$d(x, x') = \|x - x'\|^2. \tag{1.5}$$

With a given training set x one is trying to find out such a code vector set that one can minimize the average distortion D of the quantization

$$D = \int P(x)\|x - x'(y(x))\|^2 \, dx, \tag{1.6}$$

where $P(x)$ is the probability density function of the original input samples x, and x' is determined by the quantization process from x.

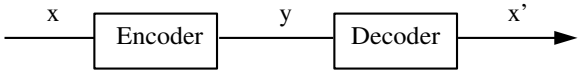

Figure 1.1: *Representation of vector quantization process. The input vector x is first encoded to a code word y and after transmission this label is decoded back to vector form x'.*

The quantization process is analogous to an encoder-decoder system (see Figure 1.1).

The minimization of the quantization error can be done using, for example, the Linde-Buzo-Gray-algorithm [Linde et al. 1980]. The algorithm searches for good quantization values in an iterative way, to fulfill the necessary conditions of an optimal solution:

A1 Given the input vector x and the code vectors x' for each code word y, the code word $y(x)$ is selected so that the distortion $\|x - x'(y(x))\|^2$ is minimized (i.e. y is selected such that the corresponding $x'(y)$ is the nearest neighbor of x).

A2 Given the code word y, the x' is the weighted centroid of those x which are coded to y ($x' = \int_{(x;y(x)=y)} x P(x)dx \,/\, \int_{(x;y(x)=y)} P(x)dx$). Thus the code vectors x' are determined given the coding of each input sample vector x to code words y.

We may include a general noise source into the transmission channel (in Figure 1.1). The noise is such that it can change the code word y to some other code word y'. The modified system is shown in Figure 1.2.

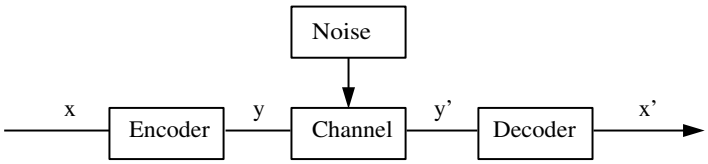

Figure 1.2: *The representation of vector quantization process with a random modifier in the transmission stage of codes. The code word y may be changed to another code word y' (e.g. by thermal noise).*

If the noise function changing the original code word y to y' is modeled as an additive noise source acting on y we can reformulate the distortion measure as

$$D_1 = \int P(x) \int \pi(n)\|x - x'(y(x) + n)\|^2 \, dn \, dx, \tag{1.7}$$

where $\pi(n)$ is the probability density function of the noise.

Optimization of modified quantizer

The optimization of the modified distortion measure now requires a somewhat different solution than the original distortion measure. However, the definition of a good solution can be stated in a similar way. The conditions to be fulfilled for an optimal solution are:

B1 Given the input vector x and the vectors x' for each code word y, the code word $y(x)$ is selected so that the distortion $\int \pi(n)\|x - x'(y(x)+n)\|^2 dn$ is minimized (i.e. y is selected such that even if the code is modified to $y + n$, the weighted distortion (weighted by the probability density function of the noise) is still minimized).

B2 Given the code word y, the decoding $x'(y)$ is determined by

$$x'(y) = \frac{\int P(x)\pi(y - y(x))\ x\ dx}{\int P(x)\pi(y - y(x))\ dx}. \tag{1.8}$$

(I.e. the $x'(y)$ is the (weighted) centroid of those x vectors which are coded to one of the possible code words $y + n$. Thus the vectors x' are determined given the coding of each vector x to code words y).

The learning rules for obtaining good vector values x' for such a modified vector quantization system may now be stated. In [Luttrell 1990] it is shown that if some rather general conditions (e.g. zero mean noise model for $\pi(n)$) are met, B1 could be approximated by A1 (i.e. the label y was determined by the nearest neighbor encoding rule).

Given the previous labeling rule, we can determine the learning rule for x' that fulfills the condition B2. The update rule is expressed in the form of the stochastic gradient descent algorithm [Luttrell 1990]: All the code vectors $x'(y')$ (selected by labels y' for which $\pi(y' - y(x)) > 0$) are adapted according to

$$x'(y') \leftarrow x'(y') + \epsilon\pi[y' - y(x)][x - x'(y')], \tag{1.9}$$

where $0 < \epsilon < 1$ and $y(x)$ is the label selected by the (reduced) nearest neighbor rule B1 (i.e. all those code vectors x' which might be selected as the representatives of input sample vector x (after noisy labeling) are modified towards the input sample vector). The adaptation constant is dependent on the probability density function π, the probability of changing the label.

To check the consistency of the results one might define the probability density function as $\pi(n) = \delta(n)$ (a Dirac delta function) and see that the above conditions B1 and B2 are reduced to the standard vector quantization conditions A1 and A2.

The update rule 1.9 is functionally identical to the update rule of Kohonen (1.2). The probability density function $\pi(n)$ is equivalent to the neighborhood function $h(i, c, t)$ in (1.2), to the extent that those labels which are easily confused in the encoding process have similar code vectors. Thus the overall distortion is minimized in the encode/decode system despite the noise.

Applications of the SOM vector quantization

A good example of the use of the robust mapping property of Self-Organizing Map in vector quantization in noisy environment was described in [Bradburn 1989]. There the problem was to design a codebook for digital transmission of speech where noise in the transmission line caused some bit substitutions in the code labels. The objective was to select such codes for the signal amplitude values that despite the bit substitutions the signal was changed as little as possible in the reconstruction stage.

Using the SOM algorithm in the design process the author was able to double the number of allowable bit substitutions (thus allowing the transmission line characteristics to become worse) without an unacceptable degradation in speech signal quality compared to traditional signal value quantization algorithms.

Another example of using the Self-Organizing Map in vector quantization was explained in [Carrato et al. 1993] where the ordering property of Map algorithm was utilized. The objective was to compress image data with vector quantization algorithm. The idea was to

predict the code vector of a subimage by the code vector of a neighboring subimage, and to save only the difference between the prediction and the true codes. The topological nature of the map was utilized when the prediction errors (in the index terms) were small due to the correlations of the nearby subimages.

The above idea was used also in [Jumpertz et al. 1993] where image sequences were coded. The topological properties of SOM were utilized in coding subsequent images with the difference coding of corresponding code vector indexes.

1.5.2 Energy Functions

The formal studies of Self-Organizing Maps and their properties would be greatly simplified, if it could be proven that the adaptation laws are derivable from some energy functions that are minimized. Traditional techniques (for example, the Liapunov function approach[1]) could then be used to ensure the convergence properties of the algorithm.

The possibility of the existence of energy functions has been studied in many papers, see e.g. [Kohonen 1991a], [Erwin et al. 1992a], and [Tolat 1990]. In [Erwin et al. 1992a] it was shown that for a general case it is not possible to create any such global energy function. The best possible construction would be a system consisting of a set of energy functions, one for each weight value (proposed in [Tolat 1990]). Such a set of energy functions was described in detail in [Erwin et al. 1992a].

Existence of energy functions

In [Erwin et al. 1992a] one dimensional maps were used in the system derivations. The following is a short explanation of these derivations.

To simplify the computation of border points between the Voronoi cells[2] in the case of non-ordered chain of units we now define permutation functions correlating the original and 'new' indexes. The new indexes are such that the corresponding weight values grow as the index grows, i.e. if $a < b$ then $u_a < u_b$. The transformation from the new index a to the old one i is $i = T(a)$.

The average change $V_i(m)$ of the weight vector m_i from the adaptation step is

$$V_i(m) = \alpha \sum_{c=1}^{N} h_{ci} \int_{x \in \Omega(c)} (x - m_i) P(x)\, dx, \tag{1.10}$$

where c is the index of the winning unit in a chain of N units and $\Omega(i)$ is the Voronoi cell of unit i. From the assumption that the map is one dimensional and taking the base of input pattern to be the set $x \in [0, 1]$, the Voronoi cells can be given as:

$$\begin{aligned} \Omega(1) &= \{x | 0 < x < (m_{T(1)} + m_{T(2)})/2\}, \\ \Omega(n) &= \{x | (m_{T(n-1)} + m_{T(n)})/2 < x < (m_{T(n)} + m_{T(n+1)})/2\}, \\ &\qquad \text{for } 1 < n < N, \\ \Omega(N) &= \{x | (m_{T(N)} + m_{T(N-1)})/2 < x < 1\}. \end{aligned} \tag{1.11}$$

Assuming that the probability density function of samples $P(x)$ is constant, and taking the above Voronoi cell definitions into account, we get an average change

$$V_{T(a)}(m) = \sum_{b=2}^{N-1} h_{T(b)T(a)} [(m^2_{T(b+1)} - m^2_{T(b-1)})/8$$

[1] The key idea in Liapunov function approach is to find an energy function that decreases on every possible training step. Therefore the system converges to the minimum of the energy function.

[2] A Voronoi cell of unit i consists of those points which are closer to weight vector m_i than to any other weight vector.

$$
\begin{aligned}
&+m_{T(b)}(m_{T(b+1)} - m_{T(b-1)})/4 + m_{T(a)}(m_{T(b-1)} - m_{T(b+1)})/2] \\
&+h_{T(1)T(a)}[(m_{T(1)} + m_{T(2)})^2/8 - m_{T(a)}(m_{T(1)} + m_{T(2)})/2] \\
&+h_{T(N)T(a)}[1/2 - m_{T(a)} - (m_{T(N)} + m_{T(N-1)})^2/8 \\
&+m_{T(a)}(m_{T(N)} + m_{T(N-1)})/2].
\end{aligned} \tag{1.12}
$$

The change can be taken as an average force acting on the weight $m_{T(a)}$. Now the problem is to find out if there is a potential function $E(m)$ such that

$$
\partial E(m)/\partial m_i = -V_i(m). \tag{1.13}
$$

The potential function $E(m)$ would be the required Liapunov function. In [Erwin et al. 1992a] it was proven that there cannot be such a function.

System of energy functions

In [Tolat 1990] it was suggested that instead of a global energy function, a set of energy functions with one for each unit could be used. The adaptation of the weight values could then be derived by minimizing each individual energy function separately. In [Tolat 1990] a simple system of three units in a one dimensional map was considered and corresponding energy functions were derived.

In [Erwin et al. 1992a] the set of energy functions was given in more general terms. The energy function for unit $i = T(a)$ was

$$
\begin{aligned}
E_{T(a)}(m) &= \tilde{E}_{T(a)}(m) + X_{T(a)}(m) \\
&= \epsilon \sum_{b=1}^{N} h_{T(a)T(b)} \int_{x \in \Omega(b)} \frac{1}{2}(x - m_{T(a)})^2 P(x) dx \\
&\quad + \frac{\epsilon}{48} \sum_{b=2}^{N} (m_{T(b)} - m_{T(b-1)})^3 (1 - h_{T(b)T(b-1)}) P(\frac{1}{2}(m_{T(b)} + m_{T(b-1)})).
\end{aligned} \tag{1.14}
$$

The above energy function consists of two terms. The first term is the weighted (quantization) error given by the locations of the weight values. The second term is a correction to the total energy arising from the changes in the Voronoi cell locations during an adaptation step. When the map is ordered the second term is generally quite small and the first term dominates. In the beginning of the learning, when the map is not ordered, the second term contributes significant values. In [Erwin et al. 1992a] the above set of energy functions was additionally generalized to multi dimensional maps.

Error function: discrete input data sets

In [Ritter et al. 1988b] a slightly different approach was described. An error function was defined for a discrete case of SOM algorithm where the input set consisted of a finite number of samples. In the derivation of the error function it was noted that the Self-Organizing Map algorithm is a Markov process with some transition probabilities between states m comprising of all the weight vectors m_r.

For a discrete probability density $P(x) = \sum_{i=1}^{N} p_i \delta(x - x_i)$ of samples $x_i \in R^n$ there exists an error function $V(m)$ (h_{ci} is the neighborhood effect between units c and i)

$$
V(m) = \frac{1}{2} \sum_{ci} h_{ci} \sum_{x_j \in F_c(m)} p_j (x_j - m_i)^2, \tag{1.15}
$$

with $F_c(m)$ defining the Voronoi cell of the best matching unit c

$$
F_c(m) = \{x \in R^n | \ \|x - m_c\| \leq \|x - m_i\| \ \ \forall m_i\}. \tag{1.16}
$$

The expected change of the error function $V(m)$ is

$$E(\Delta V|m) = -\epsilon \sum_i \|\nabla_{m_i} V\|^2. \tag{1.17}$$

From the above consideration it can be seen that on average the error V decreases and the algorithm eventually minimizes V.

A somewhat similar error function was derived in [Kohonen 1991a]. A locally weighted mismatch of input sample x was defined as

$$E_1 = E_1(x, m_1, m_2, \ldots, m_N) = \sum_{i=1}^{N} h_{ci} \|x - m_i\|^2. \tag{1.18}$$

Using a discrete set of input samples $S = \{x_1, x_2, \ldots, x_T\}$ one can define a global mismatch metric

$$E_2 = \frac{1}{T} \sum_{t=1}^{T} \sum_i h_{ci} \|x_t - m_i\|^2, \tag{1.19}$$

which should be minimized.

Using stochastic approximation techniques, the mismatch metric E_2 was minimized using local approximations, by minimizing the instantaneous mismatch of each sample. Thus, the mismatch E_1 was used: the gradient of E_1 was minimized by the adaptation of all weight vectors m_i

$$\begin{aligned} m_i(t+1) &= m_i(t) - \tfrac{1}{2}\alpha(t) \nabla_{m_i(t)} E_1 \\ &= m_i(t) + \alpha(t) h_{ci} [x(t) - m_i(t)]. \end{aligned} \tag{1.20}$$

This adaptation rule is equivalent to the original adaptation rule in Self-Organizing Map algorithm.

Error function: continuous case

The continuous error functional (the energy function) was discussed in [Kohonen 1991a]. The functional is

$$E = \sum_i \int_{x \in \Omega_i} \sum_k h_{ik} \|x - m_k\|^2 p(x) dx, \tag{1.21}$$

where integration is done over the Voronoi cells Ω_i. In the derivation of adaptation laws it was shown that the gradient consisted of two parts (compare [Erwin et al. 1992a]). The first term is due to the changing of the weight vectors inside the previous Voronoi cells, the second term takes into account the changing of the borders of the Voronoi cells.

1.5.3 Ordering proofs

The ordering property of SOM has been examined in many articles since 1981 (see e.g. [Kohonen 1982a], [Kohonen 1988c], [Cottrell et al. 1987], [Erwin et al. 1992a], [Bouton et al. 1991], [Bouton et al. 1992b], [Ferrán 1992]). The proofs are all restricted to the one dimensional cases where the *order of map* can be easily defined. (For an exception see [Lo et al. 1991].)

The general ordering property of SOM can be stated as follows:

> Given a set of n randomly chosen initial (one dimensional, scalar) weight values $\{m_i\}$ in a one dimensional map, and a set of random input samples $x(t)$, the process of self-organization defined by equations (1.1) and (1.2) will adapt the weight values such that the set of weight values become ordered ($m_i < m_{i+1}, \ \forall i < n$ or $m_i > m_{i+1}, \ \forall i < n$). After the weight values are arranged, they do not lose their order.

Index of the order in SOM

There are two basic approaches to this analysis. In the first approach some kind of index is defined as an order metric of the map, and it is shown that there are more re-ordering training steps that decrease this index, than there are such re-ordering training steps that increase this index, until an order is reached (in [Kohonen 1982a], [Kohonen 1988c], [Ferrán 1992]). It also shows that a map cannot loose its order.

The analysis of the ordering process in these papers was formally valid. However, the relative frequencies of the different re-ordering steps occurring in natural training session cannot be determined and therefore the proof was not complete. In [Kohonen 1988c] a more rigorous method of proof was mentioned but it was not derived.

This index method for order proofs was used again in [Lo et al. 1993] where it was shown that the adaptation neighborhood could be defined as follows:

$$h(i,c,t) = \begin{cases} 1, & \text{if } \ i \in \{c\}, \\ \lambda, & \text{if } \ i \in \{c+1, c-1\}, \\ 0, & \text{otherwise}, \end{cases} \tag{1.22}$$

where $1 \geq \lambda \geq 0$. Formally this proof is also valid if the proof in [Kohonen 1982a] is valid. In the current form, however, the proof is incomplete.

Markovian state process

In the second approach, the unit weight values are taken collectively as a state of a Markovian random process. It shows that there is a sequence of input events (with a non-zero occurrence probability) that would order the map (change the weight values so that they become ordered) and since the ordered states of the map are absorbing states of the process and because the inputs are selected randomly, the map is ordered almost surely (i.e. with probability 1) (in [Cottrell et al. 1987], [Kohonen 1988c], [Erwin et al. 1992a], [Bouton et al. 1991] [Conti et al. 1991]).

All of the proofs in the second set of papers are similar. Their main difference lies in the neighborhood function used. In [Cottrell et al. 1987], [Kohonen 1988c], and [Bouton et al. 1991] the neighborhood is defined as the nearest neighbors on both sides of the best matching unit. In [Erwin et al. 1992a] the neighborhood function was selected with more freedom when it was defined as $h(c,i,t) = H(|c-i|)$ such that

$$1 \geq H(0) \geq H(1) > H(2) > \ldots > H(n-2) > H(n-1) \geq 0, \tag{1.23}$$

where n is the number of units in the (one dimensional) map.

Order in multiple dimensions

No valid proofs exist for ordering of map units in higher-dimensional spaces. The main reason is that there does not exist any accepted definition for the ordered states of multi dimensional maps. There has been some attempts to define such a measure (see e.g. [Lo et al. 1991], and [Bauer et al. 1992]).

In [Lo et al. 1991] the order of the map was defined in *arbitrary* dimensions. The map was defined to be 'topologically ordered' if $\|x - m_a\| < \|x - m_b\|$ if and only if $\|c - a\| < \|c - b\|$, where c is the location of the winning unit, a and b are the locations of any other units, x is the input sample vector, and m_a and m_b are the weight vectors of units a and b, respectively. However, the topological order thus defined cannot be defended[3]. The erroneous definition of order leads to erroneous proofs of ordering. (The same definition of order was, by coincidence, given in [Erwin et al. 1992a] but it was not used in any proofs.)

[3]The error in the above definition of order can be clarified with a counterexample:

Topographic product

In [Bauer et al. 1992] the order of the map in any dimension has been defined as a topographic product that is a measure of the preservation of neighborhood relations (i.e. the neighbor samples in input (output) space should be neighbors in output (input) space also) in maps between spaces of possibly different dimensionality. There were no attempts to prove any ordering properties of SOM in [Bauer et al. 1992], the topographic product was used to attain some quantitative values to measure the correspondence between the input space dimension and the topological dimension of the map.

1.5.4 Convergence theorems

The problem of convergence of the weight values is different than the problem of the ordering of units. The weight values may converge to unique fixed values even if the units are not ordered. This is true in cases where the algorithm has local minima due to an unsuitable neighborhood function (see [Erwin et al. 1992b] for examples of local minima in one dimensional maps).

It might also be possible to find the converged weight values of maps in any dimension, while the order is defined only for one dimensional maps (see section 1.5.3).

Convergence: one dimensional maps

The convergence of the weight values were studied in [Kohonen 1982a][Kohonen 1988c]. There the map, as well as the input vectors, were one dimensional. The maps were assumed to be pre-ordered (in ascending order, i.e. if $s > r$ then $m_s > m_r$). The weights remain ordered during the adaptation process, as shown in the ordering proofs.

In [Kohonen 1982a] the expected values of the weights were studied. It was shown that these numbers converge to unique values. The neighborhood function was defined as

$$h_{ci} = \begin{cases} 1, & \text{if } c-1 \leq i \leq c+1 \\ 0, & \text{otherwise.} \end{cases} \tag{1.28}$$

Because of the neighborhood function used, the adaptation laws will affect only the immediate neighbors of the best matching unit. Thus we can determine the interval around each unit i into which the sample x should belong to so that the weight m_i is affected. If there are at least 5 units, the intervals S_i are

$$i = 1 \quad : \quad S_i = [a, \frac{1}{2}(m_2 + m_3)]$$

We can define the weight values directly (now in one dimensional case)

$$m_i = \begin{cases} 0.1 * i/n, & \text{if } i \in \{1, \cdots, n\}, \\ 0.1 + 0.9 \times (i-n)/(N-n), & \text{if } i \in \{n+1, \cdots, N\}, \end{cases} \tag{1.24}$$

where $N = 2 \times n$ is the number of nodes. The weight values are now ordered in ascending order in a very intuitive way ($m_i < m_{i+1}, \quad \forall i < N$).

When we now take $c = n$, $a = n-2$ and $b = n+1$, (and $x = m_c$,) the indexes have

$$\|c - a\| > \|c - b\| \tag{1.25}$$

and the distances between the weight values are

$$\begin{aligned} |x - m_a(t)| &= |0.1 - 0.1 \times (n-2)/n| = 0.1 \times \tfrac{2}{n}, \\ |x - m_b(t)| &= |0.1 - (0.1 + 0.9 \times (n+1-n)/(N-n)| = 0.9 \times \tfrac{1}{n}, \end{aligned} \tag{1.26}$$

from which we can see that

$$\|x - m_a(t)\| < \|x - m_b(t)\|. \tag{1.27}$$

which contradicts the definition in [Lo et al. 1991]. □

$$
\begin{aligned}
i = 2 \quad &: \quad S_i = [a, \frac{1}{2}(m_3 + m_4)] \\
3 \leq i \leq N-2 \quad &: \quad S_i = [\frac{1}{2}(m_{i-2} + m_{i-1}), \frac{1}{2}(m_{i+1} + m_{i+2})] \\
i = N-1 \quad &: \quad S_i = [\frac{1}{2}(m_{N-3} + m_{N-2}), b] \\
i = N \quad &: \quad S_i = [\frac{1}{2}(m_{N-2} + m_{N-1}), b]
\end{aligned}
\tag{1.29}
$$

where $[a, b]$ is the interval from which the input samples x are read and N is the number of units.

For each unit i the expected values $E(m_i)$ of the weights can be solved from the above equations (at least in principle), if the probability density function $p(x)$ is known. In the simplified case when $p(x) =$ constant, the expected values are solutions to the equation

$$
Am + h = 0, \tag{1.30}
$$

where m is the vector of the weight values m_i

$$
m = [m_1, m_2, \ldots, m_N]^T, \tag{1.31}
$$

$$
A = \frac{1}{4}\begin{bmatrix}
-4 & 1 & 1 & 0 & \ldots & & & & \\
0 & -4 & 1 & 1 & & & & & \\
1 & 1 & -4 & 1 & & & & & \\
0 & 1 & 1 & -4 & & & & & \\
\ldots & & & & & & & & \\
 & & & & & -4 & 1 & 1 & 0 \\
 & & & & & 1 & -4 & 1 & 1 \\
 & & & & & 1 & 1 & -4 & 0 \\
 & & & & & 0 & 1 & 1 & -4
\end{bmatrix} \tag{1.32}
$$

and

$$
h = \frac{1}{2}[a, a, 0, 0, \ldots, 0, b, b]^T. \tag{1.33}
$$

The equation 1.30 has a solution

$$
m' = -A^{-1}h, \tag{1.34}
$$

if A^{-1} exists. As shown in [Kohonen 1982a] and [Lo et al. 1993] the inverse matrix A^{-1} exists and therefore the equilibrium solution also exists. In [Kohonen 1982a] it was also shown that the system will converge to the solution from any initial state (that satisfies the conditions stated above).

A similar approach was used in [Lo et al. 1993] to prove the convergence of the weight values in one dimensional maps to certain weight values for a slightly more general neighborhood function (for the neighborhood definition in Equation 1.22). It was shown that there exists a unique solution for the weight values. An extension to the proof of two dimensional maps in two dimensional input spaces was also described. The equations were given and it was then stated that there was a proof for the unique solution of weight values for the two dimensional maps.

Similar approaches were stated in [Bouton et al. 1991] and [Bouton et al. 1992b] where the question of existence of an equilibrium state was considered in a slightly different mathematical framework. The neighborhood function was the same as above (in [Kohonen 1982a]). The existence of an equilibrium was shown, and the convergence of map to that equilibrium was proven for input samples with uniform probability density function. (In [Bouton et al. 1992a]

the existence of an equilibrium was studied in non-uniform probability density functions of input samples.)

Convergence: multi dimensional maps

In [Ritter et al. 1986] the final state of the SOM was studied in more general configurations than in the studies mentioned above. The correspondence between input sample probability density function and the density of weight values of the units were studied in all cases where the input dimension and the map dimension were identical.

The results (the correspondence between the density functions) were given by a pair of equations that have a simple solution in only a few special cases. The first case is the typical one dimensional situation; a one dimensional input space and a one dimensional map. In the second case both the input space and the map can be two dimensional, if the input sample probability density function is a product of two (one dimensional) probability density functions.

In both of the above cases the 'local magnification factor' (the number of units in a unit area of input space) is proportional to the probability density function of input samples

$$M(m) \propto P(m)^{2/3}, \tag{1.35}$$

where M is the local magnification factor and P is the probability density function of the input samples.

The correspondence between the density functions can also be determined in cases where weights m are represented by a complex function $m = (\mathrm{Re}\ \omega, \mathrm{Im}\ \omega))^T$ with ω analytic in $z = x + iy$. Then the correspondence can be found as

$$M(m) \propto P(m). \tag{1.36}$$

As noted, in general the correspondence between the magnification factor $M(m)$ and the input sample probability density function $P(m)$ cannot be found except for one dimensional maps.

In [Ritter et al. 1988a] the convergence studies of the Self-Organizing Maps process were generalized to cases where the input space and map dimensions can be different. The studies were based on generating a Fokker-Blank equation of the behaviour of the map in time, during the adaptation. This approximation was valid when the training constant was kept small and the modifications in the map were small for each input sample.

From the equations one can derive the conditions for the corresponding process (the SOM algorithm) to converge to stationary weight values. The necessary and sufficient conditions were stated:

$$\lim_{t \to \infty} \int_0^t \epsilon(t')dt' = \infty, \tag{1.37}$$

$$\lim_{t \to \infty} \epsilon(t) = 0. \tag{1.38}$$

In [Ritter et al. 1988a] the 'fluctuations' of the map around its equilibrium solution were also studied. For that reason a special kind of input space was constructed where the variances of the first two input vector components were greater than for the rest of the components. Thus the two dimensional map would be naturally oriented so that the array of units initially followed the first two components. However, if the variance in the extra dimensions is increased and it eventually becomes large enough, the map would try to follow the distribution of input samples by creating 'bumps' perpendicular to the plane defined by the first two component axes. In [Ritter et al. 1988a] some results of the 'height' of 'bumps' and the probability of them as function of the ratio of variances were described. It was also shown that when the

ratio exceeds some predefined (critical) value, the map created large distortions, and the map tries to find the most important input dimensions by performing a large reorganization.

The analysis of the dynamical state of Self-Organizing Map during the learning process was continued in two related papers [Der et al. 1993a] and [Der et al. 1993b]. In these papers the behaviour of the map units were studied near the critical value of input space coordinate variations.

Magnification factor

The results above (in [Ritter et al. 1986] and [Ritter et al. 1988a]) were restricted to situations where there is an infinite number of units in the map and the neighborhood function therefore covered an infinite number of units on both sides of the selected unit. In practice the map can consist of only a fixed number of units. In [Ritter 1989] and [Ritter 1991] the situation was restricted so that only n neighbors in both side of the winning unit were adapted in a one dimensional map. It was shown that the asymptotic level density (the magnification factor) of the SOM mapping could be derived exactly.

It turned out that the magnification density depended on the number n of adaptable units on both sides of the selected unit

$$M(m) \propto P(m)^{\alpha}, \tag{1.39}$$

where the exponent α is

$$\alpha = \frac{2}{3} - \frac{1}{3n^2 + 3[n+1]^2}. \tag{1.40}$$

To check the consistency of this result with earlier figures one can solve for α as $n \to \infty$ and arrive at $\alpha = 2/3$ as expected (from the results in [Ritter et al. 1986]). If $n = 0$ (as in traditional vector quantization learning, e.g. *k-means*) alpha becomes $\alpha = 1/3$ which coincides with the results of Zador [Zador 1982].

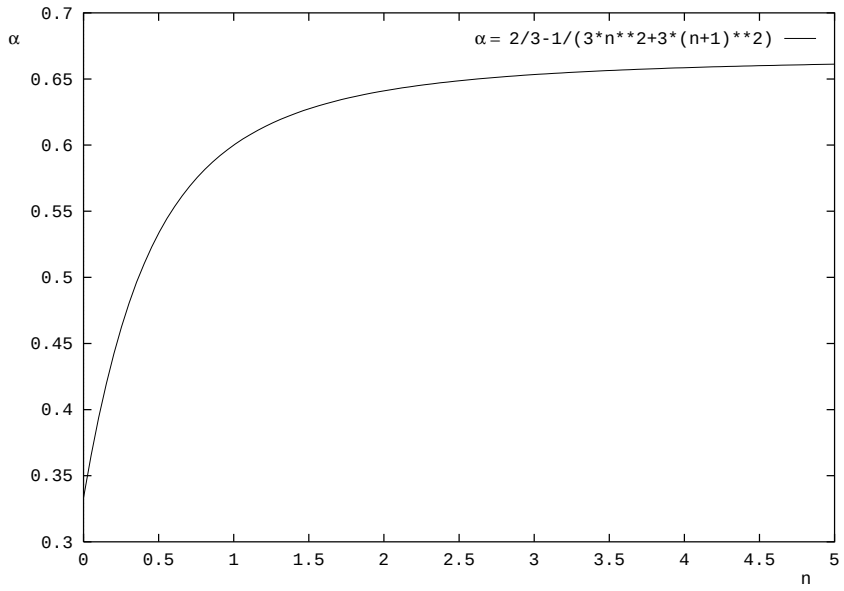

Figure 1.3: *The dependence of α on n in one dimensional maps with one dimensional input.*

One interesting point in this result was the observation that the density function strongly depends on the number of neighbors n if $n \in [0, 1, 2]$ (see Fig. 1.3). In normal practice the neighborhood radius is slowly decreased during the adaptation, and now one can conclude

that if the map has converged to a neighborhood of $n = 1$, then when the value of n is further decreased the convergence points of the weight values are radically changed and the adaptation requires (substantially) more time to reach the equilibrium values again.

The probability density function of codebook vectors was studied also in [Luttrell 1991]. The approach was derived from the vector quantization studies described above (see section 1.5.1). The density of units was studied for one dimensional maps with one dimensional input data.

The studies in [Luttrell 1991] were related to similar considerations in [Ritter 1989], a finite number of units were used and the neighborhood size was fixed. The most important difference was that Luttrell did not use the original Map algorithm at all but selected the best matching unit using the expected quantization error in the coding/decoding system (see above in chapter 1.5.1). Therefore the unit was not selected by the nearest neighbor rule but by a rule where a weighted average of the unit values were computed.

The resulting mapping from the input space to unit space was shown to be independent of the number n of neighbors

$$M(m) \propto P(m)^{1/3}, \tag{1.41}$$

which was the same as the classic result for vector quantization (without any neighborhood effects, see [Zador 1982]). This result differs radically from equation 1.39.

Rate of convergence

The rate of convergence of unit weight vectors as a function of the form of neighborhood function has been studied in two papers, in [Lo et al. 1991] and [Erwin et al. 1992b]. In the first one the convergence speed was studied by simulations with two different neighborhood functions. The studies were more qualitative than quantitative showing some interesting effects. Two different neighborhood functions were used: the original n-nearest neighbors

$$h(i,j) = \begin{cases} 1 & \text{if } |i-j| \leq n, \\ 0 & \text{otherwise,} \end{cases} \tag{1.42}$$

where i, j are unit indexes, and a Gaussian type function

$$h(i,j) = \begin{cases} a + be^{(-\frac{c(i-j)^2}{2n})}, & \text{if } |i-j| \leq n, \\ 0 & \text{otherwise,} \end{cases} \tag{1.43}$$

where a, b and c are suitably selected constants. The observed learning speeds were taken as sufficient proof to use the Gaussian function because of faster training observed in the (few) simulations.

In [Erwin et al. 1992b] the existence of 'metastable' states of the (one dimensional) map during training was shown. These states were such that the expected corrections of the weights due to training algorithm vanished despite the fact that the map was not yet ordered. It was shown that the existence of the metastable states depended on the *shape* of neighborhood function.

Metastable states (local minima)

In [Erwin et al. 1992b] a set of equations was derived to obtain the average change of the weight values in the SOM algorithm for each input sample. Using the equations with a few special types of neighborhood functions it was shown that in some cases the map could be 'stuck' in a local minimum in the weight space (non-ordered states), even though the map eventually frees itself from the minimum (and becomes ordered). Time required for the training depends heavily on the existence/nonexistence of these local minima. The paper also includes some proofs of the existence of these metastable states depending on the neighborhood function.

In the proofs, *convex* and *concave* neighborhood functions were defined. A neighborhood function was said to be convex on a certain interval $I \equiv \{0, 1, 2, \ldots, N\}$ if

$$|s - q| > |s - r|, |r - q| \Rightarrow [h(s, s) + h(s, q)] < [h(s, r) + h(r, q)], \tag{1.44}$$

holds for all $s, r, q \in I$. A neighborhood function was said to be concave otherwise.

The main result is that if the neighborhood function is convex, then there are no other stable states than the ordered ones. If the neighborhood function is concave, there exist metastable states that slow down the ordering process by orders of magnitude. The theoretical results were demonstrated with some simulation runs. The results demonstrated the observed phenomenon that the map was most effectively trained using first a broad neighborhood function that ensures that no metastable states were present and then (after ordering) contracting the radius of neighborhood function to allow more precise mapping.

1.6 Other Analysis Papers of Self-Organizing Process

In [Ishida et al. 1992] the effect of quantizing the weight values into a small number of levels was studied. It was shown by simulation runs (using the Traveling Salesman Problem as a test problem) that the weight quantization does not affect the accuracy provided that there are reasonably many levels; in these experiments about 20 levels were sufficient.

Another problem studied in [Ishida et al. 1992] was the case where the successive input samples to the SOM were correlated. In the original algorithm it was required that the samples be uncorrelated and in [Ishida et al. 1992] the effect of correlated input patterns was shown to affect the distribution of weight values considerably. The results were based on simulations with very small SOMs, though, where all three (3!) units were always adapted.

In [Critchley 1992] the effect of the form of the neighborhood function (either block neighborhood or Gaussian neighborhood) and the function of decreasing neighborhood size and adaptation gain were studied. A new approach to the problem of when and how to decrease the neighborhood size and adaptation gain was proposed. (In the original algorithm these parameters are decreased by a fixed function of adaptation steps, either linearly or exponentially.) Critchley proposed the following decreasing law:

> The size of neighborhood R and adaptation gain η have fixed values until the weight values have relaxed to stable positions, the indication of which is given by the value of
>
> $$V = \langle \sum_{i=1}^{n} m_i^2 \rangle, \tag{1.45}$$
>
> where m_i is the weight vector of unit i. When the value V has become stable the neighborhood size and the adaptation gain are reduced by
>
> $$R \to \alpha R, \quad \eta \to \alpha \eta, \tag{1.46}$$
>
> where $\alpha < 1$.

It was shown by simulation runs that the Gaussian function with the new dynamic parameter decreasing scheme resulted in a faster fit for the input data distribution than the original block neighborhood.

In [Siemon 1992] some studies on how to determine 'optimal' training parameters for SOMs were described. The objective was to find the optimal values for the neighborhood radius and the adaptation gain to order the SOM in the typical one dimensional situation (a one

dimensional input space and a one dimensional map). The results were based on simulation runs.

In [Varfis et al. 1992] the problem of the sensitivity of the SOM mapping to initialization was considered. An original solution to find 'good' mappings was proposed. The solution was based on training several SOM mappings with different initial values with the same input data and then determining the similarities between the mappings. It was argued that those mappings that clearly form a large group of similar mappings are better than those mappings that differ from all the others. The system was demonstrated with a small experiment of multi dimensional data vectors.

Part II

Analysis of Pattern Sequences by the Self-Organizing Map

Chapter 2

Visualization of Pattern Sequences by Self-Organizing Maps

The Self-Organizing Map algorithm creates a mapping from input patterns to Map units. The mapping can be applied to pattern sequence analysis by finding the mapping locations of subsequent patterns and observing the trajectory, that is the curve of the locations in time. The pattern sequence in multi dimensional input space can in that way be transformed into a two dimensional trajectory which can be used, for example, in process monitoring applications.

In this chapter some applications where Self-Organizing Map algorithm has been used in visualization of speech signal and speech signal variations in time are described. Other applications where the Self-Organizing Map algorithm has been used for process state monitoring are described in the end of the chapter.

2.1 Detection of Dysphonic Voices

The quality of a vowel sound depends on voice which is determined by phonation and resonance. Normally the repetitive movements of the vocal chords are synchronous and the cycle-to-cycle variation is small, especially in long vowels. Both functional disorders and anatomical deformations may induce irregularities in the glottal sound[1].

Voice disorders of laryngeal origin (dysphonias)[2] are easily recognized by a human listener, while it has been difficult to characterize them using 'traditional' speech signal analysis methods. Consequently it has proven to be difficult to design automatic methods for detection of such voice disorders.

It is typical to train people to describe voice disorders using suitable descriptive words. The description system makes it possible to achieve reliable characterization of voice quality for clinical practice. However, some objective acoustic measures of voice quality are needed for the quantitative evaluation of therapeutic operations. The standardization of voice evaluations would also benefit from an objective measure.

In clinical practice the evaluation of voice quality is performed by groups of expert listeners. The standardization of perceptual judgements is difficult. Training judges takes time and comparable studies would require using the same judges. Small differences in voice quality cannot often be detected by listener tests.

[1] The glottal sound is a sound produced in the opening between the vocal chords.

[2] Larynx is located at the upper end of the human trachea (windpipe), containing the vocal chords and serving as the organ of voice.

Many voice disorders are characterized by time-dependent deviations in the speech waveform, and their detection is thus a problem of classification of the time dependent features. In some cases the normal and abnormal sounds intermingle, while in some other cases the spectral deviations are permanent. Temporal deviations are sometimes observed as continuous deformation of the glottal sound whereas in other instances long periods of normal regular sound are cut by sudden irregularities. It is therefore required that the automatic evaluation system is capable of analyzing the speech signal for hundreds of milliseconds or even several seconds.

Simple spectrographic analysis has been applied to the description of voice quality. The extraction of perceptually meaningful features from the spectrograms has proven to be difficult. The analysis of spectrograms also requires some training, and therefore they cannot easily be used as a tool in voice therapy.

In [Leinonen et al. 1992] an approach to develop an automatic device for voice quality analysis was described. In this paper the Self-Organizing Map was used to visualize the speech signal and the speech signal variations in time, and from the mapping created by the SOM process it was further possible to construct some quantitative measures of the voice quality. In the studies the voice quality of vowels was analyzed.

Dysphonic voices can be characterized by typical variations in the speech waveform. Both irregularities of the glottal origin and deformations of the glottal waveform induce changes in the speech spectrum and these changes can be visualized by the Self-Organizing Map based visualization tool. Spectral changes due to articulation, and the great variability of speech spectrum regarded as normal make the extraction of dysphonic features difficult. The analysis system should find the meaningful variations and discard the meaningless ones in much the same way a human listener does. Analysis on that level is possible only if it is based on the speech signal statistics.

2.1.1 Self-Organizing Map in Visualization of Speech

The Self-Organizing Map algorithm creates a mapping from the speech signal space to a two dimensional plane determined by the processing units of the map. Similar speech sounds are mapped to nearby locations in the map (plane). The mapping thus preserves the similarity of the acoustic signals (to a certain degree). The mapping also takes into account the probability density function of the input samples so that the distribution of inputs to the map units approximates the density function directly.

Perceptually meaningful features of speech derive from complex features in the speech spectra. Such features are represented by Self-Organizing Map as locations on a two dimensional map. Visualization of voice quality with SOM is therefore easily comprehendable when compared with, for example, spectrograms. Because the probability density function of the input samples is also taken into account, those sounds that are more common are represented with more details than less frequently occurring sounds. Thus more emphasis is put on the deviations of more common sounds. It is possible to directly observe the similarities of the sounds by observing the distance between the respective representations on the map.

The Self-Organizing Map method is especially useful for the visualization of the changing of speech spectra in time. The representations of consecutive short-time spectra form a trajectory on the map and changes in time can be observed from the representations. The abrupt changes in the speech signal observed in some dysphonic patients are depicted by the trajectory curve. Some examples of speech trajectories on self-organized maps are shown in Figures 2.1 and 2.2.

The maps in Figures 2.1 and 2.2 were used to analyze the voice during long /a:/ vowels. We measured the smoothness and regularity of the speech sound by computing diagnostic figures from the trajectory. For the evaluation of the stability of the samples, the following

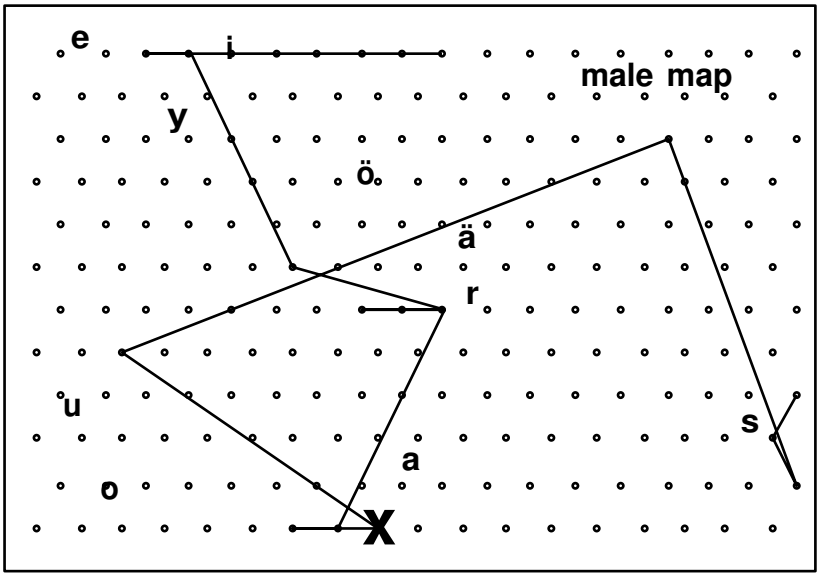

Figure 2.1: *A self-organized map computed with Finnish speech samples from 15 men. The map areas for the vowels, /s/ and /r/ are shown on the map. The trajectory produced by an utterance of /sa:ri/ is indicated by the line beginning from /s/ area and ending in /i/ area.*

parameters were calculated:

1. The total length of the trajectory during an interval of 150 ms,
2. the mean length of shifts, and
3. the number of cases where consecutive samples have the same representations in the map.

The length of trajectory indicates large scale changes in consecutive samples, and the number of consecutive samples having the same representations gives an indication of a small scale stability.

The permanent changes in the speech spectra (as opposed to cycle-to-cycle variations) can also be observed from the trajectory. In these cases, the trajectory is smooth without any abrupt 'jumps' in it but it is located on a 'wrong' area on the map. When normal speakers utter a known phoneme, we expect the samples to be projected into a specific area. Normal interspeaker differences are seen as small differences in locations within this area. By determining a sample trajectory in reference to this 'normal area', we can thus diagnose the 'normality' of the speech sound.

2.1.2 Speech Data Used

Speech data for the experiments (and for the Self-Organizing Map training) were collected from people that were treated for phonation disorders and from people that were not treated for any such problems.

There were 58 subjects who were not treated for voice problems (25 men and 33 women). The dysphonic speech samples were obtained from 24 patients (6 men and 18 women). The ages of he patients varied from 18 to 55 years (the mean age being 38 years for men and 41 years for women). The subjects were diagnosed as follows: there were 10 hyperfunctional

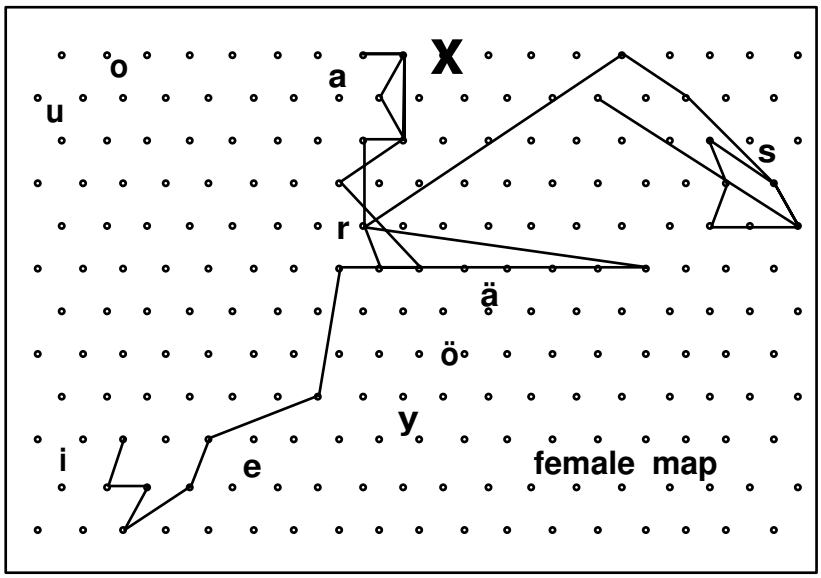

Figure 2.2: *A self-organized map computed with Finnish speech samples of 18 women.*

cases, 7 hypofunctional cases, 2 Reinke's edema cases, 3 unilateral cases, 1 granuloma case and 1 mutational case ([Leinonen et al. 1992]).

The analysis of the dysphonic samples was done using the long /a:/ vowels occurring in Finnish words /sa:da/ or /sa:ri/. Both of the words are common Finnish words.

The test words were evaluated independently by two speech-language pathologists. The test words were embedded into a list of words and the pathologists judged the speech quality using the same criteria as in their everyday practice. The voice quality was evaluated with a scale $0-3$ along four dimensions: the degree of pathology, roughness, breathiness, and strain.

Based on the perceptual judgments the speech samples were divided into four classes:

a) samples that both pathologists judged normal (18 subjects, 7 men and 11 women)).

b) samples that both pathologists judged mildly dysphonic (23 subjects, 7 men and 16 women).

c) samples that both pathologists judged severely dysphonic (17 subjects, 6 men and 11 women).

d) samples that one pathologist judged normal and another judged mildly dysphonic (mixed, 24 subjects, 11 men and 13 women).

2.1.3 Acoustic Maps

The Self-Organizing Maps used in the experiments were computed using speech data from subjects whose voices were judged normal. Two acoustic maps were trained, one with 200-word samples of 18 women and another with 200-word samples of 15 men. The resulting maps are as in Figures 2.1 and 2.2.

The speech signal was preprocessed using the routines developed for a speech recognition device [Kohonen 1988b] (see 3.2). The signal was collected at a sampling rate of 13.02 kHz. Short-time power spectra were calculated every 9.83 ms using 256-point FFT. The logarithms of the power spectra were smoothed in the frequency dimension by low-pass filtering and the feature vector was formed from 15 amplitude values between 50 Hz and 5 kHz. The resulting 15 dimensional vectors were fed as the input to the map.

2.1.4 Results

In this section the main results reported in [Leinonen et al. 1992] are described. Some of the preliminary results were already reported in [Leinonen et al. 1991].

Visual evaluation of the samples on the self-organized map already showed that there were easily detectable differences between the normal and dysphonic samples. Some clearly differing cases have been collected to Figures 2.3 and 2.4. The data points of the normal voice samples are projected into one or a few nearby locations, whereas the trajectories of the dysphonic voice samples are longer with long jumps between the projections of the data points. The spectral stability was characterized by the mean length of shifts: the length of the trajectory during 150 ms was divided by the number of samples (the distance between two neighboring map locations was one).

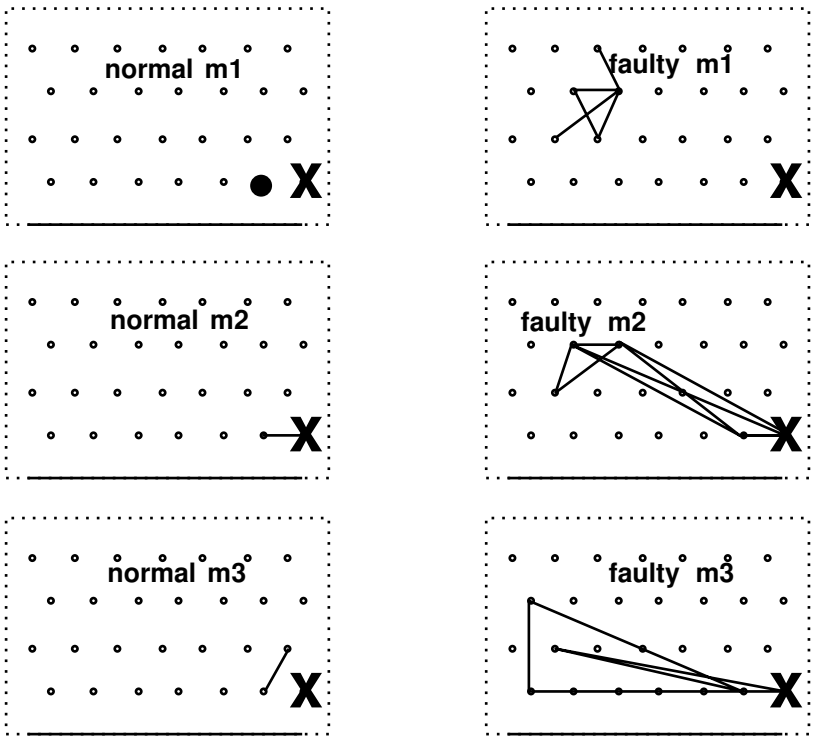

Figure 2.3: *The map trajectories of 150 ms speech samples of /a:/ uttered by 6 men (only the relevant part of the map is shown). The normal voices (normal m1-m3) generate localized, short trajectories, while the dysphonic voices (faulty m1-m3) generate long jumpy trajectories. The 15 consecutive data points of normal m1 are all projected to the same location, indicated by the large dot.*

The mean lengths of the shifts for both the normal and dysphonic voices are shown in Tables 2.1 and 2.2. The mean lengths of shifts of normal voices differed significantly from the mean lengths of shifts of both for mildly dysphonic and severely dysphonic voices both in men and women (Kruskal-Wallis one-way analysis of variance and Tukey test [Zar 1984]).

Three repetitions of the test words were collected from a few of the subjects. In these cases the stability of the voice (of healthy and dysphonic voices) could be checked by repeated trajectory generation tests. The repeated samples of subjects with healthy voices were similar: the mean lengths of shifts were low and the variation was small. In the dysphonic samples the mean length of shifts were higher and the difference between the repeated utterances were larger than for the samples of healthy voices.

As judged by the mean length of shifts, most of the dysphonic voices were unstable compared

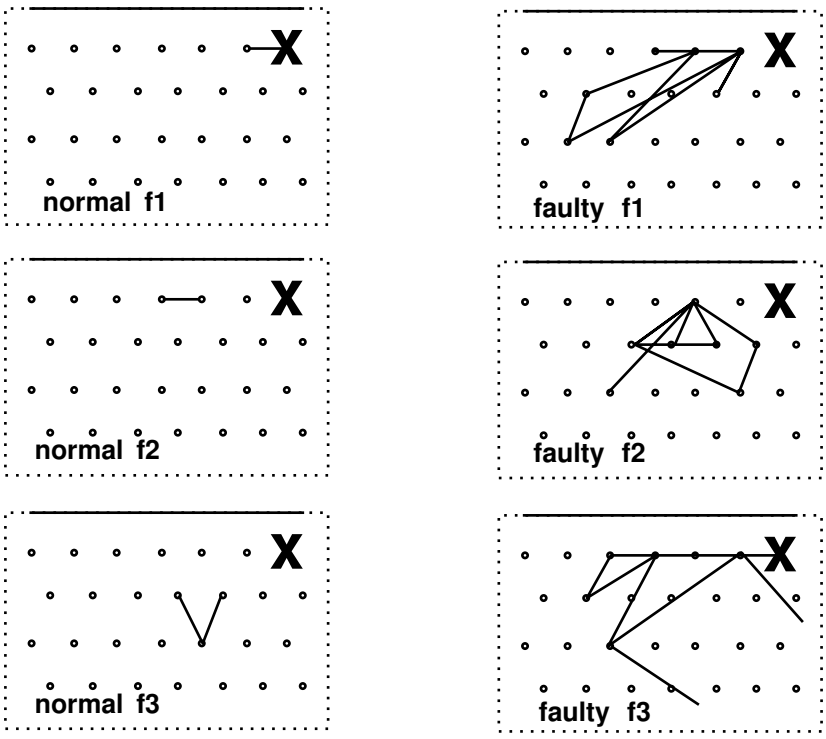

Figure 2.4: *The map trajectories of 150 ms speech samples of /a:/ uttered by 6 women. The normal voices (normal f1-f3) clearly differ from the dysphonic voices (faulty f1-f3). The trajectory of faulty f3 even deviates momentarily outside the /a/ area to the area of silence.*

Voice quality	Number of subjects	Range	Mean length of shifts
Normal	11	0.00 - 0.69	0.35
Mild dysphonia	16	0.13 - 2.44	0.93
Severe dysphonia	11	0.40 - 3.62	1.31

Table 2.1: *The mean lengths for shifts in female samples of long /a:/ judged as normal or dysphonic by two speech pathologists.*

to the normal ones. Some dysphonic voices, however, did not differ from the normal voices with respect to the speech stability but only with respect to the spectral composition. The sample trajectories were located, at least partly, outside the map area determined by the projections of the healthy samples. Figure 2.5 shows the distribution of normal and breathy samples, and the data points of 'rough' samples which were outside the area.

Some examples of the location shifts can be seen in Figure 2.6. In these cases the voices were diagnosed as severely dysphonic.

2.1.5 Discussion

The Self-Organizing Map method described above made it possible to easily detect time-dependent changes in speech sound composition and detect features characteristic of dysphonic voices. The measurement system differentiated between normal and dysphonic voices in a statistically significant way.

Voice quality	Number of subjects	Range	Mean length of shifts
Normal	7	0.00 - 0.60	0.28
Mild dysphonia	7	0.73 - 2.72	0.90
Severe dysphonia	6	1.26 - 2.75	1.94

Table 2.2: *The mean lengths for shifts in male samples of long /a:/ judged as normal or dysphonic by two speech pathologists.*

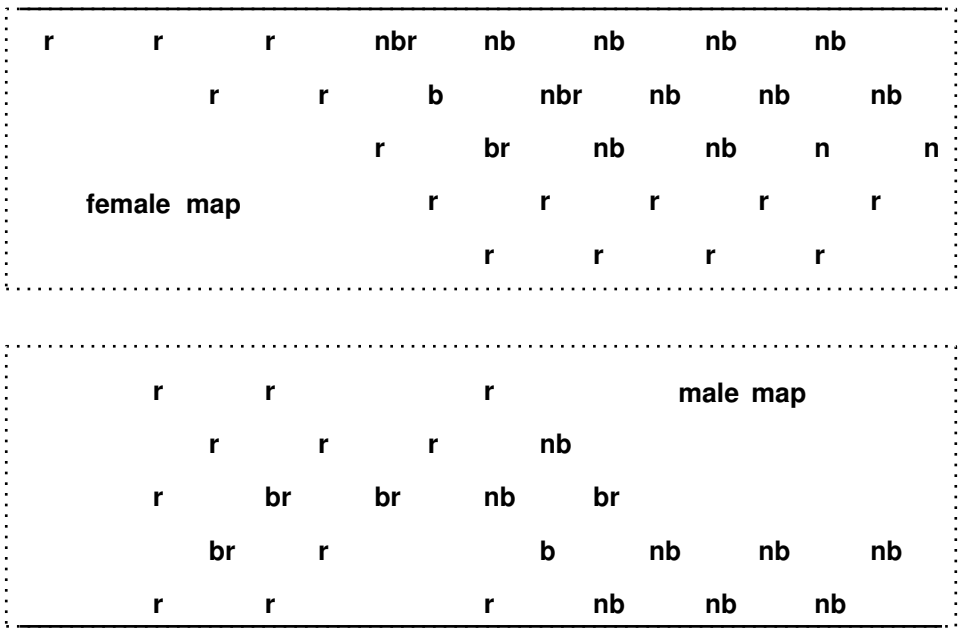

Figure 2.5: *Distribution of the sample projection points of different voice qualities. All normal samples project to locations marked 'n' and breathy samples to 'b'. Samples judged rough ('r') included data points that were projected outside the area of normal and breathy samples.*

Although the method provides quantitative measures of the voice quality, in many applications main advantage is the on-line visualization of speech. The graphic representation of speech can be obtained in real-time once the self-organized maps are trained. The graphic display of speech is easily understood by subjects who have no previous knowledge on the acoustic speech analysis methods. Thus the system can be used as a feedback device in voice therapy. The training of speech pathologists could also take advantage of the graphic representation of speech.

Real-Time Implementation

In [Utela et al. 1992] a microcomputer implementation of the speech signal analysis and visualization [Leinonen et al. 1991] is described. The system was implemented using a portable PC/AT microcomputer containing a signal processing card for acoustic preprocessing and Self-Organizing Map computing. This real-time system is intended to be used for demonstrations. The implemented system will also be used for studying the clinical applicability of the analysis method.

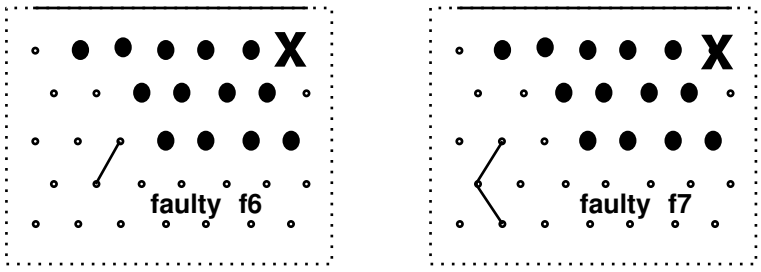

Figure 2.6: *Map trajectories of severely dysphonic samples of long /a:/ uttered by two women. The trajectories are outside the area of normal voice samples (large dots). The mean lengths of shifts (0.40 and 0.73) were similar to those of normal voice samples.*

2.2 Analyses on Articulation

2.2.1 Recognition of /s/ Misarticulation

In [Mujunen et al. 1993] another application of the Self-Organizing Map algorithm for speech signal analysis is described. In this paper the self-organized map was used to distinguish between the /s/ samples perceptually classified as acceptable/unacceptable, as judged by 21 speech pathologists. It was shown that the map was a suitable tool for the extraction and measurement of acoustic features corresponding to the audible deviations of /s/.

Speech Data

The speech data consisted of 125 words collected from 11 women aged 16-38 years. All women had histories of /s/ misarticulation. The subjects uttered a word list including the test words /sa:s/, /si:s/, /so:s/, and /su:s/. The word initial /s/ samples were acoustically analyzed and perceptually judged as 'acceptable' or 'unacceptable' by a group of speech pathologists.

Fifteen dimensional feature vectors (the amplitude values of different frequencies) were computed from the digitized data as described in 2.1.

The self-organized map was computed using 200-word speech data from 18 women, without co-articulation disorders. Figure 2.7 shows the map and the trajectory of the test word /si:s/ uttered by a woman whose /s/ was judged as 'acceptable'. The map area representing the Finnish vowels and /s/ fricative are indicated by the letters over the map.

Measurements

The map patterns of each /s/ (after exclusion of two first sample points) were characterized using the following measures:

1. The number of data points projected into the /s/ area on self-organized map.
2. The length of the /s/ trajectory and the mean length of shifts (indicating the spectral stability).
3. The average location of the /s/ samples in the $x - y$ coordinates of the map.

Results

The /s/ samples inside the acceptable area were compared to the /s/ samples outside the acceptable area with respect to the 'unacceptable' judgements and they differed in a statistically significant way ($p < 0.002$, Mann-Whitney U test, two-tailed [Siegel 1956]). The acceptable area on the map was defined on the basis of data of previous studies.

The mean length of shifts or the durations of the fricatives did not help to differentiate between the acceptable and unacceptable samples.

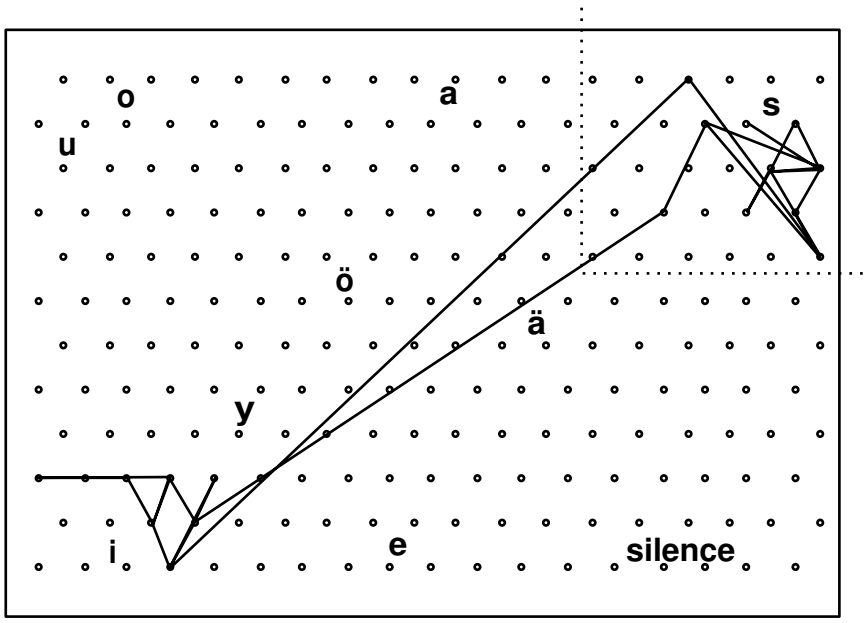

Figure 2.7: *The test word /si:s/ uttered by a woman on a self-organized map. The map areas for the Finnish vowels and /s/ fricative are by the letters on the map.*

Discussion

The Self-Organizing Map representation offered a method to visually and quantitatively differentiate between psychoacoustically acceptable and unacceptable /s/ samples. The location changes are due to the differences in the short time spectra as computed from the speech samples. The unacceptable /s/ sounds differ from the normal ones, and therefore they are projected to different units in the SOM plane.

The continuity in the SOM model vector distribution (the nearby model vectors are more similar than the model vectors located farther) gives the user an intuitive tool to visualize the differences and common features inherent in different speech samples. The user has an immediate visual feedback of the pronunciations to the graphic display where the location of best responding units indicate the quality of the speech sound, quite effectively. The visual feedback tool could obviously be used, for example, in voice therapy for people who have difficulties hearing the differencies between normal and deviant /s/ sounds and for deaf people learning to speak or improve their speech.

These results were in accordance with another study where the Self-Organizing Map was used in the analysis of speech disorders [Räsänen et al. 1990]. The quality of /r/ and /s/ sounds of one patient was studied by SOM mapping both before and after a session of voice therapy.

2.2.2 Detection of Co-articulation Effects

From acoustic voice analysis using spectrograms it can be observed that the vowel following a word-initial /s/ clearly affects the spectra towards the end of the /s/ sound. Thus the following vowel type could be recognized based on the changes in spectra due to the preparatory movements of lips and tongue for the following vowel.

In [Leinonen et al. 1993b] and [Leinonen et al. 1993a] the phenomenon was further studied to find out if the Self-Organizing Map algorithm could extract useful, perceptually meaningful features in the /s/ sound. The objective was to see if the representations of the /s/ samples before different vowels could consistently be recognized from the trajectories produced by

projecting the consecutive speech samples into a map plane.

Speech Data

There were slight differences in the speech data and the signal processing methods between [Leinonen et al. 1993b] and [Leinonen et al. 1993a]. In the following the data and the preprocessing methods used in the first paper are described.

The speech samples were collected from 10 women which all had normal /s/ sound (as judged by a speech pathologist and an inexperienced listener). The subjects repeated a list of seven words three times (words: /sa:ri/, /si:s/, /su:si/, /se:s/, /so:si/, /sy:s/, /sä:si/).

The signal preprocessing was done as in the experiments of section 2.2.1. Fifteen dimensional feature vectors (short time power spectra) were computed every 10 ms from the digitized data.

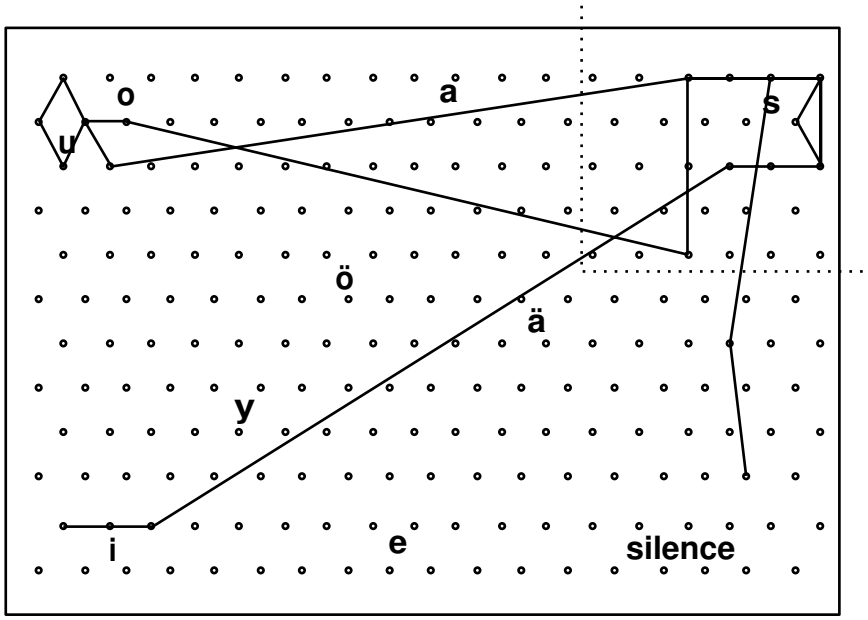

Figure 2.8: *The Self-Organized Map trained with speech samples from 18 women judged not to have any voice problems. The /s/ area is on the right upper corner of the map.*

The self-organized map was trained using 200-word speech samples from 18 women judged not to have any voice or articulation disorders, Figure 2.8.

The following measures were computed for each /s/ sample:

1. Number of samples in the /s/ area.
2. Length of the trajectory during that time (indicating the spectral stability). The mean length of shifts.
3. The average location of the /s/ samples in the $x - y$ coordinates of the map.

Results

The visual evaluation of the results revealed that the map locations of the /s/ samples before the rounded (/o/ and /u/) and unrounded (/a/, /i/, /e/, and /ä/) vowels were clearly different. One example of the effect of co-articulatory lip-rounding can be seen in Figure 2.9. The average locations of /s/ samples in front of the rounded vowels /o/ and /u/ were above and somewhat to the left from the /s/ samples in front of the unrounded vowels /a/, /i/,

/e/, and /ä/. The locations of /s/ samples before /y/ could not be distinguished from the locations of /s/ in front of unrounded vowels.

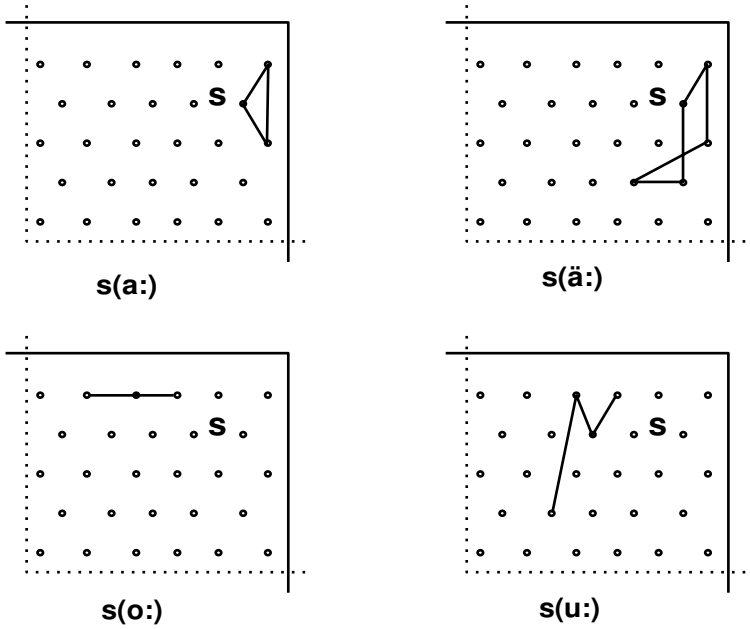

Figure 2.9: *The mean locations of /s/ fricative followed by vowels /a/, /ä/, /o/, or /u/ are located on different area. In the figure only the /s/ area of the map is shown. The four samples are from the same female speaker.*

Discussion

The Self-Organizing Map representation of the speech sound was shown to differentiate consistently between the spectral samples of /s/ sounds before either rounded (/o/, /u/) or unrounded (/a/, /i/, /e/, /ä/) vowels. The SOM visualization was thus proven to be a rather sensitive method for detecting the co-articulatory variations of /s/ fricative.

2.3 Process Monitoring and Control by Self-Organizing Maps

The analysis and control of complex nonlinear processes is one of the problem areas where neural networks are potentially useful, because of their ability to adapt. The monitoring systems could adapt to respond to the characteristic features of the observed process, without any specific prior knowledge of the application area. In the following some applications are explained where the Self-Organizing Map approach has been used for process state monitoring and control.

2.3.1 Process State Monitoring

In process state monitoring applications the problem is to visualize the typically complex relations between system states in an efficient and understandable way. The online process status measurements should be converted to some simple displays that, despite the dimensionality reduction, preserve the relationships between states as well as possible for the user to follow the process state development.

The Self-Organizing Map method makes a mapping from a multi dimensional space to two dimensional surface of the processing units. The mapping is, furthermore, done in such a way

that the topological relations between the input state vectors is preserved. Thus the Self-Organizing Map algorithm might easily be applied to process state monitoring applications.

In [Tryba et al. 1991] the applicability of the SOM algorithm to process state monitoring and control was explained. The SOM method was explained in sufficient detail and the benefits of the algorithm as applied to monitoring applications were pointed out. The example application was a distillation process. In [Sorsa et al. 1993] another study of using neural networks in fault diagnosis of a chemical process was explained.

In [Alander et al. 1991] another preliminary study of the potential of the SOM algorithm for process monitoring was described. Self-Organizing Map was used to detect abnormal states in a real-time process by examining the quantization error between the best matching unit in the Map and the input vector. The detection was based on the fact that the map units were distributed in the space occupied by those input samples that were encountered during the training. Rising quantization error obviously indicated some process state that had seldom occurred during the training period. A similar principle of error detection was used also in [Kasslin et al. 1992] where error states were detected also by observing the projection of samples to 'forbidden' areas in to Map, determined using a known set of error samples.

In [Cumming 1993] two neural network models (Self-Organizing Map and a kind of autoassociative back-propagation model) were applied to a monitoring application of an engine condition. The results were promising. The authors found the SOM to be especially useful in the visualization of data properties and highlighting the deviant data values. In [Ultsch 1993] a U-matrix method for the visualization of the process properties through the map was explained (the U-matrix method was introduced in [Ultsch et al. 1989], [Ultsch et al. 1990]).

In [Martín-del-Brío et al. 1993] the Self-Organizing Map was used to analyze some financial data from a group of Spanish banks. It was shown how different regions in the Map represented solvent and bankrupt banks, respectively. It was also possible to follow the time evolution of the banks from the trajectory created by mapping the data of successive years.

Continuous Process Control

The system monitoring applications have sometimes been extended to control systems where the actions are based on the SOM output.

In [Tryba et al. 1991] the principles of how to control processes by the SOM monitoring systems is described. The simplest method given in that paper was to associate the control actions to each unit and to activate them whenever the unit was selected by the input sample. In [Ritter et al. 1989a] and in greater detail in [Martinetz et al. 1990] the SOM algorithm was applied in a robot arm control system. The principle was the same as above, the control actions were associated with the map units each representing a possible system state requiring some action. In [Walter et al. 1991] the robot control system was slightly modified to apply a modified SOM algorithm. In [Meyering et al. 1992] the mapping algorithm developed for control actions was used in reverse order to recognize hand postures (to be used as command interface for robot arm controller). In [Hyötyniemi 1993] a similar control application was explained where the dynamic system to be controlled was known beforehand.

In [Govekar et al. 1992] the input data consisted of two parts: the measurement data and a descriptive part of the process parameters during the measurement. The SOM was used to store the input data distributions. The trained system was later used for process parameter estimation from the measurement samples by using the SOM to find the best matching unit by the measurement vector part only, and looking for the descriptive part of the selected unit. The experiments with the real system data confirmed that the process parameters could be detected and estimated by the SOM system quite reliably. The system, furthermore, made only small recognition errors in the sense that even if the parameters were incorrectly recognized, the correct parameters were usually the most similar ones.

Chapter 3

Utilizing Contextual Information with the Self-Organizing Map

Soon after the introduction of the Self-Organizing Map algorithm [Kohonen 1981], the self-organizing principle was demonstrated with natural speech signal [Kohonen 1982b]. It was shown that the Self-Organizing Map creates an accurate representation of the spectral relations between speech samples. The extension of the system from the visualization of spectral features to a system capable of pattern recognition of speech samples was straightforward [Kohonen et al. 1984]. Since then, the work on speech recognition has continued. The goal has been to devise a real-time speech recognition system where the basic recognition of short time speech samples is based on the Self-Organizing Map.

The Self-Organizing Map algorithm used in the speech recognition system presented in [Kohonen et al. 1984] was a special *supervised* SOM. Later the Self-Organizing Map algorithm was replaced by the Learning Vector Quantization (LVQ) algorithm, although the SOM algorithm yielded nearly as good recognition results ([Torkkola et al. 1995]).

The first speech recognition devices using the Self-Organizing Map analyzed static signal patterns, one sample at a time. The temporal variations of the speech signal were not taken into account until after recognition of the (quasi)phoneme was done. The following subsections describe some attempts to increase the analysis power of the SOM model by suitably manipulating the pattern sequences. Most of the models are tested with real speech signals for speech recognition. Some other works analyzing pattern sequences are described at the end of the chapter.

3.1 Neural Networks and Analysis of Pattern Sequences

Extensive research has been done into the analysis of pattern sequences and consequently the properties of various 'traditional' analysis models are well known. However, there are several reasons that explain the interest in new artificial neural network models. Firstly, the construction of a pattern recognizer, for example, often requires a lot of expertise on the subject because the model based systems are not applicable without applying a rather strict correspondence between the studied process and the constructed model. Neural networks offer robust model-free methods without requiring as much expertise.

Secondly, the artificial neural network models can be used in an adaptive mode, that is, they learn while in use. This offers a natural way, for example, to compensate for the wear of measuring devices. Thirdly, the neural network models are naturally parallel systems, offering more speed in computation (if suitable hardware implementations are used) and fault tolerance compared to traditional computing models.

A lot of work has been done on applying various artificial neural networks models to the analysis of sequential data structures. The basic models are not directly applicable to the task, since they usually analyze static samples. Fortunately, the neural models can be easily modified for the analysis of some sequences.

In the following the two most often cited temporal neural network models are described. The first one, Time-Delay Neural Network (TDNN), is based on the idea of transforming input sequence into a spatial pattern before processing, and the second, recurrent error back-propagation, models the pattern sequence using a state machine (see e.g. [Mozer 1993]).

3.1.1 Time-Delay Neural Network

One example of the delay line approach is the Time Delay Neural Network (TDNN) (see, for example [Waibel et al. 1989] and [Lang et al. 1990], also [Sánchez-Sinencio et al. 1992]) which has been used extensively in speech recognition experiments to utilize the context knowledge available in a sequence of speech samples (see Fig. 3.1). In the TDNN model, the speech signal is classified to phonemic labels using a modified error back-propagation model extended with vector concatenation. The speech recognition is completed by applying Hidden Markov-models to the phonemic labels.

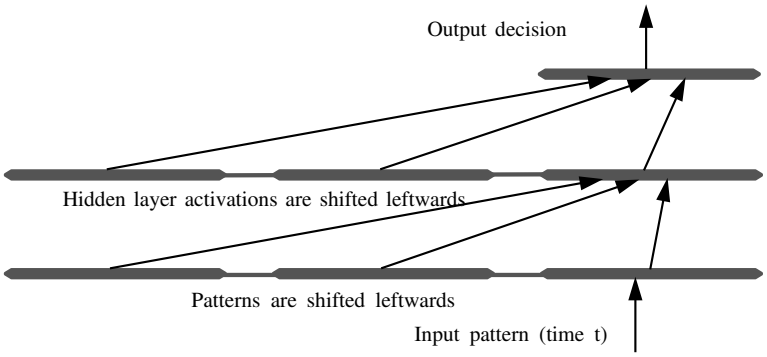

Figure 3.1: *The structure of the TDNN phonemic classifier. The most recent pattern vector is read into the rightmost slot in the bottom layer and the stored samples are shifted to the left. All stored samples are used in the computation of the middle layer activations and the old activations are again shifted to the left. The decision is read from the top level. The number of storage slots in each layer (here three are shown) can be selected quite arbitrarily based on the problem area and complexity.*

The structure of TDNN is shown in Figure 3.1. The newest pattern vector is read into the rightmost slot in the bottom layer of the model, while the previously read samples are shifted leftward. All the pattern vectors on the bottom layer are then used to compute the 'activations' of the middle layer. The middle layer vector is similarly constructed and then used in the computation of the upper level 'activation'. The classification is based on the values on the highest level only. Because of the shifting in time, the system takes into account the time structure of the signal. The training of the TDNN is performed using presegmented speech data and any 'standard' error back-propagation algorithm.

The TDNN model experiments described in [Waibel et al. 1989] gave good results. The experiments were limited, however, consisting of the recognition of voiced plosives of three speakers. Moreover, the utterances were isolated manually, so the recognizer had no mechanism for phoneme segmentation. The model is computationally intensive, because the pattern

vectors become long and the training times increase when longer time spans are used.

A slightly different time delay neural network system for speech recognition (single speaker, digits from zero to nine) is described in [Unnikrishnan et al. 1991]. In this work the delay lines were used to collect the features (amplitude values from several frequency channels) during the utterance. In a later paper [Unnikrishnan et al. 1992], the recognition system was generalized to a multispeaker system.

3.1.2 Recurrent Error Back-Propagation

A state machine model can be implemented using a modified error back-propagation neural network with a feedback loop (see Fig. 3.2). The internal activations of the nodes are stored and act as short time memories. The models do not differ greatly from the original error back-propagation models in the layout but the feedback loop makes the design of efficient learning algorithms more difficult. Some efficient learning algorithms have been proposed, for example in [Almeida 1987], [Pearlmutter 1989], [Pineda 1989] and [Williams et al. 1990].

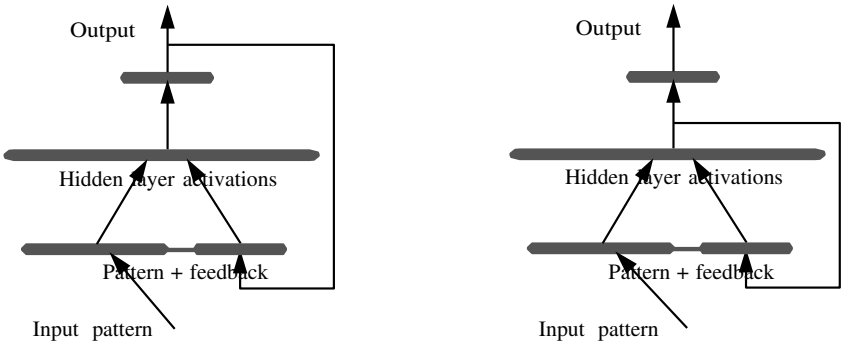

Figure 3.2: *Different recurrent error back-propagation neural network models that have been used in the analysis of sequential signals. In both models the input pattern is concatenated with a feedback pattern vector.*

The recurrent error back-propagation models keep parametric representations of their previous outputs or hidden level activations in the feedback connections. Thus the previous samples indirectly affect the future computations. In principle feedback loops keep an infinitely long sequence of signal values in memory in a compressed form.

3.1.3 Other Models

Some researchers treat the neural implementations of the Hidden Markov models as original artificial neural network systems (e.g. [Kung 1993]). These models (combined with the efficient learning algorithms of regular Hidden Markov models) implement a state machine for sequence processing in a rather compact form.

In some applications a useful representation of a pattern sequence can be obtained by averaging input patterns in time. Such a model is proposed in [Chappell et al. 1993]. The input patterns were averaged using suitable weights such that the most recent sample was given the largest weight.

Self-Organizing Maps

The Self-Organizing Map algorithm has not been used as often in the analysis of sequential signals as the various error back-propagation models which are more easily modified for se-

quence analysis. The Self-Organizing Map algorithm, however, has some advantages over the other artificial neural network models. For example, the SOM is an unsupervised learning algorithm while most of the other models require that samples are preclassified for the supervised learning. Thus a far greater amount of input data is readily available for training of the SOM. The Self-Organizing Map algorithm also creates an ordered mapping of the input pattern space in the neural layer, making it possible to use the mapping as a tool for sequence monitoring (as described in chapter 2).

In the following sections we describe some applications in which the Self-Organizing Map algorithm has been used in pattern sequence analysis. These applications are intended to demonstrate the applicability of the Self-Organizing Map algorithm to different needs, for example, in speech analysis.

3.2 Speech Recognition

In 1984 the Self-Organizing Map principle was applied to the visualization of speech signal and to the recognition of phonemic units [Kohonen et al. 1984]. The work into speech recognition based on Self-Organizing Maps continued and in [Kohonen 1988b] real-time speech recognition device developed at the Helsinki University of Technology is described in detail.

The Phonetic Typewriter is a real-time speech recognizer for Finnish language [Kohonen 1988b] [Kohonen et al. 1988]. The speech recognition system is a speaker dependent (the system can easily be adapted to a new speaker) phoneme detection based system where the words are collected from the recognized phonemes and no vocabulary is used in the base system.

The speech recognition system is functionally divided into two parts. The first part is the acoustic processor module that transcribes the speech signal into phonemes and the second part is a post processing module that collects the phonemes into higher level entities like words (see Fig 3.3).

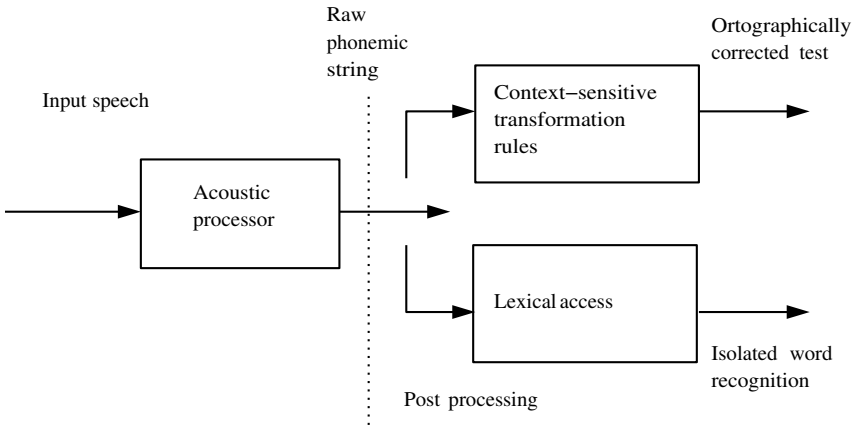

Figure 3.3: *System configuration of the Phonetic Typewriter. The acoustic processor finds the phonemic representation of the speech. The postprocessor corrects some phoneme errors using contextual knowledge.*

Only the acoustic processor module is described here. It is further divided into three parts (see Fig. 3.4 and the subsections below). The first part, preprocessing of the acoustic signal, produces a stream of feature vectors. The second part classifies and labels the feature vectors into phonetic classes using a Self-Organizing Map trained using speech samples. The phoneme

labels are then collected into a quasiphoneme string. The third part of the acoustic module segments the quasiphoneme string (sequence of phonemic labels) into 'true' phonemes.

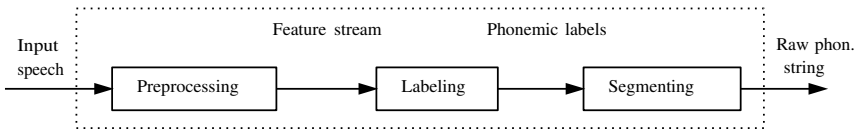

Figure 3.4: *The acoustic processor module. The acoustic processor preprocesses the signal to produce separate samples, labels each sample into the corresponding phonetic class and segments the phonemic string thus collecting 'true' phonemes.*

3.2.1 Preprocessing of the Speech Signal

Preprocessing of the speech signal was done with a rather conventional frequency analysis technique using the fast Fourier transformation and grouping of the power spectra channels into a small number of feature vector components. The selection of preprocessing methods was made for the efficiency of computations.

The preprocessing of the speech signal was done as follows:

1. The signal from a microphone was preamplified and filtered with a 5.3 kHz low-pass filter. The speech signal was collected using a 12-bit A/D converter with a 13.02 kHz sampling rate.
2. A 256-point fast Fourier transform of Hamming windowed signal was computed every 9.83 ms (there was a 128-point overlap between the successive windows).
3. The logarithm of the power spectra was computed and the result was filtered by a low-pass filter.
4. Spectral channels were grouped into 15 ranges to obtain a real valued pattern vector.
5. The average was subtracted from all vector components and the resulting vector was normalized.

The preprocessing results were thus real valued normalized 15-component vectors produced every 9.83 ms where each component represented the energy of the speech signal on a range of frequencies. The lowest frequency component was located at about 0.20 kHz and the highest frequency component was located at about 4.65 kHz.

The overall energy of the speech signal was provided as an extra component of the feature vector. It was not used, however, except in the detection of the beginning and ending of the words by comparing the amplitude of the signal to the background noise.

3.2.2 Labeling by Phonotopic Map Method

The sample vectors are next labeled. In principle it is done by dividing the input space, the space of possible input vectors, into disjoint classification regions, each of which would represent one phoneme, and then giving the label to each sample depending on which region the sample falls. In practice, however, it is often not possible to divide the input space such that no misclassifications occur. Furthermore, locating the optimal border locations (minimizing the number of misclassifications) directly from the speech samples is often rather difficult.

In Phonetic Typewriter the Self-Organizing Map algorithm was used to find the division of input space. When the SOM is trained with a great number of feature vectors computed from the speech signal, units on the Map tend to resemble feature vectors. The distribution of the features in the Map depends on the distribution of input pattern vectors. The representation of the feature vectors on the Map can be used in classification. If the Map units are augmented with labels such that each unit in the Map is associated with the (phoneme) class that it most resembles, then the discrimination borders (of the classes) are defined automatically between the units. During the recognition stage the unknown feature vector is classified to the class with which the corresponding best matching unit in the Map is associated. In Phonetic Typewriter the Map units were labeled with a set of presegmented speech samples where the phonemes are known.

3.2.3 Segmentation of a Quasiphoneme String

The spectral feature vectors are taken every 9.83 ms. Thus we get many quasiphoneme labels for each 'true' phoneme pronounced. When the length of the phonemes in experiments varied from, say, 50 to 200 ms, there were from 5 to 20 quasiphonemes per 'true' phoneme.

To find out the phonemes the quasiphoneme strings have to be 'segmented', the borders between the phonemes have to be located and the phonemes have to be named. In Phonetic Typewriter the segmentation was done by a straightforward (phoneme specific) 'm out of n'-rule, that is, if there were more than m instances of the same phoneme in a substring of n successive quasiphonemes, then a phoneme is identified.

There are other possible algorithms for the segmentation of the quasiphoneme string. For example, the smoothness of the signal could be taken into account by taking the discontinuities in the signal properties as borders between the phonemes.

3.2.4 Later Improvements

The Phonetic Typewriter was based on accurate phoneme recognition. Therefore the recognition stage should be as error free as possible. The errors in the phoneme recognition stage arise from the erroneous labels in the quasiphoneme string and from the errors in string segmenting. In later research work it has been possible to decrease the number of errors in these modules.

The development work of the speech recognition system is described in [Torkkola et al. 1991]. The original architecture of the speech recognizer had been revised in many respects. The main modifications to [Kohonen 1988b] were as follows:

1. The classifier was fine tuned using the Learning Vector Quantizer (LVQ) algorithm ([Kohonen 1988a], [Kohonen 1990]) after the initialization of the classifier by the Self-Organizing Map algorithm.

2. The feature vectors were changed from Fourier spectra to mel-scaled cepstra computed from the spectra (see [Torkkola et al. 1991]).

3. Several cepstral vectors were concatenated to form longer vectors in order to include some time dependent features in the sample.

4. The phonemic segmentation was done using Hidden Markov Models (see [Rabiner 1989], [Juang et al. 1991]).

5. A phonemic (adaptive) grammar based post-processing algorithm (Dynamically Expanding Context, DEC, [Kohonen 1986]) was applied to correct systematic errors in the phonemic strings to produce more correct written forms.

The recognition accuracy was considerably improved by the above mentioned modifications. The error rate dropped more than 50 % of the original combination of the system components.

Recently some other research groups have reported studies where similar system configurations have been used in speech recognition applications. In [Koizumi et al. 1991] and [Zhao 1992] the superficial structure of the recognizer is similar to the Phonetic Typewriter. In [Koizumi et al. 1991] the SOM does not produce phoneme labels but a sequence of map coordinates. However, the coordinates are given to a Hidden Markov model (as was the label string in Phonetic Typewriter) to find the spoken words. In [Zhao 1992] a three dimensional SOM has been used as a vector quantizer for a Hidden Markov model. In [Wu et al. 1992] the trajectory of the spectral vectors on the SOM is used to distinguish between the Chinese phonemes.

3.3 Recognition of Non-Stationary Phonemes by Auxiliary Maps

The speech recognition system described in [Kohonen 1988b] based the decisions of the phonemes on the *quasiphoneme sequence* produced by the *phonotopic map*. The detection of phonemes was done by searching the sequence for static areas, that is, for subsequences where the same phoneme label was repeated.

For stationary phonemes, for example vowels in the Finnish language, the quasiphoneme sequence is an adequate base for decisions since nearly exclusively the same quasiphoneme is repeated for the whole phoneme utterance. Diverse labels are often given only on phoneme borders. The same is not true for non-stationary phonemes where the changing spectra is an essential feature of the phoneme. In Finnish such phonemes are e.g. the unvoiced plosives /k,p,t/ which produce a short 'explosive' (phoneme dependent) burst sound after a short silence. The identity of the unvoiced plosive must be decided during the burst sound. The burst sound is rather short and the spectra changes significantly during it. Therefore, the quasiphoneme sequence does not contain a repeated sequence of the same quasiphoneme and the phoneme recognition cannot be based on just a few labels.

In [Kangas et al. 1989] a system is described that concentrates on the analysis of difficult non-stationary speech sounds using an auxiliary Self-Organizing Map to perform the classification. In the Finnish speech recognizer, the auxiliary SOM recognized the unvoiced plosives, and in a system constructed for Japanese also the voiced plosives /b,d,g/ were recognized separately. The time-sensitive nature of these phonemes are taken into account by the combination of a heuristic search for the exact location(s) of the interesting change areas (where further analysis is concentrated) and a concatenation of original sample vectors to attain a sufficient coverage of the speech signal in one classifiable sample.

The unvoiced plosives /k,p,t/ consist of two parts: the silence where some part of the speech channel is closed, and the burst sound where the speech channel is opened suddenly and air flows rapidly through the channel. The silent part is the same for all plosives, the differences can only be heard during the burst. The 'location of interest' was found by following the speech signal directly. During silence the amplitude of the signal was very low. When the power of the signal exceeds a certain threshold, the burst part of the plosive is indicated.

The auxiliary unvoiced plosive-SOM on the speech recognizer is used in combination with the original system as follows:

1. Presence of an unvoiced plosive is detected by the main classifier. It is reasonably safe to assume that unvoiced plosives are almost always detected correctly because they differ clearly from other phonemes.

2. The speech signal amplitude is followed until the beginning of the explosive burst sound is located.

3. Two speech sample vectors from the burst (20 ms apart) are concatenated and classified by the auxiliary SOM.

4. After unvoiced plosive has been classified the focus is returned to the classification with the main SOM classifier.

The beginning of words have the same characteristics as the plosives: first there is silence, then the amplitude of the signal rises suddenly. Therefore the beginning of a word, the glottal stop without an unvoiced plosive, has to be included into the plosive classifier, as an alternative to the plosives.

3.3.1 Experiments

In [Kangas et al. 1989] comparison tests of the speech recognition system quality with and without the auxiliary Self-Organizing Map are described.

The speech data used in the tests was collected from four male speakers. Each speaker read aloud twice a list of 300 words. One set was used for training the recognizer and the other for the tests. The speech signals were collected using a sampling frequency of 13 kHz. The power spectra of the speech signal was computed every 9.83 ms using a discrete Fourier transform (DFT) routine with 256 samples. From the spectra 15 amplitude values were collected to form a feature vector. The auxiliary maps used 30 component feature vectors formed by concatenating two 15 component vectors.

For the training two to three 30 component feature vectors were collected from every occurrence of unvoiced plosive to ensure some tolerance for the detection of the burst. The recognition stage used only one feature vector to define a phoneme class.

The recognition experiments used only the phonemic recognition stage of the system, that is, no postprocessing of the phonemic strings was performed to get better (more correct) written forms.

The recognition system produced a phonemic string for each utterance it was given. These strings were compared with the correct character strings using a Dynamic Programming algorithm to reveal several possible types of errors. Three different kind of errors were found:

1. *Wrong/Missing.* The plosive was classified incorrectly as a phoneme other than a plosive or the plosive detection was missing completely.

2. *Extraneous.* A plosive was found where there should not have been any phoneme.

3. *Plosive.* A plosive was incorrectly classified as another plosive.

The classification results of these experiments are collected in Tables 3.1 and 3.2.

The percentage of errors was computed by counting all of the cases where a certain error was found and dividing that number by the total number of plosives in the word list. Because there were also cases where extra plosives were found, the sum of the percentages exceeds 100 % (the sum of the first, second and fourth columns in Table 3.2 is 100 %).

Including the plosive classifier greatly improved the speech recognizer. Previously the unvoiced plosives were left undifferentiated, only the fact that the phoneme is from the plosive group was indicated and the further recognition was left to postprocessing. Using the plosive map one can differentiate the unvoiced plosives as well as the other Finnish phonemes.

Speaker	Without the auxiliary map	With the auxiliary map	Plosive recognition
KT	75.8 %	85.8 %	84.8 %
OV	74.1 %	84.1 %	88.2 %
KM	79.2 %	89.4 %	85.5 %
JK	72.4 %	79.4 %	77.9 %

Table 3.1: *Phonemic recognition accuracy without (left column) and with (middle) the auxiliary map. The recognition accuracy of the unvoiced plosives (right) is comparable to the other phonemes.*

Speaker	Correct	Wrong/Missing	Extraneous	Plosive
KT	84.8 %	7.6 %	7.2 %	7.6 %
OV	88.2 %	7.1 %	5.4 %	4.7 %
KM	85.5 %	3.1 %	5.9 %	11.4 %
JK	77.9 %	11.1 %	12.5 %	11.0 %

Table 3.2: *Error types in unvoiced plosive recognition experiments.*

The unvoiced plosives are difficult phonemes to recognize, even with the auxiliary maps. The problem has been approached lately using the Hidden Markov models (HMMs) together with several classifiers producing the necessary symbol strings where one of the classifiers is dedicated to differentiating between the unvoiced plosives (and is thus a descendant of the original auxiliary plosive map that is described above).

3.4 Time-Delay Self-Organizing Map

The auxiliary maps used for unvoiced plosive detection [Kangas et al. 1989] took into account some time dependent features of the speech signal. The complete system, however, based the decision of most phonemes on a very short duration of the speech signal. The recognition accuracy of unvoiced plosives was clearly improved when the pattern vectors were concatenated. The same improvement in recognition accuracy was not expected with the other phonemes but in later experiments some increase in phoneme recognition accuracy was achieved.

3.4.1 Time-Delay SOM

In the unvoiced plosive classification task (section 3.3), the recognition of unvoiced plosives /k,p,t/ was based on a model where two speech samples (20 ms apart) were stored into the memory and the recognition was based on that collective data. The concatenation model can be extended to help in the recognition of all phonemes. Although most of the phonemes (for example, in the Finnish language) do not have such a clear time dependent nature as the plosives, the new model can improve the recognition of phoneme borders, for example, where the signal is changing rapidly. Accurate border detection could then, in turn, improve the phoneme recognition by ensuring that the number of phonemes to be recognized is correct.

Concatenating several pattern vectors, as is done in the TDNN model (section 3.1.1) can be done with Self-Organizing Maps as well. The main differences are that SOM is a one-level model, that is, there are no hidden layers and therefore the concatenation is only done for the input pattern vectors, and that SOM is an unsupervised learning algorithm.

In [Kangas 1990], [Kangas 1991a] and [Kangas 1991b] the model of SOM with concatenated input pattern vectors is explained and some experiments are described. It was shown that

the concatenated vectors store some additional information about the temporal nature of the speech signal in a useful way compared to the original (short-time) pattern vectors.

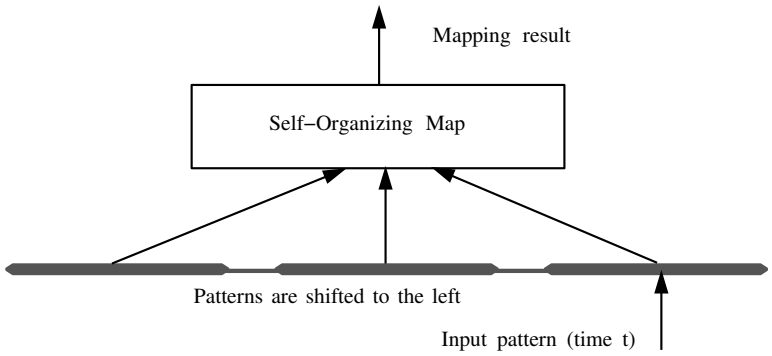

Figure 3.5: *The structure of Self-Organizing Map model with concatenated input pattern vectors. The current pattern vector is read into the rightmost slot of the bottom layer and the stored samples are shifted to left. Stored samples are used in the computation of the SOM response. The number of storage slots in the shift layer can be selected quite arbitrarily.*

In Figure 3.5 the model is shown. The lowest row of the shift register contains the n most recent input patterns, concatenated together. The map sees them in parallel and the decision of the most active area in the map is made the usual way.

3.4.2 Experiments

The speech data was gathered from a native Finnish speaker who read aloud four times (on different days) 311 Finnish words (in an office environment). The sound was digitized using 16 bits, collected at 25.6 kHz and reduced to 12.8 kHz. The digitized speech signal was windowed using a 256 point Hamming window every 10 ms. Successive windows thus shared 128 points.

The windowed data was transformed using a DFT algorithm to produce a 128 point power spectrum. A cepstrum representation of the speech samples was built by collecting mel-scaled channels by applying triangular filters to the power spectrum and cosine-transforming these values. The experiments used 22 cepstrum features (for information about cepstrum features see [Schafer et al. 1990] and [Davis et al. 1990]). The mel-scaled channels were situated as follows: 10 channels spaced evenly between 0 and 1 kHz and 12 channels spaced logarithmically between 1 and 5 kHz.

Four separate tests were done: at a given time one word set was used as a test set while the other three were used for training (leave-one-out experiments). The speech recognition system consisted of several stages. In these experiments the phoneme classification accuracy was tested both before and after the segmentation of quasiphoneme strings. The results are, therefore, given for the two scenarios:

1. The maps that were used for recognition were fine tuned using the LVQ1 (Learning Vector Quantization, [Kohonen 1990]) algorithm and the experiment consisted of recognizing known (presegmented) speech signal sequencies.

2. The maps were fine tuned using the LVQ1. In the experiment the speech data consisted of continuous speech signal without any knowledge of phoneme segment boundaries.

The segmentation was done using the Hidden Markov-models based segmentation algorithm (see e.g [Rabiner 1990]). The classification results included errors from the segmentation phase and errors in labeling of the detected segments.

The recognition of unvoiced plosives /k,p,t/ in the Phonetic Typewriter was done using a separate auxiliary map [Kangas et al. 1989]. In these experiments the plosives were either classified into a single class ('kpt-class') or classified separately ('kpt-diff'). The results are shown in separate tables. All other Finnish phonemes were classified separately.

Test data	Isolated phonemes	Unsegmented speech data
Old system	10.4 %	9.2 %
Time delays	5.0 %	7.6 %

Table 3.3: *The error percentages in phonemic recognition using one pattern sample vector at a time (Old system) and using concatenated pattern vectors (Time delays). The unvoiced plosives have been recognized as a 'kpt' group only.*

As can be seen from Table 3.3 that the recognition accuracy improved when temporal features were taken into account. The improvement was especially clear in the isolated phoneme recognition experiment (before segmentation). A smaller increase in recognition accuracy could be seen in the unsegmented speech system where segmentation of the signal was done automatically.

Test data	Isolated phonemes	Unsegmented speech data
Old system	17.8 %	15.8 %
Time delays	7.8 %	10.1 %

Table 3.4: *The error percentages in phonemic recognition as in Table 3.3, however, the unvoiced plosives (/k,p,t/) has been recognized separately.*

When the unvoiced plosives are recognized separately, the relative improvement of the recognition accuracies are even greater. It follows from the fact that the spectral properties of the unvoiced plosives /k,p,t/ vary greatly during the pronunciation and the time delay method used in the experiments can compensate for it rather well.

It can be seen that the complete speech recognition system using the HMM segmentation algorithm does not result in as much improvement in recognition results as was achieved for the isolated phoneme recognition task. The main reason is probably due to the HMM model itself which also takes the time varying nature of speech signal into account. In principle the HMM model uses the same information about the time signal that was hoped to be used with the time delay neural model.

3.5 Self-Organizing Maps with Leaky Integration of Signals

In the previous section, the signal (for a certain time interval) was stored in shift register to hold the information for a fixed length of time. Obviously some kind of signal memory has to exist in all systems where knowledge about previous samples is used in signal processing. Information about the previous signal values does not have to be stored in its raw form although it is the simplest way. In the following experiments the signal values were stored in a processed form for use in the speech recognition system.

3.5.1 SOM Responses

The Self-Organizing Map performs a mapping of input signal samples to the output space. The output space can be defined in several ways. In most applications we are only interested in the location of the best matching map unit, and the absolute value of the response of each map unit is not considered. The output space is defined by the coordinates of the map units.

In other applications we might also be interested in the absolute or relative values of the responses. For example, the relative values of the responses tell us whether the best matching unit is significantly better than the next best ones.

In [Kangas 1990], [Kangas 1991a] and [Kangas 1991b] the Self-Organizing Map algorithm is used to transform the pattern vectors to 'images' of the samples. These images are then stored into a leaky integrative memory that contain the traces of recent signal values in a compact form. The image is then input to another SOM which classifies it. In Figure 3.6 the system model is shown. (A similar spatiotemporal system is introduced also in [HN 1990].)

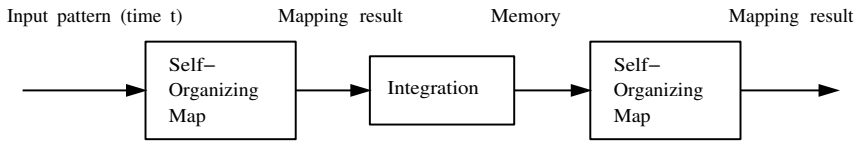

Figure 3.6: *The structure of the Self-Organizing Map model with transformed signal patterns stored in leaky integration memory. The present pattern vector is mapped by the first Self-Organizing Map (leftmost) and the transformed form is saved in the memory (middle). Stored samples are then used in the computation of the second SOM response (rightmost) which yields the recognized class.*

3.5.2 Storage of the Signal in the Integrators

The responses for the integration memory model were computed as follows: for each unit of the SOM we computed the squared Euclidean distances between the input pattern vector and the stored weight vectors. The squared distances were then inverted and normalized so that the nearest unit got a normalized value of 1.0 (and the other values were smaller than that).

The normalized values were 'sharpened' by squaring them a few times. Thus the small values become even smaller and the (usually few) large values were emphasized.

The responses were then collected in the memory using so called 'leaky integrators'. Each integrator contained a value for one map unit in the first SOM. The values were adapted by the latest response values through the following equation:

$$s(i,t) = s(i,t-1) + \beta(r(i,t) - s(i,t)), \tag{3.1}$$

where $r(i,t)$ was the response for map unit i at time step t and $s(i,t)$ was the memory value for map unit i on the same time step. The adaptation factor β was finally selected to be 0.3 which value gave the best recognition results in the following experiments.

(Somewhat similar ideas of storing the signal values to leaky intergrators for SOM classification have been published also in [Chappell et al. 1993] and [Reiss et al. 1991]. These proposals are rather simple, however, when the signal is stored and integrated directly without any transformations.)

3.5.3 Experiments with Speech Data

In [Kangas 1990], [Kangas 1991a] and [Kangas 1991b] some experiments using natural speech signal are detailed. The speech data and preprocessing routines, as well as the postprocessing routines, were similar to the experiments explained in section 3.4.2. The objective was to recognize the Finnish phonemes in a list of utterances as well as possible.

The phoneme recognition results are collected to the Tables 3.5 and 3.6 below.

Test data	Isolated phonemes	Unsegmented speech data
Old system	10.4 %	9.2 %
Integration	7.0 %	8.3 %

Table 3.5: *The error percentages in phonemic recognition using either one pattern vector at a time (Old system) or using the system with integrated responses of SOM as the preprocessed pattern vector (Integration). The unvoiced plosives have been recognized as a group only.*

Test data	Isolated phonemes	Unsegmented speech data
Old system	17.8 %	15.8 %
Integration	10.5 %	11.1 %

Table 3.6: *The error percentages in phonemic recognition as above. Each unvoiced plosive has been separated.*

As above (with the concatenation of pattern vectors), inclusion of the time dependent features in the recognition system improves the recognition accuracies compared to the base line system. The improvement is quite clear in the first case where only the isolated phonemes are recognized. In the second case the full scaled recognizer has been used containing the HMM segmenter and the improvement is not very clear, possibly because the HMM model already takes the time dependent nature of the phonemes into account.

The integration of the signal values was implemented in the simplest possible manner, using the linear weighted average. Some other collection methods might produce better results by taking into account the nature of the classified objects.

3.6 Hypermap Structures

In [Kohonen 1991b] a new computing model derived from the basic Self-Organizing Map algorithm is introduced. The new model, called Hypermap architecture, incorporates a method of using several levels of information to select the best matching unit from the Map units. For example, the context in which the pattern occurred is first used to find a subset of units from which the final selection is done with the pattern itself. Thus, the model could use information from several sources to select the final match.

The Hypermap architecture can be applied in the processing of sequential data structures. The context of the 'interesting' pattern could be the most recent patterns or some processed form of them. Then the final recognition can be done using the pattern itself, and the identification of identical patterns are thus affected by the history, the context in which they occurred.

3.6.1 Experiments

In [Kohonen 1991b] some phoneme recognition experiments are explained. The objective was to recognize isolated phonemes when the phoneme location was already known. The pattern vectors used in the examples were concatenations of three parts, each part being an average of three cepstral vectors containing 20 components (computed 10 ms apart). The context vector was a concatenation of four parts, each part being an average of five cepstral vectors (see Figure 3.7). Both pattern vectors and context vectors were centered around the middle of the phoneme to be recognized.

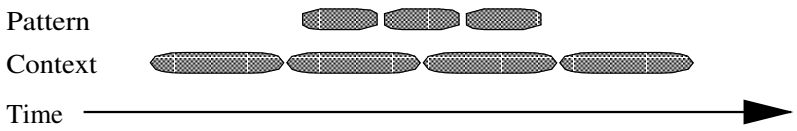

Figure 3.7: *The pattern and context parts of the hypermap input pattern vector were collected from the cepstral vector stream. The pattern vector was concatenated from three averages of three cepstra, the context vector was concatenated from four averages of five cepstra. Both windows were centered around the middle part of the phoneme.*

The number of units in the maps was rather large, there were 2000 units in all cases, while in the real time speech recognition systems described in previous sections the number of units was usually 216. The wide variety of the possible context vectors, however, requires more map units.

The recognition accuracies achieved in the tests are shown in Table 3.7. It was shown that the recognition accuracy was improved when the context knowledge is used, although the difference between the one phase and two phase recognition is rather small when concatenated vectors are used. The error percentage is decreased from 3.2% to 2.3%.

Pattern vectors	Correct phonemes
Single cepstra only	89.6 %
Concatenated cepstra only	96.8 %
Two-phase recognition system	97.7 %

Table 3.7: *The recognition percentages for phonemic recognition using a single cepstrum in a one-phase recognition system, concatenated cepstra in a one-phase recognition system and a two-phase recognition system with concatenated cepstra in the pattern and context vectors.*

It should be noted that the context part may be used in the selection of speaker specific parts of the classification maps as well. There may be subareas in the SOM for several speakers and it is not necessary to know the identity of the speakers during the training.

In [Brückner et al. 1992] the Hypermap architecture is generalized to contain an arbitrary number of context levels (although the original paper [Kohonen 1991b] does not in fact restrict the number of context levels to two). Some experiments were briefly stated demonstrating the power of the extended model in a speech representation. The work by Brückner et el. was continued in [Brückner et al. 1993] where an application of the Hypermap structure to speech recognition was explained.

3.7 Other Studies and Applications

In this subsection we have collected some studies and applications of Self-Organizing Maps used in the analysis of pattern sequences. In these examples some contextual information has been collected from the input pattern sequences.

3.7.1 Analysis of Protein Sequences

Through some important developments in molecular biology the number of known protein sequences is growing rapidly. The number of sequences will increase further when the Human Genome Sequencing Project is in full progress. A problem arising from the great number of these sequences is the rapidly growing volume of comparison work needed when looking for *homologies* between the potentially new sequence and the whole set of the old sequences in the protein databases.

In a series of papers [Ferrán et al. 1991a], [Ferrán et al. 1991b], [Ferrán et al. 1992a], [Ferrán et al. 1992c], [Ferrán et al. 1992b] an application of the Self-Organizing Map algorithm to the classification of protein sequence is explained. The SOM was used to cluster the sequences according to similarities in the amino acid sequences, thereby the number of potentially similar sequences was diminished for more thorough comparison.

The proteins are composed of a long sequence of amino acids, selected from a list of 20 different acids. Functionally similar proteins usually have rather similar amino acid sequences. However, some changes in the amino acid sequence do not affect the functioning of the protein. First of all, some amino acids may be changed while the number of acids in the sequence is kept fixed. Secondly, the length of the sequences may vary radically because of some additional parts in the sequences. Thus the straightforward comparison of sequences is difficult.

For the comparison task using the SOM algorithm the amino acid sequences were transformed to suitable input patterns. This transformation was done by computing the bipeptide composition frequencies of each protein. In other words, a 20 by 20 matrix was constructed where the frequencies of each amino acid pair present were tabulated. After normalization the matrix was then used as the input pattern vector.

The SOMs were trained with a variable number of protein sequences. The resulting mappings of protein sequences to map positions were as expected: the related proteins were mapped to nearby locations in the map. In classification tests of unknown sequencies the results indicated that the protein sequences can be clustered quite reliably into known protein families.

Protein evolution

The amino acid sequences of functionally identical proteins of different species differ to some degree. The difference is smaller for more related species and larger for less related species. In [Ferrán et al. 1992b] the similarities between the cytochrome *c* proteins belonging to different species were examined in the SOM. The SOM mapped similar sequences to nearby locations in the map thus indirectly revealing some evidence of the relations between the species from which the proteins have been extracted.

The resulting order of proteins in the map was intuitively very appealing. For example, the proteins from vertebrates were located in a continuous zone on the map, with sequences of fishes, amphibian, reptiles, birds and mammals associated to different neurons. The proteins from plants were located in another continuous zone on the map.

3.7.2 Analysis of Syntactic and Semantic Knowledge using Self-Organizing Maps

Extracting the meaning of arbitrary sentences in natural language has proven to be a very challenging problem. Traditional solutions have been based on the Artificial Intelligence (AI) approaches but it has been found that these methods are not flexible enough to attain the power of natural languages. The AI techniques usually require strict definitions of the rules by which the meaning is extracted from the sentences, and it is not possible to define the rules of the natural language production in such details.

The artificial neural network methods, on the other hand, are self-learning. Thus the rules need not be stated but are learned from examples, enabling the neural network systems to deal with the exceptions in the usage of natural language. There have been many papers where different neural network models have been used for natural language analysis.

Extraction of Syntactic Knowledge

In [Ritter et al. 1989b] it was demonstrated that the Self-Organizing Map algorithm was capable of creating a mapping which reflected some syntactic relations between words (in sentences which were generated using a simple grammar). The feature vectors extracted from the sentences were categorized to syntactic groups defined by their role and meaning. The SOM representation of the sentences was created by introducing some logical distance measures between the words. One of the measures was based on the definition of similarities in the attribute lists associated to each word, the other was defined as follows:

> The three word sentences were coded word by word with arbitrary (random) vectors such that each word given in the context of predecessor/successor words was given a code. The distance between words was then measured using the Euclidean distance between these codewords.

The distance measure as defined in the second measure gives some indication of the roles of the words through the context part of the code words: words which have similar context parts have more similar codes than words that occur in dissimilar contexts even though the word part itself does not convey any information about the similarity.

In [Honkela et al. 1991] it is argued that the Self-Organizing Map might be used to model the imprecise meanings of words in natural expressions for application where some symbolic representations of words are needed. The SOM algorithm is well suited to the applications above. Similar words are naturally clustered together in the map surface and if the codification of the words (and the context) is done properly, the syntactic elements are also displayed in the mapping.

Land value appraisal

The appraisal of land value for sale proposals is a demanding task because the price depends on many locational and psychological factors. In [Carlson 1991] the Self-Organizing Map algorithm was used for visualizing the dependencies of the various factors for price determination. The feature vectors consisted of a group of factors (including sale time, location, area, direction to lake, distance of sight, etc.), that were determined for each property and the map was trained with these input data vectors. The maps could then be used to extract the dependencies between the various parameters and thus to approximate the prices for new properties.

Data categorization

The syntactic structure of strings of words has also been applied in [Scholtes 1991c], [Scholtes 1991a], [Scholtes 1991b], and [Scholtes 1992] where a Self-Organizing Map was used to find the word categories. In [Gera 1992] the idea of semantic SOMs ([Ritter et al. 1989b])

was utilized to generate the feature vectors for detailed analysis of structural data. Each SOM ('low level module') was dedicated to the extraction of features from one information source and the results of several SOMs were combined to be processed by 'higher level modules'. The information in the experiments was taken from different sources, containing strings, images, etc. The objective was to obtain the attributes for syntactic analysis automatically from low level information.

In [Finch et al. 1992] the Self-Organizing Map was used to cluster the data according the context knowledge available. Two experiments were conducted. In the first the letters of common words were clustered by SOM using the previous two letters and the following two letters as the context. In the resulting maps the vowels and the consonants were separated. In the second experiment words were clustered. The context was from the previous and following two words. Some of the clusters formed in the maps have direct interpretation as obvious word categories.

The extraction of relevant rules is often the most difficult part when compiling an expert system for a specific task. In [Ultsch 1992] the Self-Organizing Map was used to extract regularities from a set of example data. The extracted rules are then used in the next stage of the system in symbolic form. The logical proofs can be automated by using, for example, Prolog language. But even though a Prolog program can mechanically compute the proof, the computations can become extremely inefficient with standard control strategies. In [Ultsch et al. 1991] the Self-Organizing Map algorithm was used to find and store successful strategies for Prolog programs to control the execution order in future programs. The results suggested that the SOM could successfully store such control strategies.

3.7.3 Image Analysis by Self-Organizing Maps

The image data consists of two dimensional sequences. In the following chapters some examples of the analysis of image data by Self-Organized Maps are explained. In these examples the sequential aspects of image data have been taken into account.

Texture[1] Classification

An important part of the image pattern recognition task is the segmentation of image components from image data. On the borders of objects, the task is reduced to the separation of two different textures, one texture for each part of the image. For human observers the separation is often rather straightforward but developing automatic devices for the classification of textures has proven to be difficult.

The Self-Organizing Map algorithm has been applied to the segmentation and recognition of textures. The SOM is well suited to the segmentation task because map units are automatically distributed to cover the input space spanned by the samples. Thus the exact properties of the unknown textures need not be known beforehand.

In [Lampinen et al. 1989] ([Lampinen 1992a]) the Self-Organized Map algorithm was used to segment different (regular) textures without prior knowledge of the texture types or number in the image data. Feature extraction was based on modeling the regular texture by the autoregressive (AR) model which is able to model the sequential nature of neighboring pixels.

The segmentation and classification of stochastic textures was described in [Visa 1990]. The Self-Organizing Map was used to analyze the stochastic texture of paper surfaces for paper quality analysis. The preprocessing of the paper textures was done using co-occurrence matrices which modeled the stochastic nature of the texture and revealed the statistical properties which the nearby pixels must share to generate the overall (regular) impression

[1] Image textures are structures where more or less similar (small) image components collectively form a rather regular structure (to the human observer the textures have often rather strong regularities in them which are usually quite hard to be quantified).

of the texture. The work on stochastic textures was later generalized to other application areas. In [Simula et al. 1992] several application areas are described where a SOM based texture segmentation device has been used. The application of the method to classification of clouds in satellite images was rather straightforward. It was also applied to the analysis of timber in saw-mills [Yläkoski et al. 1993] where the quality and expected price of timber was determined based on a visual image.

Image analysis

In [Gemello et al. 1991] the Self-Organizing Map was applied to a sequence detection task. The problem was to find the contour of a speaker in videophone images. The solution using SOM was based on finding the approximate contour points from the image and then modeling the contour sequence by neighboring units in a one dimensional map. The solution has some similarities to several studies where the Traveling Salesman Problem has been solved using Self-Organizing Map algorithm (see [Hueter 1988], [Angèniol et al. 1988]).

In [Morasso 1991] the SOM was applied to the recognition of handwritten text. The sequential data in this application was acquired directly from the pen by following the pen tip movement during the writing. The first level features were the fast movements of the pen tip (the strokes) quantized in five positions: the begin point, the end point and three points between them. The second level features consisted of 2 to 7 consecutive strokes, characterizing one letter. Both the first level features and the second level features were recognized by the SOM algorithm.

In some papers the image data considered had some syntactical structure that was directly applied to the image processing. In [Maggioni et al. 1991] the SOM algorithm was used to recognize the orientation of a three dimensional object in a two dimensional scene. The input parameters were the positions of the corners of the object.

In [Lampinen 1992b] the SOM algorithm was extended to a hierarchical system where the first layer map generated the input for the second layer map, and so on. The hierarchical SOM algorithm was used to extract the features for a recognition system capable of recognizing, for example, faces. The SOM algorithm applied gathered the features to meaningful clusters while preserving the information of the data structures while diminishing the dimensionality. The final classification of faces was done with subspace methods using the automatically extracted features.

3.7.4 Analysis of Medical Signals and Other Sequences

In [Elo et al. 1992] the problem of monitoring a very large amount of data produced by the electroencephalogram (EEG) measuring device is explained. The authors applied the SOM algorithm to reduce the amount of data to get only the relevant parts of the EEG signal by classifying the raw signal. The experiments indicated that the system could be efficiently used in EEG classification work. The EEG signal was also classified in [Brückner et al. 1992] where the novel extension of the Self-Organized Maps, the Hypermap architecture [Kohonen 1991b] was used. The experiments were preliminary and no classification accuracies were given.

In [Kallio et al. 1991] a speech sound visualizing system based on SOM algorithm was modified to classify lung sounds. The results indicated that the system was quite easily modifiable to a classification system of various normal and abnormal lung sounds.

There have also been some other papers where SOM algorithm has been used in the analysis of sequential signals. For example in [Grabec 1991] the SOM algorithm was used to model a chaotic generator. Another example is [Binks et al. 1991] where Self-Organized Maps were used to detect typical price development patterns in financial records for investment decisions.

Part III

Future Directions

Chapter 4

Direct Representation of Pattern Sequences by Self-Organizing Operator Maps

The Self-Organizing Map algorithm is usually defined as a method in which arbitrary sample vectors are compared to weight vectors using their (Euclidean) distances in the vector space. Thus, the vectors are processed one vector at a time. The formulation unnecessarily restricts the algorithm to the processing of static pattern vectors (one is forced to use suitable codings of dynamic signals before the SOM algorithm can be used).

The algorithm can easily be generalized for use in the analysis of dynamic effects in signal values by associating each unit in the Map with a dynamic operator [Kohonen 1993c], [Kohonen 1993a]. These operators can, for example, filter a dynamic signal. Quite arbitrary operators can be defined, to the extent that the parameters which optimally fit a given sample vector are not directly derivable from the input sample.

4.1 Operators in a Self-Organizing Map

In the original SOM algorithm we associated each unit i with an operator F_i, that was a measure of the distance between the weight vector m_i and a given input sample vector x (for example, $d_i = ||x - m_i||$). More complex operators, however, can be defined. The only requirement for these operators is that some kind of a matching score, a response, can be computed for every operator for every input sample vector.

For example: To perform processing of dynamic signals and analysis of sequential data the operator F_i of unit i may be defined as a filter through which the input sequence is fed and the integrated responses of the filters can be compared. The best matching unit c is thus the one with the maximum total response.

When arbitrary operators are used, finding a suitable operator parameter adaptation procedure may become difficult. How can arbitrary operators be trained? In the original SOM models the objective of the training was to improve the matching of the trained weight vectors and the input sample vector. The same goal is an obvious selection criterion for the operator maps. Even though it is trivial to derive the adaptation laws for the typical distance measures, for example the Euclidean distance, the derivation of similar laws for an arbitrary operator can be rather difficult.

One solution proposed to the training problem in [Kohonen 1993c] was to use a kind of genetic algorithm. The operators in the winner unit and its applicable neighbors (defined by

a certain radius $h(c,i,t)$ (see Equation 1.2)) were *replaced* by better matching operators if such operators could be found. The replacement was done, however, with a probability that corresponded to the adaptation factor $\alpha(t)$ in the original algorithm (see Equation 1.2).

In some cases, it may be possible to find at least an approximate gradient of the operator function and use that approximation to generate the adaptation laws, thus allowing the use of the original SOM algorithms directly. In [Kohonen 1993a] an operator map was trained using the gradient vectors of the error function.

When the genetic approach is used, the search for good candidates to replace the unit parameters is another problem. One possible solution is to generate the replacement candidates by modifying the winning operator F_c found in the Map by randomly modifying some of its parameters. Another solution given in [Kohonen 1993c] was to use a pre-computed list of possible parameter vectors from which the good candidates were searched. The following experiments have adopted the first solution.

4.2 Experiments with SOM and Operators

On the following pages some experiments using the operators with Self-Organizing Maps for analyzing speech data are explained. The experiments are intended to demonstrate that the SOM algorithm defines a topological mapping between the input sample generating process parameters and the Map units even in general.

The Linear Prediction Coding (LPC) has been used as the operator function. The LPC algorithm is described below (for more thorough description of linear coding see, for example, [Makhoul 1975]).

Prediction $s_i'(t)$ of the speech signal value (from operator i) at time t is based on a linear (autoregressive (AR)) model of the previous signal values

$$s_i'(t) = \sum_{k=1}^{n} s(t-k)m_{ik}, \tag{4.1}$$

where $s_i'(t)$ is the predicted signal value, $s(t-k)$ are the previous signal values, n is the number of previous input values used (the order of the LPC model), and m_{ik} are the weighting factors of unit i (the coefficients of the LPC model). The response e_i of the operator in unit i is the squared error of the prediction integrated over the sample vector length and normalized[1] by dividing with the energy of the signal itself according to

$$e_i = \frac{\sum_{k=n+1}^{N}(s_i'(k) - s(k))^2}{\sum_{k=n+1}^{N} s(k)^2}, \tag{4.2}$$

where N is the number of signal values in the sample vector.

The unit c is the winner, if the operator F_c (associated with the operator parameters m_c) produce the smallest response value e_c of all units.

Genetic learning

Two different methods were used to train the operator maps in these experiments. In the first method the training was done using a slightly modified version of the algorithm proposed in [Kohonen 1993c].

The parameter vector components, the LPC coefficients of the best matching operator F_c and some operators F_i in its neighborhood were *replaced* by parameter vector components

[1] The normalization ensures that a multiplication of the signals values by a constant does not show in the responses.

m_{rj} of a parameter vector F_r that produces an even better match if such a parameter vector F_r can be found, and by the parameter vector components m_{cj} of the best matching operator F_c if a better one was not found. The replacement was done, however, with a probability of $p(t) = \alpha(t)h(c,i,t)$ (see Equation 1.2):

$$m_{ij} = m_{rj}, \text{ if } p < \alpha(t)h(c,i,t), \tag{4.3}$$

where p is a random number uniformly distributed in the range [0.0, 1.0] (and independently taken for each unit and each component).

In these first experiments the new parameters, the LPC coefficient vectors, were generated on demand by directly modifying the parameter set m_c of the best matching unit c. The generation was done repeatedly using the following formula

$$m_r = m_c + ap, \tag{4.4}$$

where p is a random number vector, with each component having an even distribution in $[-1.0, 1.0]$ and a is some suitable constant (0.1 in these experiments).

No normalization of the coefficient vectors was performed.

Gradient learning

Another series of experiments (also described in [Kohonen 1993a]) was done using the (traditional) gradient optimization approach. When one is using the LPC coefficients and the error functional above, the gradient function can be derived

$$\frac{\partial e_i}{\partial m_{ij}} = \frac{-2\sum_{k=n+1}^{N}[s_i'(k) - s(k)]s(k-j)}{\sum_{k=n+1}^{N} s(k)^2}. \tag{4.5}$$

Thus training can be done by modifying the parameter vectors

$$m_i(t) = m_i(t-1) - \frac{1}{2}\alpha(t)\frac{\partial e_i}{\partial m_{ij}}, \tag{4.6}$$

where the training coefficient α has to be selected with care[2].

4.3 Speech Data

The speech data used in the experiments was collected with a sampling frequency of 16 kHz using 16-bit samples. No preprocessing was done to the signal values. The sample vectors were concatenated from 128 consecutive signal amplitude values.

The sample vectors were labeled (for further analysis) with phoneme symbols by segmenting the speech signals manually into discrete phonetic segments. The vector labels were not used during the SOM training. Only the middle third of the phoneme duration was selected and used in all the experiments. The border segments of the phonemes were discarded to attain more stable data in the preliminary experiments.

The speech data were collected from one male speaker, who was familiar with speech recognition. The speaker was asked to read a list of common Finnish words.

In the experiments the two data sets (one for training and one for testing) consisted of 12 different phonemes in the Finnish language. The following phonemes were included: /a/, /e/, /i/, /o/, /u/, /ä/, /n/, /m/, /l/, /j/, /#/, and /s/, where # represents silence (for example in plosives /k/, /p/ and /t/). The training set contained 298 sample vectors and the test set contained 305 sample vectors. The number of sample vectors for each phoneme varied from 15 to 34 (see Table 4.1).

[2]In gradient algorithms only the direction of correction is determined by the algorithm, the amount of correction has to be determined by the user.

Phoneme	/a/	/e/	/i/	/o/	/ä/	/s/
Training set	32	33	31	22	34	19
Test set	31	27	29	22	33	18
Phoneme	/u/	/l/	/n/	/m/	/j/	/#/
Training set	23	25	22	15	26	16
Test set	24	26	24	15	26	30

Table 4.1: *The number of samples of each phoneme in training and in test sets.*

In the gradient learning experiments the data sets were used as described earlier. In the evolutionary learning experiments the data sets were restricted to 6 phonemes; the following phonemes were included: /a/, /e/, /i/, /o/, /ä/, and /s/.

4.4 Genetic Training

The sample vectors in the experiments represented 128/16000 s = 8 ms of speech signal. Each value in the vectors was a signed 16 bit integer. The length of the parameter vectors, the number of LPC coefficients, was kept rather small to make the analysis of the results easier (in speech analysis work the order of LPC model, the number of coefficients is equal to the sampling frequence (in kHz) plus some small number (two to five) [Alku 1992][pp. 7.]). The experiments used either 24 or 16 coefficients for the multiplication of the previous signal values.

In the example cases shown below the maps consisted of 10 by 8 units. The training was done in two phases using the following parameter values:

- In the first phase
  - Number of training steps $N = 50\ 000$.
  - Value of initial training parameter $\alpha = 0.2$.
  - Radius of the initial training area $r = 10$.
- In the second phase
  - Number of training steps $N = 50\ 000$.
  - Value of initial training parameter $\alpha = 0.05$.
  - Radius of the initial training area $r = 3$.

The training parameter α and the radius of the training area r decreased linearly to 0.0 during the training phase.

The map units were initialized by random values and then trained in two long runs. The quality of the map was tested (using the prediction error criterion) after the initialization and at the end of both of the training sessions. In the examples shown the training run length meant about 80 iterations for each input sample vector.

4.4.1 Results

The Self-Organizing Map algorithm creates a mapping of the input space to map units. Thus there exists a representative unit for each input sample vector in the map. The quality of the mapping can then be measured by computing the difference between the original sample

vectors and the winner units (the quantization error in the original algorithms where the input space was directly partitioned). In these experiments the difference was given by the integral of the prediction errors using the parameters of the best matching unit.

From the results in Table 4.2 it can be seen that during the training procedure the prediction error decreases considerably.

map state	Error with training data	Error with test data
initialized	0.798	0.821
first phase	0.176	0.169
second phase	0.169	0.159

Table 4.2: *The prediction error with training data and independent test data in the training experiments. LPC operators with 16 coefficients have been used.*

The prediction error measures one aspect of the quality of the representation. Another aspect of the SOM algorithm is the conservation of the input signal topology in the mapping. The conservation of topology cannot, however, be measured as easily as the prediction error. In these experiments the topology preservation was estimated by visual inspection from the mapping results.

From the LPC coefficients one is able to estimate the power spectrum of the signal which would give the smallest possible prediction error for the corresponding operator (see e.g. [Makhoul 1975]). The power spectra is estimated by using the following equations:

$$H_i(e^{j\omega}) = \frac{1}{|A_i(e^{j\omega})|}, \tag{4.7}$$

where

$$A_i(e^{j\omega}) = 1.0 - \sum_{k=1}^{n} m_i(k) e^{-j\omega k}. \tag{4.8}$$

Smoother results are obtained by computing logarithm of the results of Equation 4.7.

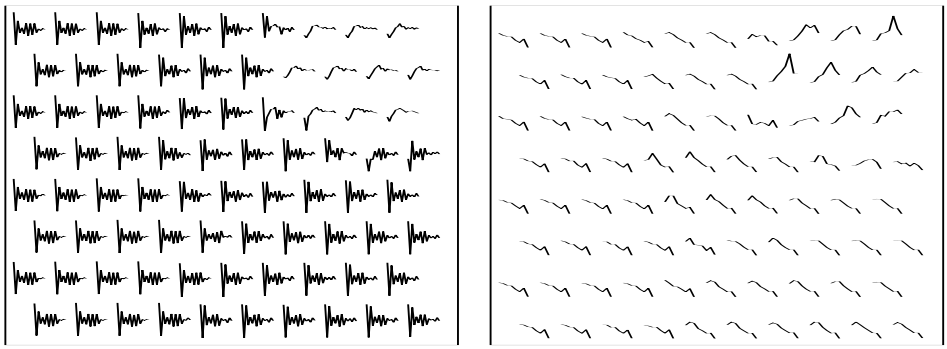

Figure 4.1: *The LPC operator coefficients (left) and the corresponding Fourier coefficients (right) for the LPC operator (16 coefficients) map trained using the genetic programming approach.*

In figure 4.1 it can be seen that neighboring units have rather similar weight vectors and therefore represent similar phonemes, which is to be expected. The spectra in figure 4.1 (computed using the Equations 4.7 and 4.8) reveal more information, one can differentiate between the units by investigating the specific amplitude peaks in the low frequencies

where different vowels have a slightly different structure. The borders between the areas are, however, not very sharp.

The LPC coefficients in all the figures have been scaled using the minimum and maximum coefficient values in the map. The Fourier coefficients have been scaled similarly. The first component of the Fourier coefficients have been removed to emphasize the differences between the components. The LPC coefficients are drawn from left to right, from m_{i1} to m_{in}.

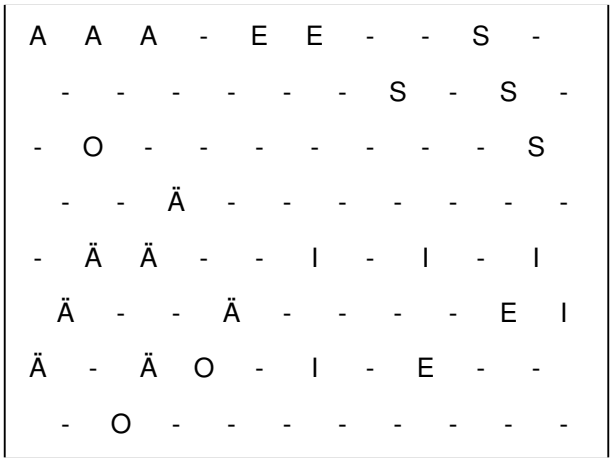

Figure 4.2: *The unit labels for the 16-coefficient LPC operator map trained using the genetic programming approach.*

It can be seen (in Fig. 4.1) that the spectra computed from the LPC coefficients matched quite well with the expectations of the spectra. That is, the LPC coefficients in the map have learned the characteristic features of the speech signal. Comparing Figure 4.1 to Figure 4.2 we notice that representations of a single phoneme are rather similar and differ from the other phonemes.

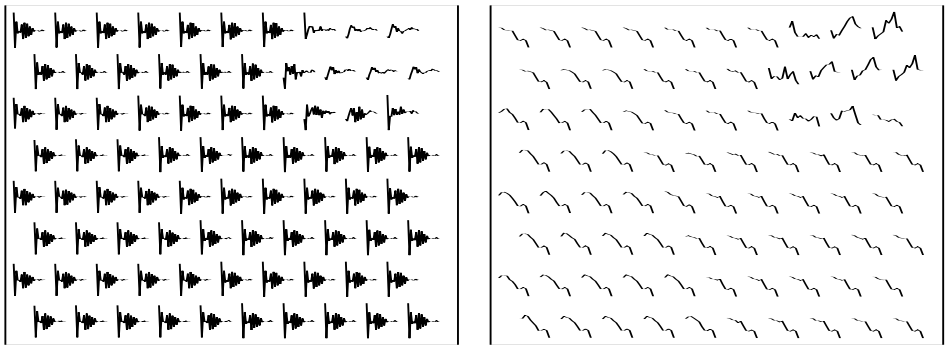

Figure 4.3: *The LPC operator coefficients (left) and the corresponding Fourier coefficients (right) for the LPC operator (24 coefficient) map trained using the genetic programming approach.*

In figures 4.3 - 4.4 the 24-coefficient LPC vector map is shown. Similar indications of ordering of the coefficient vectors are visible also in this case. The LPC coefficient vectors appear rather similar to each other, in the spectral image differences can be seen.

The quantization errors computed for the 24 component LPC maps before and after the training are collected to the Table 4.3.

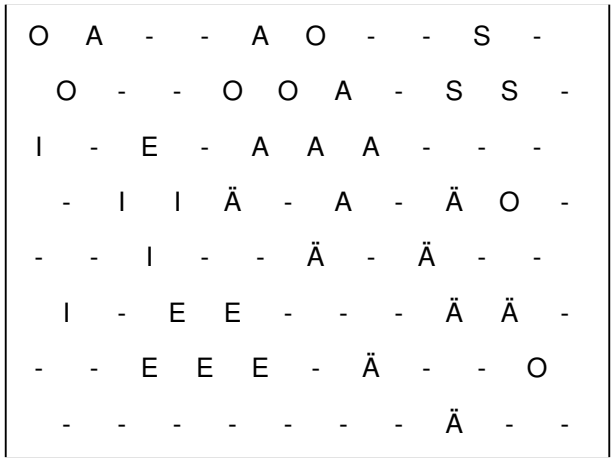

Figure 4.4: *The unit labels for the 24-coefficient LPC operator map trained using the genetic programming approach.*

map state	Error with training data	Error with test data
initialized	1.148	1.175
first phase	0.186	0.176
second phase	0.182	0.172

Table 4.3: *The prediction error with training data and independent test data in the training experiments. LPC operators with 24 coefficients have been used.*

4.5 Gradient Training

When the LPC predictor is used as the operator and the error measure is the integral of the prediction errors over some predetermined time interval, it is possible to derive equations for determining the change to operator parameters for reducing the error values.

Training of the operators can thus be based on the gradient of the error functional. The equations have been described above (the equations 4.5 and 4.6). In the following some experiments with the gradient training algorithm are explained.

The training was done in two phases using the following parameter values:

- In the first phase
  - Number of training steps $N = 20\ 000$.
  - Value of initial training parameter $\alpha = 0.08$.
  - Radius of the initial training area $r = 10$.
- In the second phase
  - Number of training steps $N = 20\ 000$.
  - Value of initial training parameter $\alpha = 0.05$.
  - Radius of the initial training area $r = 3$.

The training parameter α and the radius of the training area r decreased linearly to 0.0 during the training phase.

The map units were initialized by random values and then trained in two long runs. The quality of the map was tested (again using the prediction error criterion) after the initialization and at the end of both of the training sessions. In the examples shown the training run lengths meant about 100 iterations for each input sample vector.

4.5.1 Results

The gradient training method produces more accurate operators than the genetic algorithms described above. The quantitative evidence of this is shown in Tables 4.4 and 4.5 where the prediction error of the two maps with different operator lengths are collected.

map state	Error with training data	Error with test data
initialized	0.690	0.691
first phase	0.142	0.146
second phase	0.134	0.139

Table 4.4: *The prediction error with training data and independent test data in the gradient training experiments. LPC operators with 16-coefficient operators have been used.*

map state	Error with training data	Error with test data
initialized	0.820	0.826
first phase	0.145	0.148
second phase	0.135	0.140

Table 4.5: *The prediction error with training data and independent test data in the gradient training experiments. LPC operators with 24-coefficient operators have been used.*

The qualitative results (the topological mappings) are shown in Figures 4.5 to 4.8. The resulting mappings compare favourably with the corresponding mappings of the genetic approach if one considers the smoothness of the mappings and the distribution of the phonemes.

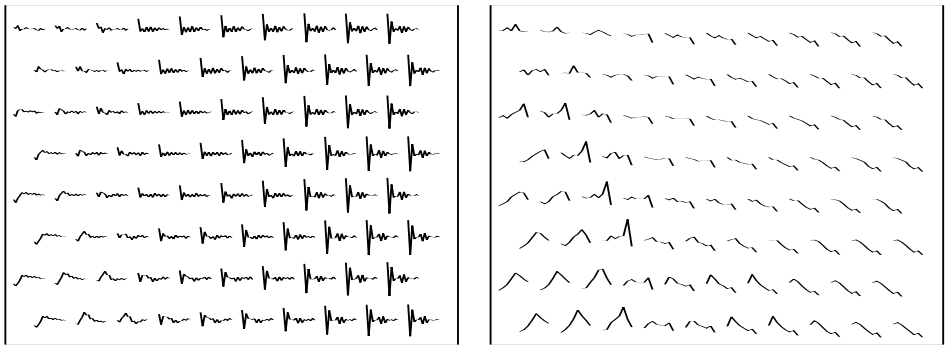

Figure 4.5: *The LPC operator coefficients (left) and the corresponding Fourier coefficients (right) for the 16-coefficient LPC operator map trained with the gradient approach.*

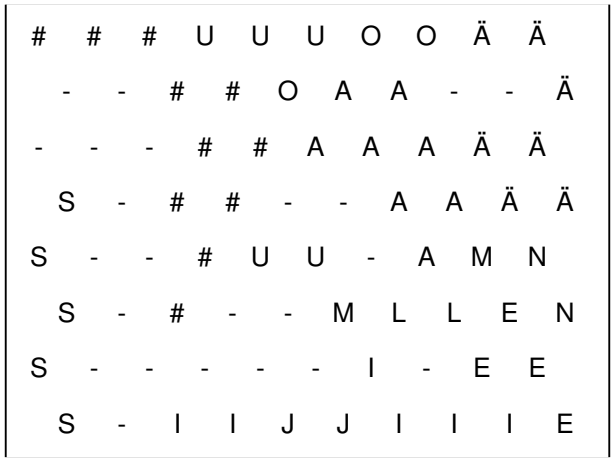

Figure 4.6: *The unit labels for the 16 dimensional LPC operator map trained with the gradient approach.*

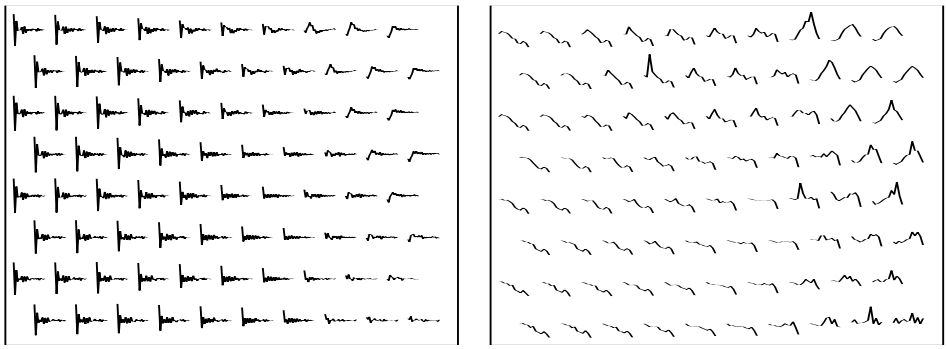

Figure 4.7: *The LPC operator coefficients (left) and the corresponding Fourier coefficients (right) for the 24-coefficient LPC operator map trained with the gradient approach.*

The quality of the maps has to be judged from the labeled maps and from the smoothness of the parameter representations. No quantitative measures are given. The figures, however, reveal that the mapping is rather smooth and the order of phonemes is as expected. The neighboring phonemes in the labeled maps are phonetically related.

4.6 Discussion

In [Kohonen 1993c] and [Kohonen 1993a] the extensions to Self-Organizing Maps to contain arbitrary (input signal) operators are explained. The extensions make it possible to implement more general signal transformations in SOMs without restricting the possible transformation types.

Operators have been used in some special applications in connection with SOMs. Autoregressive (AR) models applied to Self-Organizing Maps were used in a signal segmentation task in [Lampinen et al. 1989] ([Oja 1992]). The signal types were identified by matching LPC coefficients which were stored in the corresponding Map topology. In [Oja et al. 1990] the Self-Organizing Map algorithm was extended by using some parametric representations of curves as input vectors to SOM. The maps consisted of detectors for certain curves in an image.

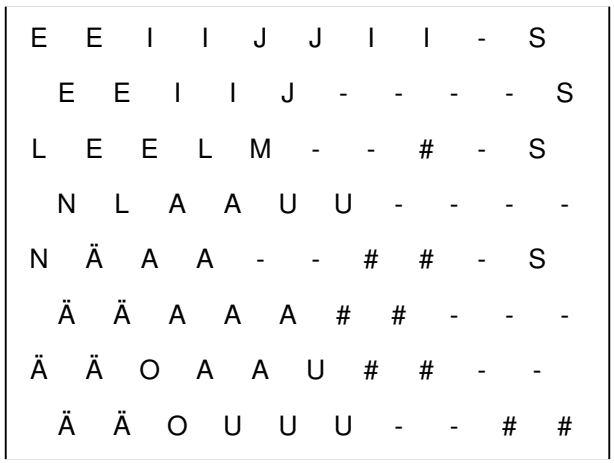

Figure 4.8: *The unit labels for the 24 dimensional LPC operator map trained with the gradient approach.*

Operators were used also in [Tóth et al. 1993] where Self-Organizing Maps and the genetic algorithms were used together. The Map units were sub-populations of solutions to an optimization problem. The solutions were general operators, which were not defined in the paper. The SOMs were used to speed up the learning of the optimal solutions by taking advantage of the topology conserving properties of the Map and the continuity of the solution space.

Chapter 5

Discussion

This work consisted of three parts. The first part, chapter 1, introduced the Self-Organizing Map (SOM) algorithm and then described the mathematical analysis of the SOM algorithm. The second part, chapters 2 to 3, concentrated on applying the Self-Organizing Map algorithm to the analysis of pattern sequences. In the third part, chapter 4, some new ideas about how to generalize the Self-Organizing Map algorithm were explained.

The Self-Organizing Map algorithm implements a non-linear projection of the input signals onto a (usually) two dimensional plane. The training algorithm incorporates non-linear terms. The SOM algorithm has turned out to be a rather difficult mathematical construction and therefore there do not exist many formal analysis papers. Chapter 1 included a review of the analysis work done about the SOM algorithm. The review highlighted the difficulty of the analysis; many of the works were restricted to very simple special cases. No general quantitative analysis results are yet available. However, even these incomplete analysis papers are valuable in describing some special properties of the Self-Organizing Map algorithm. For example, some explanations for the properties of the neighborhood preservation have been given.

The original Self-Organizing Map algorithm was designed to preserve the topological structure of the input patterns. Similar input patterns are mapped to nearby units in the Map. No attention was paid to correlations between successive input patterns; each input pattern was analyzed independently.

In this thesis it has been shown that despite the original restrictions in the Self-Organizing Map algorithm, the Map can be used for the analysis of pattern sequences. The potential of the algorithm has been demonstrated by several experiments with speech data. Other application works have been described to illustrate the diversity of applications which might benefit from the usage of the algorithm.

It was first shown that the Self-Organizing Map algorithm can be used as a visualization tool for pattern sequences. By mapping successive patterns into the Map and visually evaluating the trajectory formed of the successive mapping points, one can extract descriptive information about the development of the process from which the input patterns are derived. Each Map unit represents a state of the underlying process and the topological nature of the Map algorithm ensures that similar process states are clustered in nearby units which can be identified on the Map, thus enabling the Map to be used in process monitoring applications.

In chapter 2 the Self-Organizing Map algorithm was used in voice analysis. Visualization of the voice signal and measuring of the degree of possible deviations from a normal voice was performed using the mapping results. The maps generated by the SOM algorithm proved to be sensitive indicators of speech abnormalities and they are easily understandable tools for visualization purposes.

The next step in the analysis of pattern sequences is to transform the pattern sequence itself so that it can be represented by the static SOM. When the samples of the sequence are represented as a parallel data set, the Map algorithm can be used to analyze the properties of the input space formed.

It was shown that if the correlations between speech samples in the pattern sequence are taken into account, the speech recognition accuracy is increased. In chapter 3 several modifications to the pattern sequence transformation were explained. Some of the modifications have later been implemented in the current version of the speech recognition system developed in Helsinki University of Technology.

Since the Self-Organizing Map algorithm was originally designed for the analysis of stationary signals, direct analysis of dynamic pattern sequences by the SOM would require that the SOM algorithm itself must be modified. In chapter 4 the algorithm was generalized to using operators in the map where each unit in the map was a LPC predictor. Using the operators we were able to generate a topological mapping of speech waveforms directly. It was shown that the waveforms might be categorized using the LPC operator units with the SOM algorithm.

It is evident from the research work described above that the Self-Organizing Map algorithm has ample potential in many different application areas where the analysis of pattern sequences is required. The original SOM algorithm itself is adequate for the analysis work if the input pattern sequence can be modified to a static representation of the process. If the application requires a visualization of the development of the pattern sequence, the SOM algorithm might be used directly to visually display data. Modifications to the original SOM algorithm for sequence analysis, like the Operator Maps, are still under development. The first experiments, however, indicate that development work into different implementations of operators as Map units is worth continuing.

The Self-Organizing Map algorithm differs from other applied neural network models, and it will obviously take some time before enough experience has been gathered in order to convince researchers of the benefits of the SOM algorithm in many applications. Judging from the number of research papers it seems that the neural network research community is just now beginning to find applications where the special properties of the SOM algorithm can best be utilized.

References

[Alander et al. 1991] Jarmo T. Alander, Matti Frisk, Lasse Holmström, Ari Hämäläinen, Juha Tuominen. Process error detection using self-organizing feature maps. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1229–1232. North-Holland, June 1991.

[Alku 1992] Paavo Alku. *An Automatic Inverse Filtering Method for the Analysis of Glottal Waveforms*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 1992.

[Almeida 1987] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a compinatorial environment. In *Proceedings of the IEEE First International Conference on Neural Networks*, pages II–609–618, San Diego, CA, 1987.

[Anderson et al. 1988] James A. Anderson, Edward Rosenfeld, editors. *Neurocomputing: Foundations of Research*. The MIT Press, Cambridge, Massachusetts, 1988.

[Anderson et al. 1990] James A. Anderson, Andras Pellionisz, Edward Rosenfeld, editors. *Neurocomputing 2: Directions for Research*. The MIT Press, Cambridge, Massachusetts, 1990.

[Angèniol et al. 1988] B. Angèniol, G. D. L. C. Vaubois, J. Y. L. Texier. Self-organizing feature maps and and the Travelling Salesman Problem. *Neural Networks*, 1(4):289–293, 1988.

[Arbib 1995] Micheal A. Arbib, editor. *The handbook of Brain Theory and Neural Networks*. Bradford Books/The MIT Press, 1995. (in preparation).

[Bauer et al. 1992] Hans-Ulrich Bauer, Klaus R. Pawelzik. Quantifying the neighborhood preservation of Self-Organizing Feature Maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

[Binks et al. 1991] David L. Binks, Nigel M. Allinson. Financial data recognition and prediction using neural networks. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1709–1712. North-Holland, June 1991.

[Bouton et al. 1991] C. Bouton, M. Cottrell, J. C. Fort, G. Pagés. Self-organization and convergence of the Kohonen algorithm. In N. Bouleau, D. Talay, editors, *Probabilités Numériques*, chapter V.2, pages 163–180. INRIA, 1991.

[Bouton et al. 1992a] C. Bouton, G. Pagés. Convergence in distribution of the one-dimensional Kohonen algorithms when the stimuli are not uniform. Technical report, Laboratoire de Probabilités, Université Paris VI, Tour 56, 3$^{\text{ème}}$ étage, 4, Place Jussieu, 75252 Paris Cedex 05, France, April 1992.

[Bouton et al. 1992b] C. Bouton, G. Pagés. Self-organization and convergence of the one-dimensional Kohonen algorithm with non uniformly distributed stimuli (version 2). Technical report, Laboratoire de Probabilités, Université Paris VI, Tour 56, 3$^{\text{ème}}$ étage, 4, Place Jussieu, 75252 Paris Cedex 05, France, April 1992.

[Bradburn 1989] David S. Bradburn. Reducing transmission error effects using a self-organizing network. In *Proceedings of International Joint*

Conference on Neural Networks, pages II–531–537, Washington, D.C., 1989.

[Brückner et al. 1992] B. Brückner, M. Franz, A. Richter. A modified hypermap architecture for classification of biological signals. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1167–1170. North-Holland, September 1992.

[Brückner et al. 1993] B. Brückner, W. Zander. Classification of speech using a modified Hypermap architecture. In *Proceedings of the World Congress on Neural Networks*, pages III–75–78, Hillsdale, New Jersey, July 1993. International Neural Network Society, Lawrence Erlbaum Associates.

[Carlson 1991] Eero Carlson. Self-organizing feature maps for appraisal of land value of shore parcels. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1309–1312. North-Holland, June 1991.

[Carrato et al. 1993] Sergio Carrato, Giovanni L. Sicuranza, Luigi Manzo. Application of ordered codebooks to image coding. In C. A. Kamm, S. Y. Kung, B. Yoon, R. Chellappa, S. Y. Kung, editors, *Neural Networks for Signal Processing 3 - Proceedings of the 1993 IEEE Workshop*, pages 291–300, Piscataway, New Jersey, USA, September 1993. IEEE, IEEE Service Center.

[Chappell et al. 1993] Geoffrey J. Chappell, John G. Taylor. The temporal Kohonen map. *Neural Networks*, 6:441–445, 1993.

[Conti et al. 1991] Pierluigi Conti, Livia De Giovanni. On the mathematical treatment of self-organization: Extension of some classical results. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1809–1812. North-Holland, June 1991.

[Cottrell et al. 1987] Marie Cottrell, Jean-Claude Fort. Étude d'un processus d'auto-organisation. *Annales de l'Institut Henri Poincaré*, 23(1):1–20, 1987. in french.

[Critchley 1992] D. A. Critchley. Stable states, transitions and convergence in kohonen self organizing maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–281–284. North-Holland, September 1992.

[Cumming 1993] Simon Cumming. Neural networks for monitorig of engine condition data. *Neural Computing & Applications*, 1(1):96–102, 1993.

[Davis et al. 1990] Steven B. Davis, Paul Mermelstein. chapter Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences, pages 65–74. In *Readings in Speech Recognition* Volume 1. (eds.) Alexander Waibel and K-F. Lee, 1990.

[Der et al. 1993a] R. Der, M. Herrmann, Th. Villmann. Spontaneus symmetry-breaking effect in self-organized feature maps: A Ginzburg-Landau approach. In *Proceedings of the World Congress on Neural Networks*, pages II–461–464, Hillsdale, New Jersey, July 1993. International Neural Network Society, Lawrence Erlbaum Associates.

[Der et al. 1993b] R. Der, Th. Villmann. Dynamics of self organized feature mapping. In *Proceedings of the World Congress on Neural Networks*, pages II–457–460, Hillsdale, New Jersey, July 1993. International Neural Network Society, Lawrence Erlbaum Associates.

[Elo et al. 1992] Pekka Elo, Jukka Saarinen, Alpo Värri, Hannu Nieminen, Kimmo Kaski. Classification of epileptic EEG by using self-organizing maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1147–1150. North-Holland, September 1992.

[Erwin et al. 1992a] Ed Erwin, Klaus Obermayer, Klaus Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55, 1992.

[Erwin et al. 1992b] Ed Erwin, Klaus Obermayer, Klaus Schulten. Self-organizing maps: Stationary states, metastability and convergence rate. *Biological Cybernetics*, 67(1):35–45, 1992.

[Ferrán et al. 1991a] E. A. Ferrán, P. Ferrara. Topological maps of protein sequences. *Biological Cybernetics*, 65(5):451–458, 1991.

[Ferrán et al. 1991b] Edgardo A. Ferrán, Pascual Ferrara. Unsupervised clustering of proteins. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1341–1344. North-Holland, June 1991.

[Ferrán et al. 1992a] E. A. Ferrán, P. Ferrara. Clustering proteins into families using artificial neural networks. *Computer Applications in the Biosciences*, 8(1):39–44, 1992.

[Ferrán et al. 1992b] E. A. Ferrán, P. Ferrara. A neural network dynamics that resembles protein evolution. *Physica A*, 185(1-4):395–401, 1992.

[Ferrán et al. 1992c] Edgardo A. Ferrán, Bernard Pfugfelder, Pascual Ferrara. Large scale application of neural networks to protein classification. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1521–1524. North-Holland, September 1992.

[Ferrán 1992] Edgardo A. Ferrán. An ordering theorem that allows for ordering changes. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–165–169. North-Holland, September 1992.

[Finch et al. 1992] S. Finch, N. Chater. Unsupervised methods for finding linguistic categories. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1365–1368. North-Holland, September 1992.

[Gemello et al. 1991] Roberto Gemello, Cataldo Lettera, Franco Mana, Lorenzo Masera. Self organizing feature maps for contour detection in videophone images. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1305–1308. North-Holland, June 1991.

[Gera 1992] Michael Gera. Finding multi-faculty structure. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1357–1360. North-Holland, September 1992.

[Govekar et al. 1992] E. Govekar, E. Susič, P. Mužič, I. Grabec. Self-organizing neural network application to technical process parameters estimation. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–579–582. North-Holland, September 1992.

[Grabec 1991] Igor Grabec. Modeling of chaos by a self-organizing neural network. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–151–156. North-Holland, June 1991.

[HN 1990] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, Reading, Massach., 1990.

[Honkela et al. 1991] Timo Honkela, Ari M. Vepsäläinen. Interpreting imprecise expressions: Experiments with Kohonen's self-organizing maps and associative memory. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–897–902. North-Holland, June 1991.

[Hueter 1988] G. Hueter. Solution of the Traveling Salesman Problem with an adaptive ring. In *Proceedings of the IEEE International Conference on Neural Networks*, pages I–85–92, San Diego, California, 1988.

[Hyötyniemi 1993] Heikki Hyötyniemi. Optimal control of dynamic systems using self-organizing maps. In Stan Gielen, Bert Kappen, editors, *Proceedings of International Conference on Artificial Neural Networks (ICANN-93), Amsterdam*, pages 850–853, London, 1993. Springer-Verlag.

[Ishida et al. 1992] Kazuo Ishida, Yutaka Matsumoto, Norio Okino. The effect of correlated inputs on discrete kohonen networks. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–353–357. North-Holland, September 1992.

[Juang et al. 1991] B. H. Juang, L. R. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, August 1991.

[Jumpertz et al. 1993] Sylvie S. Jumpertz, Eduardo J. Garcia. Image sequence coding using a neural vector quantization. In Stan Gielen, Bert Kappen, editors, *Proceedings of International Conference on Artificial Neural Networks (ICANN-93), Amsterdam*, page 1020, London, 1993. Springer-Verlag.

[Kallio et al. 1991] K. Kallio, S. Haltsonen, E. Paajanen, T. Rosqvist, T. Katila, P. Karp, P. Malmberg, P. Piirilä, A. R. A. Sovijärvi. Classification of lung sounds by using self-organizing feature maps. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–803–808. North-Holland, June 1991.

[Kangas et al. 1989] Jari Kangas, Teuvo Kohonen. Transient map method in stop consonant discrimination. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech89)*, Paris, France, September 26-28 1989.

[Kangas 1990] Jari Kangas. Time-delayed self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks*, pages II 331–336, San Diego, June 1990.

[Kangas 1991a] Jari Kangas. Phoneme recognition using time-dependent versions of self-organizing maps. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 101–104, Toronto, Canada, May 14-17 1991.

[Kangas 1991b] Jari Kangas. Time-dependent self-organizing maps for speech recognition. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas,

editors, *Artificial Neural Networks*, pages II–1591–1594. North-Holland, June 1991.

[Kasslin et al. 1992] Mika Kasslin, Jari Kangas, Olli Simula. Process state monitoring using self-organizing maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1531–1534. North-Holland, September 1992.

[Knudsen et al. 1987] Eric I. Knudsen, Sascha du Lac, Steven D. Esterly. Computational maps in the brain. *Annual Review of Neuroscience*, 10:41–65, 1987.

[Kohonen et al. 1984] Teuvo Kohonen, Kai Mäkisara, Tapio Saramäki. Phonotopic maps – insightful representation of phonological features for speech recognition. In *Proceedings of the 7th International Conference on Pattern Recognition (7th ICPR)*, pages 182–185, Montreal, Canada, July 1984.

[Kohonen et al. 1988] Teuvo Kohonen, Kari Torkkola, Makoto Shozakai, Jari Kangas, Olli Ventä. Phonetic typewriter for Finnish and Japanese. In *Proceedings of the IEEE 1988 International Conference on Acoustics, Speech, and Signal Processing, New York, N.Y., April 11-14*, pages 607–610, 1988.

[Kohonen 1981] Teuvo Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In Erkki Oja, Olli Simula, editors, *Proceedings of Second Scandinavian Conference on Image Analysis*, pages 214–220, Espoo, Finland, 1981. Suomen Hahmontunnistustutkimuksen Seura.

[Kohonen 1982a] Teuvo Kohonen. Analysis of a simple self-organizing process. *Biological Cypernetics*, 44(2):135–140, 1982.

[Kohonen 1982b] Teuvo Kohonen. Clustering, taxonomy and topological maps of patterns. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 114–128, Munich, 1982.

[Kohonen 1982c] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biological Cypernetics*, 43(1):59–69, 1982.

[Kohonen 1986] Teuvo Kohonen. Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech. In *Proceedings of the 8th International Conference on Pattern Recognition (8th ICPR)*, pages 1148–1151, Paris, France, October 1986.

[Kohonen 1988a] Teuvo Kohonen. An introduction to neural computing. *Neural Networks*, 1(1):3–16, 1988.

[Kohonen 1988b] Teuvo Kohonen. The 'neural' phonetic typewriter. *Computer*, 21(3):11–22, 1988.

[Kohonen 1988c] Teuvo Kohonen. *Self-organization and associative memory.* Springer Series In Information Sciences. Springer-Verlag, Berlin Heidelberg New York, 2 edition, 1988.

[Kohonen 1990] Teuvo Kohonen. The self-organizing map. *Proceedings of IEEE*, 78:1464–1480, 1990.

[Kohonen 1991a] T. Kohonen. Self-organizing maps: Optimization approaches. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–891–990. North-Holland, June 1991.

[Kohonen 1991b] Teuvo Kohonen. The hypermap architecture. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1357–1360. North-Holland, June 1991.

[Kohonen 1992] Teuvo Kohonen. An attempt to interprete the Self-Organizing Mapping physiologically. Report A16, Helsinki University of Technology, Laboratory of Computer and Information Science, 1992.

[Kohonen 1993a] Teuvo Kohonen. Generalizations of the Self-Organizing Map. In *Proceedings of International joint Conference on Neural Networks*, Nagoya, Japan, October 1993.

[Kohonen 1993b] Teuvo Kohonen. Physiolocigal interpretation of the self-organizing map algorithm. *Neural Networks*, 6(7):895–905, 1993.

[Kohonen 1993c] Teuvo Kohonen. Things you haven't heard about the Self-Organizing Map. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1147–1156, San Francisco, California, March 1993.

[Koizumi et al. 1991] Takuya Koizumi, Joji Urata, Shuji Taniguchi. A phoneme recognition using self-organizing feature map and hidden markov models. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–777–782. North-Holland, June 1991.

[Kung 1993] Sun Yuan Kung. *Digital Neural Networks*. Prentice-Hall information and system sciences series. Prentice-Hall, London, 1 edition, 1993.

[Lampinen et al. 1989] Jouko Lampinen, Erkki Oja. Self-organizing maps for spatial and temporal ar models. In Matti Pietikäinen, Juha Röning, editors, *Proceedings of The 6th Scandinavian Conference on Image Analysis*, pages 120–127, Oulu, Finland, June 1989.

[Lampinen 1992a] Jouko Lampinen. *Neural Pattern Recognition: Distortion Tolerance by Self-Organizing Maps*. PhD thesis, Lappenranta University of Technology, Lappeenranta, Finland, 1992.

[Lampinen 1992b] Jouko Lampinen. On clustering properties of hierarchical self-organizing maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–1219–1222. North-Holland, September 1992.

[Lang et al. 1990] Kevin J. Lang, Geoffrey E. Hinton, Alex H. Waibel. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 1990.

[Lau 1992] Clifford Lau, editor. *Neural Networks: Theoretical Foundations and Analysis*. IEEE Press, New York, 1992.

[Leinonen et al. 1991] Lea Leinonen, Jari Kangas, Kari Torkkola, Anja Juvas. Pattern recognition of hoarse and healthy voices by the self-organizing map. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1385–1388. North-Holland, June 1991.

[Leinonen et al. 1992] Lea Leinonen, Jari Kangas, Kari Torkkola, Anja Juvas. Dysphonia detected by pattern recognition of spectral composition. *Journal of Speech and Hearing Research*, 35:287–295, April 1992.

[Leinonen et al. 1993a] L. Leinonen, T. Hiltunen, K. Torkkola, J. Kangas. Self-organized acoustic feature map in detection of fricative-vowel coarticulation. *Journal of the Acoustic Society of America*, 93(6):3468–3474, June 1993.

[Leinonen et al. 1993b] Lea Leinonen, Riitta Mujunen, Jari Kangas, Kari Torkkola. Acoustic pattern recognition of fricative-vowel coarticulation by the self-organizing map. *Folia Phoniatrica*, 45:173–181, 1993.

[Linde et al. 1980] Yoseph Linde, Andrés Buzo, Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[Lippmann 1987] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing Magazine*, 4(2):4–22, April 1987.

[Lo et al. 1991] Z.-P. Lo, B. Bavarian. On the rate of convergence in topology preserving neural networks. *Biological Cybernetics*, 65(1):55–63, 1991.

[Lo et al. 1993] Zhen-Ping Lo, Yaoqi Yu, Behnam Bavarian. Analysis of the convergence properties of topology preserving neural networks. *IEEE Transactions on Neural Networks*, 4(2), March 1993.

[Luttrell 1989a] S. Luttrell. Self-organization: a derivation from first principles of a class of learning algorithms. In *Proceedings of International Joint Conference on Neural Networks*, pages II–495–498, Washington, D.C., 1989.

[Luttrell 1989b] Stephen P. Luttrell. Hierarchical self-organizing networks. In *Proceedings of 1st IEE Conference of Artificial Neural Networks*, pages 2–6, London, 1989.

[Luttrell 1990] Stephen P. Luttrell. Derivation of a class of training algoritms. *IEEE Transactions on Neural Networks*, 1(2):229–232, June 1990.

[Luttrell 1991] Stephen P. Luttrell. Code vector density in topographic mappings: scalar case. *IEEE Transactions on Neural Networks*, 2(4):427–436, July 1991.

[Maggioni et al. 1991] Christoph Maggioni, Brigitte Wirtz. A neural net approach to 3-D pose estimation. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–75–80. North-Holland, June 1991.

[Makhoul 1975] John Makhoul. Linear perdiction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[Martín-del-Brío et al. 1993] Bonifacio Martín-del-Brío, Carlos Serrano-Cinca. Self-organizing neural networks for the analysis and representation of data: Some financial cases. *Neural Computing & Application*, 1(3):193–206, 1993.

[Martinetz et al. 1990] T. Martinetz, H. Ritter, K. Schulten. Three-dimensional neural net for learning visuo-motor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136, 1990.

[Mehra et al. 1992] Pankaj Mehra, Benjamin W. Wah, editors. *Artificial Neural Networks: Concepts and Theory*. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[Meyering et al. 1992] Andrea Meyering, Helge Ritter. Learning to recognize 3D-hand postures from perspective pixel images. In I. Aleksander,

J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–821–824. North-Holland, September 1992.

[Miikkulainen 1991] Risto Miikkulainen. Self-organizing process based on lateral inhibition and synaptic resource redistribution. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–415–420. North-Holland, June 1991.

[Morasso 1991] Pietro Morasso. Self-organizing feature maps for cursive script recognition. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1323–1326. North-Holland, June 1991.

[Mozer 1993] Michael C. Mozer. chapter Neural net architectures for temporal sequence processing. In *Predicting the future and understanding the past* (eds.) A. Weigend and N. Gershenfeld, 1993.

[Mujunen et al. 1993] Riitta Mujunen, Lea Leinonen, Jari Kangas, Kari Torkkola. Acoustic pattern recognition of /s/ misarticulation by the self-organizing map. *Folia Phoniatrica*, 45:135–144, 1993.

[Oja et al. 1990] E. Oja, L. Xu, P. Kultanen. Curve detection by an extended self-organizing map and the related RHT method. In *Proc. of the International Neural Network Conference, Paris, France, July 9-13*, pages I 27–30, 1990.

[Oja 1992] Erkki Oja. Self-organizing maps and computer vision. In Harry Wechsler, editor, *Neural networks for perception*, chapter II.9, pages 368–385. Academic Press, Inc., 1992.

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

[Pineda 1989] F. J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1(2):161–172, 1989.

[Rabiner 1989] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[Rabiner 1990] Lawrence R. Rabiner. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–298. In *Readings in Speech Recognition* Volume 1. (eds.) Alexander Waibel and K-F. Lee, 1990.

[Rappa et al. 1992] Michael A. Rappa, Koenraad Depackere. Neural network community survey. Technical Report MIT, E52-538, Massachusetts Institute of Technology, Cambridge, May 1992.

[Räsänen et al. 1990] T. Räsänen, S. K. Hakumäki, E. Oja, M. O. K. Hakumäki. Analysis of r and s disordes in finnish by using a laboratory computer. *Folia Phoniatrica*, 42:135–143, 1990.

[Reiss et al. 1991] Michael Reiss, John G. Taylor. Storing temporal sequences. *Neural Networks*, 4:773–787, 1991.

[Ritter et al. 1986] H. Ritter, K. Schulten. On the stationary state of Kohonen's self-organizing sensory mapping. *Biological Cybernetics*, 54(2):99–106, 1986.

[Ritter et al. 1988a] H. Ritter, K. Schulten. Convergence properties of Kohonen's topology preserving maps: fluctuations, stability, and dimension selection. *Biological Cybernetics*, 60(1):59–71, 1988.

[Ritter et al. 1988b] H. Ritter, K. Schulten. Kohonen self-organizing maps: exploring their computational capabilities. In *Proceedings of IEEE International Conference on Neural Networks*, pages 109–116, San Diego, California, July 1988.

[Ritter et al. 1989a] H. Ritter, T. M. Martinez, K. J. Schulten. Topology-conserving mappings for learning visuomotor-coordination. *Neural Networks*, 2:159–168, 1989.

[Ritter et al. 1989b] Helge Ritter, Teuvo Kohonen. Self-organizing semantic maps. *Biological Cybernectis*, 61(4):241–254, 1989.

[Ritter et al. 1992] Helge Ritter, Thomas Martinetz, Klaus Schulten. *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[Ritter 1989] Helge Ritter. Asymptotic level density for a class of vector quantization processes. Technical Report Report A9, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1989.

[Ritter 1991] Helge Ritter. Asymptotic level density for a class of vector quantization processes. *IEEE Transactions on Neural Networks*, 2(1):173–175, January 1991.

[Sánchez-Sinencio et al. 1992] Edgar Sánchez-Sinencio, Clifford G. Y. Lau, editors. *Neural Networks: Paradigms, Applications and Hardware Implementations*. IEEE Press, New York, 1992.

[Schafer et al. 1990] Ronald W. Schafer, Lawrence R. Rabiner. chapter Digital Representations of Speech Signals, pages 49–64. In *Readings in Speech Recognition* Volume 1. (eds.) Alexander Waibel and K-F. Lee, 1990.

[Scholtes 1991a] J. C. Scholtes. Kohonen feature maps in natural language processing. Technical report, Department of Computational Linguistics, University of Amsterdam, March 1991.

[Scholtes 1991b] J. C. Scholtes. Recurrent Kohonen self-organization in natural language processing. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages II–1751–1754. North-Holland, June 1991.

[Scholtes 1991c] J. C. Scholtes. Unsupervised learning and the information retrieval problem. In *Proceedings of the International Joint Coneference on Neural Networks*, Singapore, 1991. (submitted).

[Scholtes 1992] J. C. Scholtes. Resolving linguistic ambiguities with a neural data-oriented parsing (DOP) system. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1347–1350. North-Holland, September 1992.

[Siegel 1956] Sidney Siegel. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill Series In Psychology. McGraw-Hill, New York Toronto London, 1956.

[Siemon 1992] H. P. Siemon. Selection of optimal parameters for kohonen self-organizing feature maps. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1573–1577. North-Holland, September 1992.

[Simula et al. 1992] Olli Simula, Ari Visa. Self-organizing feature maps in texture classification and segmentation. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1621–1628. North-Holland, September 1992.

[Sorsa et al. 1993] Timo Sorsa, Heikki N. Koivo. Application of artificial neural networks in process fault diagnosis. *Automatica*, 29(4):843–849, 1993.

[Tolat 1990] V. V. Tolat. An analysis of Kohonen's self-organizing maps using a system of energy functions. *Biological Cybernetics*, 64(2):155–164, 1990.

[Torkkola et al. 1991] Kari Torkkola, Jari Kangas, Pekka Utela, Sami Kaski, Mikko Kokkonen, Mikko Kurimo, Teuvo Kohonen. Status report of the finnish phonetic typewriter project. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–771–776. North-Holland, June 1991.

[Torkkola et al. 1995] Kari Torkkola, Teuvo Kohonen. chapter A Hybrid Approach to Speech Transcription. In *The handbook of Brain Theory and Neural Networks* (eds.) Micheal A. Arbib, 1995. (in preparation).

[Tóth et al. 1993] Gábor J. Tóth, András Lőrincz. Genetic algorithm with migration on topology conserving maps. In *Proceedings of the World Congress on Neural Networks*, pages III–168–171, Hillsdale, New Jersey, July 1993. International Neural Network Society, Lawrence Erlbaum Associates.

[Tryba et al. 1991] Viktor Tryba, Karl Goser. Self-organizing feature maps for process control in chemistry. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–847–852. North-Holland, June 1991.

[Ultsch et al. 1989] A. Ultsch, H.P. Siemon. Exploratory data analysis: Using Kohonen networks on transputers. Technical Report 329, Univ. of Dortmund, Dortmund, December 1989.

[Ultsch et al. 1990] A. Ultsch, H. P. Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In *International Neural Network Conference (INNC 90) Paris, France, July 9-13*, pages 305–308. Kluwer Academic Publishers, 1990.

[Ultsch et al. 1991] A. Ultsch, R. Hannuschka, U. Hartmann M. Mandischer, V. Weber. Optimizing logical proofs with connectionist networks. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–585–590. North-Holland, June 1991.

[Ultsch 1992] Alfred Ultsch. Knowledge acquisition with self-organizing neural networks. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–735–738. North-Holland, September 1992.

[Ultsch 1993] Alfred Ultsch. Self organized feature maps for monitoring and knowledge aquisition of a chemical process. In Stan Gielen, Bert Kappen, editors, *Proceedings of International Conference on Artificial Neural Networks (ICANN-93), Amsterdam*, pages 864–867, London, 1993. Springer-Verlag.

[Unnikrishnan et al. 1991] K. P. Unnikrishnan, John J. Hopfield, David W. Tank. Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections. *IEEE Transactions on Signal Processing*, 39(3):698–713, 1991.

[Unnikrishnan et al. 1992] K. P. Unnikrishnan, John J. Hopfield, David W. Tank. Speaker-independent digit recognition using a neural network with time-delayed connections. *Neural Computation*, 4(1):108–119, 1992.

[Utela et al. 1992] Pekka Utela, Jari Kangas, Lea Leinonen. Self-organizing map in acoustic analysis and on-line visual imaging of voice and articulation. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–791–794. North-Holland, September 1992.

[Varfis et al. 1992] A. Varfis, C. Versino. Selecting reliable kohonen maps for data analysis. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages II–1583–1586. North-Holland, September 1992.

[Visa 1990] Ari Visa. *Texture Classification and Segmentation Based on Neural Network Methods.* PhD thesis, Helsinki University of Technology, Espoo, Finland, 1990.

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

[Waibel et al. 1990] Alexander Waibel, K-F. Lee, editors. *Readings in Speech Recognition*, volume 1. Morgan Kaufman Publishers, Inc., San Mateo, California, 1990.

[Walter et al. 1991] Jörg A. Walter, Thomas M. Martinetz, Klaus J. Schulten. Industrial robot learns visuo-motor coordination by means of 'neural gas' network. In T. Kohonen, K. Mäkisara, O. Simula, J. Kangas, editors, *Artificial Neural Networks*, pages I–357–364. North-Holland, June 1991.

[Weigend et al. 1993] A. Weigend, N. Gershenfeld, editors. *Predicting the future and understanding the past.* Addison-Wesley Publishing, Redwood City, California, 1993.

[Williams et al. 1990] Ronald J. Williams, Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990.

[Wu et al. 1992] P. Wu, K. Warwick, M. Koska. Neural network feature maps for Chinese phonemes. *Neurocomputing*, 4(1-2):109–112, 1992.

[Yläkoski et al. 1993] Ilkka Yläkoski, Ari Visa. A two-stage classifier for wooden boards. In *Proceedings of The 8th Scandinavian Conference on Image Analysis*, pages I–637–641, 1993.

[Zador 1982] Paul L. Zador. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE, Transactions on Information Theory*, IT-28(2):139–149, 1982.

[Zar 1984] J. H. Zar. *Biostatistical Analysis.* Prentice-Hall International, Englewood Cliffs, NJ, 1984.

[Zhao 1992] Zuqiang Zhao. Integration of neural networks and hidden markov models for continuous speech recognition. In I. Aleksander, J. Taylor, editors, *Artificial Neural Networks, 2*, pages I–779–782. North-Holland, September 1992.