博士学位論文

論文題目 Self-Organizing Neural Networks Approximating Nonstationary Probability Density Functions
(非定常確率密度関数を近似する自己組織化ニューラルネットワーク)

提出者 東北大学大学院情報科学研究科
情報基礎科学 専攻

学籍番号 8ds8

氏名 中島 平

指導教官	中村　維男　教授
審査委員 （○印は主査）	○ 中村維男 教授 1 山本光璋 教授　2 小林広明助教授 3 藤木澄義 ~~教授~~ 講師　4 　　教授 5 　　教授　6 　　教授

Doctoral Dissertation
Self-Organizing Neural Networks Approximating Nonstationary Probability Density Functions

Taira Nakajima
Graduate School of Information Sciences,
Department of Computer and Mathematical Sciences
Tohoku University

January 20, 1999

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
CH	Competitive Hebbian learning rule
CON	KLN with the conscience mechanism
CSL	Competitive and Selective Learning
K/CH	Kohonen Learning combined with the CH
KL	Kohonen Learning
KLN	Kohonen Learning Network
LOJ	Law-of-the-Jungle
LOJ/CH	LOJ combined with the CH
LOJ/OD	LOJ with On-line Decision of convergence
LOJN	KLN with the LOJ mechanism
MLP	Multi-Layer Perceptron
MSE	Mean Square quantization Error
NG	Neural Gas
NKLN	Normal Kohonen Learning Network
NVQ	Neural Vector Quantizer
PDF	Probability Density Function
SDW	Standard Deviation of Win probability
SOM	Self-Organizing Map
SONN	Self-Organizing Neural Network
TPN	Topology Preserving Network

Acknowledgements

I could not complete this thesis without having a lot of support from many people. Therefore, I thank the people who have directed and helped me accomplish this thesis.

Firstly, I would like to sincerely thank Professor Tadao Nakamura, who is my supervisor. He has given me a precious opportunity to study such an interesting subject. His guidance and encouragement have greatly helped me accomplish this thesis. I also wish to express my appreciation to Professor Mitsuaki Yamamoto and Assistant Professor Sumiyoshi Fujiki for their careful review and helpful suggestions.

I express my sincere gratitude to Associate Professor Hiroaki Kobayashi for his precise comments, enthusiastic discussion, and kindhearted direction throughout the overall course of this research. He has given me precious advice to improve my research.

I also wish to thank Research Associate Hiroyuki Kitajima for his appropriate help.

Special thanks to Hiroyuki Takizawa, a colleague in Nakamura Lab, who has given me various deep suggestions to make me correct some vague points in my research. I am grateful to all members in Nakamura Lab for their interest and critical response to my research activities.

Finally, I must thank my parents for their warm support in my life.

1 Introduction

1.1 Artificial Neural Networks

How does the brain work? It is a traditional but still challenging question for human beings. The following gives some brief representative examples from the viewpoint of human memory. Many people from many study fields have attempted to answer the question. Aristotle[1], who was a philosopher, thought in 400 B.C. that the elementary unit of memory is a sense image. In 1885, Ebbinghaus[2], who was the first psychologist, inspected short term memory. Proust[3], who was a novel writer, precisely described how memory associates in 1910s. Marr[4], who was a mathematician, proposed a theory of cerebral cortex with memory elements in 1970. Kohonen[5], who is an engineer, and Anderson[6], who is a physiologist, simultaneously proposed a mathematical model of associative memory in 1972. As seen on the above, there are various approaches to the brain model.

A powerful way to understand such a complex system is to decompose the system into simple elements and analyze them. The human brain consists of about 10^{11} neurons interconnected by 10^{15} synapses[7][8]. McCulloch and Pitts[9] proposed the first computational model of neuron in 1943. A computational model of a biological neuron is called an artificial neuron. An artificial neural network(ANN) is an information processing system that consists of about $10^1 - 10^3$ artificial neurons[10]. Study on ANN has become an interdisciplinary study subject including mathematics, physics, engineering, and so on. ANNs have been expected to reveal information process of the brain through detailed simulation of biological neurons on computers[8]. They has also received an increasing interest as a new parallel computing model, which are alternative of the existing sequential information processing model[11][12][13].

ANNs for revealing functions of the brain use complex artificial neurons whose dynamics are like biological ones[14][15]. Such ANNs usually work very slow because of the complexity in their units, or each of the ANNs consists of a rather small number of artificial neurons. On the other hand, ANNs for new information processing technology use simple artificial neurons, such as the linear thresholding model[16][17]. Such ANNs are usually excessively simplified, or

sometimes have biologically unacceptable functions to satisfy performance required in practical uses. There seems to be little relationship between these two approaches, but indeed they are not independent. While a new idea to explain functions of the brain could produce a new computational method, a new information processing method could be adopted as a brain model[10].

There are two kinds of popular ANNs used in engineering, such as:

- Multi layer perceptrons(MLPs)[19],
- Self-organizing neural network(SONNs)[18].

Both use "simple" artificial neurons as their building elements. An MLP performs a supervised learning: it receives learning data, each of which includes an input with a correct output, and adjusts its internal parameters in order to correctly map the input to the output. MLPs can form any map between input and output[10]. Owing to this property, many researches from theoretical analysis to engineering applications have been performed[20][21][22][23]. Especially, MLPs are successfully applied to noisy pattern recognition problems[24][25]. However, an MLP does not learn at all unless a supervisor does not exist. Therefore, MLPs cannot be used for the problems that do not have the correct answer for each input. Estimation of a probability density function(PDF) is also one of such problems.

On the other hand, an SONN performs an unsupervised learning: it only receives inputs, and changes its internal parameters by itself in order to reflect stochastic property of the inputs[18]. We discuss SONNs more precisely in the next section.

1.2 Self-Organizing Neural Networks

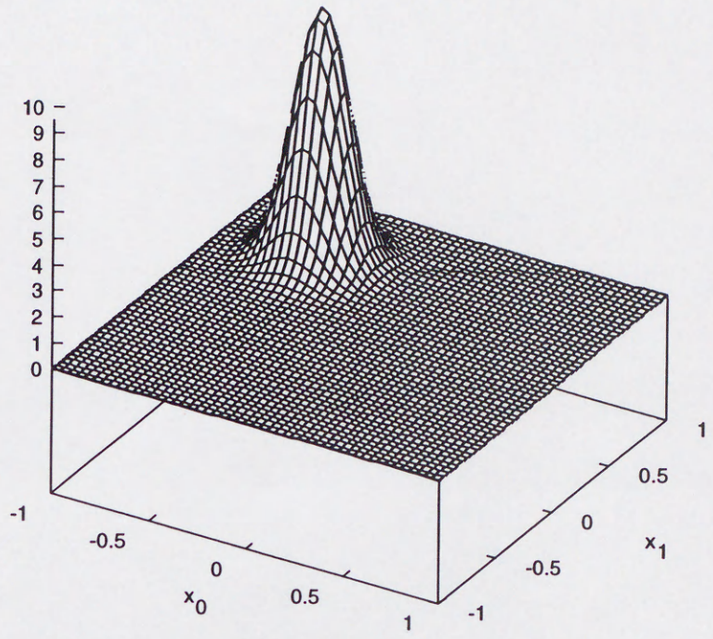
An SONN learns stochastic property from an input sequence. The most popular learning rule used in SONNs is called the Kohonen learning, which is an unsupervised learning rule[18]. It is described in[10] that L.M.Stark, M.Okajima, and G.H.Whipple[26] first invented the original procedure based on the unsupervised learning rule in 1962. After many researchers have studied the rule, T. Kohonen found the most proper goal for the rule, which is to form an equi-probable sample set from input distribution[18]. The followings show a procedure of the Kohonen learning. An SONN has the fixed number of nodes which receive a sequence of inputs generated from an unknown PDF. The inputs are independent each other: an input does not affect the next input. When an SONN receives an input, a competition among nodes is performed in the SONN, and the node fitting to the input best is selected as the winner node. Then the winner node changes its internal parameters to adapt to the input. The learning continues till the SONN satisfies a certain criterion. There are several criteria, for example, thresholding of mean square quantization error(MSE) is a typical criterion.

SONNs are classified into the following two representive classes.

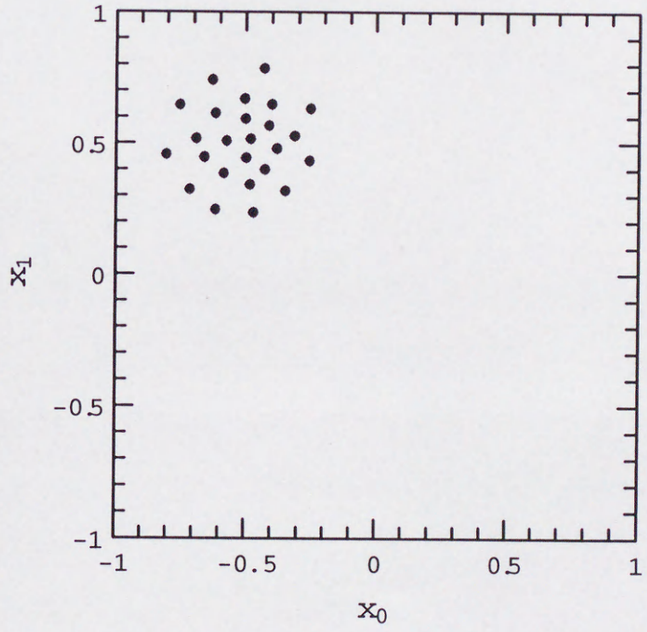
- Neural Vector Quantizers(NVQs)
- Topology Preserving Networks(TPNs)

An NVQ consists of N nodes, each of which indicates a point of input space. An NVQ quantizes a PDF that produces a series of inputs: the NVQ receives the inputs to form node distribution over the input space in proportion to the probability distribution. Figure 1.1 shows an example of a PDF and its quantization created by an NVQ with 64 nodes. The PDF is a 2-dimensional Gaussian centered at the point $(-0.5, 0.5)$. As shown in the node distribution, the node density reflects the probability density. NVQs are mainly applied to data analysis by using their ability to form approximation of PDFs, and lossy data compression by using their ability to form quantized representation of information[30].

A TPN performs quantization of an input PDF like NVQ but it has additional lateral connections(edge) among nodes. Each of the edges determines the neighborhood relationship between two nodes. Nodes and edges of a TPN form a graph G that represent a topological structure.



A probability density function.

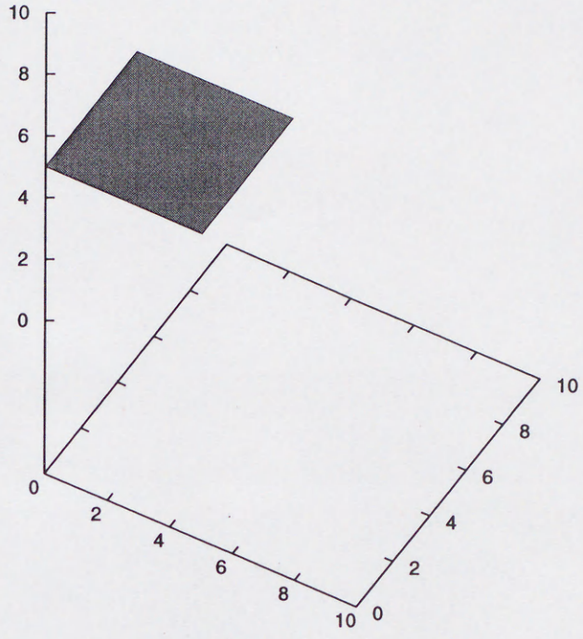


Node distribution.

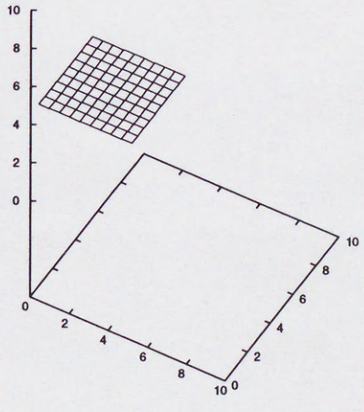
Figure 1.1: An example of a PDF and its quantization using an NVQ.

The TPN receives a series of input vectors $\boldsymbol{x} \in \Re^D$ where the probability of their occurrence obeys some unknown PDF. Then, the TPN forms a map to project regions of the input space onto the nodes of the TPN where adjacent regions are projected onto adjacent nodes, and vice versa. More precisely, the TPN forms a topology preserving map from the manifold $M \subseteq \Re^D$, to graph G, and vice versa. Here, M consists of some regions where probability of input occurrence is non-zero. Figure 1.2 shows an example of a PDF and its topology preserving map created by a TPN with 100 nodes. The PDF has a 2-dimensional uniform distribution in 3-dimensional space. Only connections of the TPN are shown on the map. The manifold of the PDF is a square and the map is correctly formed by the TPN preserving the topology of the manifold. TPNs are mainly applied to data visualization by extracting features from complex data structures, and fault tolerant systems[1] [13].

[1] if a node breaks, the adjacent nodes can be used for its alternatives because the neighboring nodes have similar property



A probability density function.



A map.

Figure 1.2: An example of a PDF and its map using a TPN.

1.3 The Motivation and Objective of This Thesis

Conventional data analysis can be classified into two main groups[10].

- Time series analysis[27]
- Estimation of PDFs[18]

Within the general case of time series analysis, changes in an observed value through time are represented by a stochastic process[28]. Order of observed values is very important in time series analysis because periodicity or self correlation of the values are used in the analysis. Estimation of values observed in future is a main purpose in time series analysis. Targets of time series analysis are, for example, a stock price estimation, transition of population in a city, and so on[29].

Within estimation of PDF, a distribution formed by a group of observed data is represented by an estimated PDF. Different from time series analysis, the observed data are assumed to be independent in the estimation[10]. Therefore, a datum received at a time does not affect the other data received at different time. Targets of estimation of PDFs are, for example, distribution of mean temperature on the earth in a day, human age distribution in a city, and so on. As several methods can be used to estimate a PDF, which is formed by a group of data, using an SONN is known as one of the best methods from the viewpoint of generality and the number of data to be required[18]. Although a large number of studies have been made on SONNs, almost all of the studies have focused on stationary PDFs[13], and nonstationary PDFs are almost ignored. There would be two main reasons for preventing researchers from nonstationary PDFs. One is that dealing with a PDF that may change is difficult because a datum from a nonstationary PDF has little information to estimate the PDF. The other would be a historical one that a set of input data is assumed to be a finite set, which prevents researchers from having the idea to use SONNs in the nonstationary environment. However, we are living in the nonstationary environment, which produces more nonstationary phenomena than stationary ones. Therefore, it is important to study methods to estimate nonstationary PDFs.

In this thesis, "nonstationary phenomena" are classified into two classes.

1. Phenomena that are always changing[29]
2. Phenomena that are almost stationary, but may change[31]

A typical example of a changing phenomenon is transition in stock price distribution. Within such phenomena, it would be very difficult to make a precise approximation of the distribution because it is formed by a large number of data and is always changing. In addition, analysis of such phenomena would be required to reveal dynamics of distribution rather than to know current distribution. In this thesis, we do not focus on such phenomena that are always changing and required to estimate future distribution.

On the other hand, a typical example of the second class of phenomena is distribution of block patterns in an image sequence. In an image compression task, an image is divided into blocks each typically having 4×4(or 8×8) pixels, and the patterns of blocks form some distribution that should be analyzed. In the case of a continuous image sequence, we can consider that each block pattern comes from a nonstationary PDF. Furthermore, the pattern distribution would not frequently change from one image to the next image. Within these phenomena, it is generally important to precisely know current distribution. In addition, we could decompose a nonstationary PDF to some stationary PDFs that makes approximation ease. To simplify the problem, this thesis discusses "nonstationary phenomena" that are almost stationary, but may change.

The objective in this thesis is to establish a basic model of ANNs that extracts stochastic information from nonstationary phenomena. To this end, some methods for SONNs to approximate nonstationary PDFs will be proposed, and evaluated through experiments. Especially the following three problems will be discussed.

- Quantizing nonstationary PDFs
- Forming topology preserving maps from nonstationary PDFs
- Forming memory from nonstationary PDFs

Quantization of nonstationary PDFs is a fundamental problem in approximation of PDFs. A learning method named a Law-of-the-jungle(LOJ) mechanism to quantize nonstationary PDFs is proposed. An SONN using the Kohonen learning with the LOJ mechanism(we call it a LOJ

network) can be applied to problems like motion image compression, which needs adaptive coding under the nonstationary environment.

Forming a topology preserving map from nonstationary PDF is realized by integrating an conventional learning algorithm (the competitive Hebb rule[32]) and the learning algorithm of the LOJ network. The proposed TPN could be applied to problems like feature extraction from motion image, which require an ability to extract a data structure from nonstationary data distribution.

Memory representation of nonstationary PDFs is achieved by using a system consisting of multiple LOJ networks. The system deals with a nonstationary PDF that consists of some stationary PDFs, and can memorize and recall stationary pieces from a nonstationary PDF. The system could be regarded as a biological memory model. Moreover, it could be applied to a problem like process control by taking advantage of an ability to model nonstationary phenomena.

1.4 Thesis Organization

Section 1 is an introduction.

Section 2 discusses an NVQ quantizing nonstationary PDFs. First, we describe the concept of stationary and nonstationary PDFs. Second, we define a nonstationary PDF and discuss some conditions to approximate nonstationary PDFs by using SONNs. Third, we propose the LOJ mechanism, which is a learning method for SONNs to quantize nonstationary PDFs, and evaluate a LOJ network through experiments. Finally, we discuss a method to dynamically evaluate approximation accuracy of a LOJ network during learning.

Section 3 discusses a TPN forming topology preserving maps from nonstationary PDFs. First, we briefly review some related works that include the competitive Hebb rule and the LOJ mechanism. Then we propose a LOJ/CH network and evaluate it through experiments in terms of forming a topology preserving map in the nonstationary environment.

Section 4 discusses an artificial neural system that forms memory from nonstationary PDFs. First, we introduce previous works and point out their limitations. Next, a neural system using multiple LOJ networks is proposed. The system receives a sequence of inputs obeying a nonstationary PDF, and memorizes stationary piece from the nonstationary PDF. Through some experiments, the system ability of memorizing and recalling is evaluated.

Section 5 gives some conclusions and future works.

2 A Neural Vector Quantizer

2.1 Introduction

A Neural vector quantizers(NVQ) can approximate an unknown probability density function(PDF) $p(\boldsymbol{x})$ with a fixed number of nodes[18]. Each node has a weight vector that indicates a point in input space. Within the approximation, an NVQ is required to satisfy certain criteria determining the optimality of node distribution. There are two kinds of representative criteria[33]. One is to minimize the average distortion, which is useful for image or speech coding, and the other is to maximize the entropy of the nodes, which is appropriate for pattern recognition or statistical analysis[10]. The entropy maximization ensures that each of the nodes is used equally in approximating the PDF.

An artificial neural network(ANN) using the Kohonen learning is known as an NVQ and was historically designed for the entropy maximization[10][18]. In the Kohonen learning, a node with the nearest weight vector from an input vector is selected as the winner node through a competition, and the learning is accomplished by moving the weight vector of the winner node toward the input vector. However, a node with the weight vector far from any input may never win, and therefore never learn. From the above reason, the Kohonen learning would not generally form the optimal set of weight vectors, where all the vectors win the competition equally.

DeSieno proposed the conscience mechanism to form the optimal set of weight vectors[34]. Within the conscience mechanism, a certain handicap is given to each node according to their win probability. A node that frequently wins begins to "feel guilty" and prevents itself from winning excessively, and then the other weak nodes can win the competition. As a result, every node can win the competition with the almost equal probability. From this point of view, the conscience mechanism is based on so called the equiprobable principle[33]. The conscience mechanism improves the Kohonen learning to form better approximation with fewer learning steps. However, the node with the lowest win probability must wait until the other nodes win a lot and leave the competition. This causes a slow convergence of networks. Although some

researchers also proposed similar algorithms[33][35], which introduce a certain handicap to the competition according to the win probability, the slow convergence problem remain in those algorithms.

Ueda *et al.* proposed the competitive and selective learning(CSL)[36]. The CSL is based not on the equiprobability principle, but on the equidistortion principle, which minimizes the average distortion. The equidistortion is achieved by the node selection, where useful nodes are multiplied and useless node are deleted. The usefulness of a node is proportional to the average distortion of the node. While the CSL can form an optimal set of weight vectors, it assumes the finite number of inputs.

The conventional algorithms mentioned above are all designed for approximating stationary PDFs. Therefore, these algorithms have some limitations to approximating nonstationary PDFs, which will be discussed in Section 2.3. However, nonstationary environments are more common in the real world. For example, real time video coding and data analysis under nonstationary environments will be promising applications. Therefore, study on approximation of nonstationary PDFs would also be very important.

Section 2.2 gives the concept of stationary and nonstationary PDFs to avoid confusion in terms of time series analysis[27]. Section 2.3 presents a definition of nonstationary PDFs discussed in this thesis and describes some requirements for approximating nonstationary PDFs. Section 2.4 proposes a new learning method named "the Law of the Jungle(LOJ)" and evaluate a SONN with the LOJ mechanism(we call the SONN LOJ network) through experiments. Section 2.5 discusses a method to dynamically estimate approximation accuracy of an LOJ network during learning. Section 2.6 gives conclusions of this section.

2.2 Concept of Stationary and Nonstationary Probability Density Functions

The words "stationary" and "nonstationary" used in this thesis to classify PDFs may be confusing because the same words are also used in time series analysis to classify stochastic process[29]. To make matter worse, in stochastic process, a series of data is considered to be generated by a PDF. Therefore, the usage of the words to classify "PDF" and "stochastic process" should clearly be distinguished. This section firstly gives a brief description of time series analysis and an explanation of the words "stationary" and "nonstationary" in stochastic process. Then a description about approximation of PDFs and concept of stationary and nonstationary PDFs are presented.

2.2.1 Stationary and Nonstationary Stochastic Process in Time Series Analysis

A main purpose of time series analysis is to make changes in (usually one) observed value through time clear. In time series analysis, stochastic process is used to model a series of data. Stochastic process is a mathematical model represented as a set of random variables $\{X(t); t \in T\}$ with a time parameter t[37]. If range T is a set of natural numbers or integers, the process is called stochastic process of discrete time. On the other hand, if T is a set of real numbers, the process is called stochastic process of continuous time. To classify the property of stochastic process, a joint PDF[38] $p(X_{t_1}, \cdots, X_{t_n})$ is introduced by taking the finite number of random variables $(X_{t_1}, \cdots, X_{t_n})$ from the process. The joint PDF is a function that simultaneously generates n data $(X_{t_1}, \cdots, X_{t_n})$ that form a time series. Stochastic process $\{X(t); t \in T\}$ is classified into several classes by determining the property of its joint PDF.

Strongly stationary process shown in Equation 2.1 is stochastic process where a joint PDF of process $\{X(T)\}$ does not change by any translations of time τ.

$$p(X_{t_{1+\tau}}, \cdots, X_{t_{n+\tau}}) = p(X_{t_1}, \cdots, X_{t_n}), -\infty < \tau < \infty \tag{2.1}$$

This condition is very strict and usually useless in time series analysis[28]. Weakly stationary process means more relaxed conditions where the mean value of $\{X(t)\}$ is a constant and the covariance of $X(t)$ and $X(t+\tau)$ has no time parameter t. Weakly stationary process is

commonly used in time series analysis[38].

Nonstationary process is stochastic process that is not stationary process. Nonstationary process is usually translated to modified (weakly) stationary process in order to analyze the process easily. For example, there is a method to analyze differences $\{Y_t; Y_t = X_t - X_{t-1}\}$ of nonstationary process $\{X_t\}$ as weakly stationary process[39].

2.2.2 Stationary and Nonstationary Probability Density Functions

A main purpose of approximating PDFs in this thesis is to clarify the probability distribution formed by a group of inputs. In the PDF approximation, each input is assumed to be generated from an unknown PDF $p(\boldsymbol{X})$. Here, $\boldsymbol{X} = (X_0, X_1, \ldots, X_{k-1})$ is a k-dimensional vector of random variables that **does not** have a time parameter. Joint PDF $p(X_{t_1}, \cdots, X_{t_n})$ in time series analysis simultaneously produces time series of data $(X_{t_1}, \ldots, X_{t_n})$. On the other hand, PDF $p(\boldsymbol{X})$ in the PDF approximation produces spatially data $\boldsymbol{X} = (X_0, X_1, \ldots, X_{k-1})$ at a time. This is the major difference between joint PDF $p(X_{t_1}, \cdots, X_{t_n})$ and PDF $p(\boldsymbol{X})$. This thesis concentrates on the latter PDF.

Stationary PDF $p(\boldsymbol{X})$ discussed in this thesis is a PDF that does not have a time parameter. As a stationary PDF never changes, all the data generated from the PDF have the same stochastic property. In general, nonstationary PDF $p(\boldsymbol{X})$ is a PDF that has a time parameter: nonstationary PDFs may change sometimes. It would be very difficult to deal with a PDF that may change because a datum has little information to estimate the PDF. Therefore, to limit the difficulty of the problem, this thesis defines a nonstationary PDF as a PDF that consists of several stationary PDFs. This approach conceptually assumes that a sufficient number of data to approximate a stationary PDF are obtained by a stationary piece of the nonstationary PDF. The stationary PDFs differ from each other, and only one stationary PDF appears at a time. In this case, one group of data comes from a stationary PDF, and then another group of data comes from another stationary PDF. In other words, stochastic property of data generated from a nonstationary PDF changes at some points of time.

2.3 Approximation of Nonstationary Probability Density Functions

2.3.1 A Definition of Nonstationary Probability Density Functions

The purpose of SONNs discussed in this thesis is to model a phenomenon that occasionally changes its behavior. A series of data observed from the phenomenon makes some distribution, and each of the data is independent. To model such a phenomenon, we define n-dimensional nonstationary PDF $p(\boldsymbol{x}, t); t \in T, \boldsymbol{x} \in \Re^n$ shown in Equation 2.2 where a PDF consisting of m stationary PDFs $p_1(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})$. Taking account of independence of each datum, we consider only the case that time parameter t is discrete.

$$p(\boldsymbol{x}, t) = \begin{cases} p_1(\boldsymbol{x}) & (t = 0, .., t_1 - 1) \\ p_2(\boldsymbol{x}) & (t = t_1, .., t_2 - 1) \\ \ldots & \ldots \\ p_m(\boldsymbol{x}) & (t = t_{m-1}, .., t_m - 1)\ ;\ 0 < t_1 < \ldots < t_m \end{cases} \tag{2.2}$$

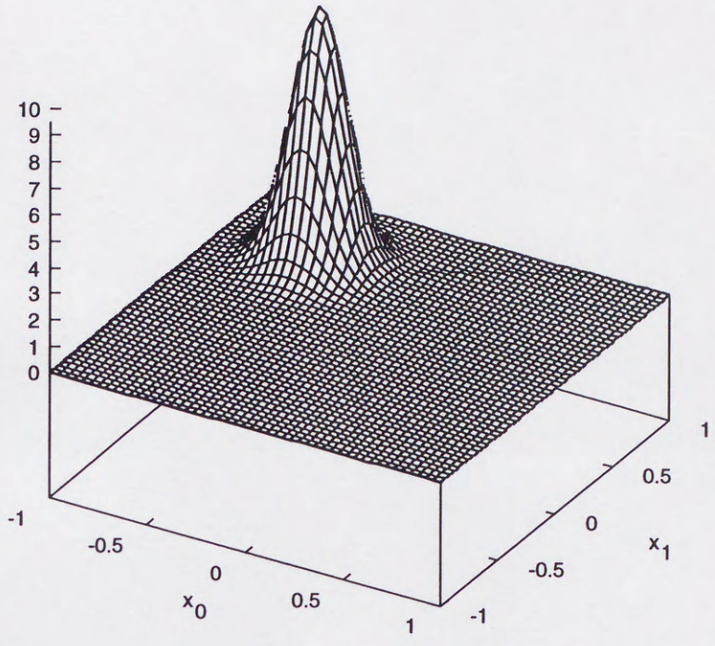
Figure 2.1 and Equation 2.3 show an example of a nonstationary PDF. The nonstationary PDF consists of two 2D-Gaussian stationary PDFs (a) and (b).

$$p(\boldsymbol{x}, t) = \begin{cases} p_1(\boldsymbol{x}) & (t = 0, .., 4999) \\ p_2(\boldsymbol{x}) & (t = 5000, .., 9999) \end{cases} \tag{2.3}$$

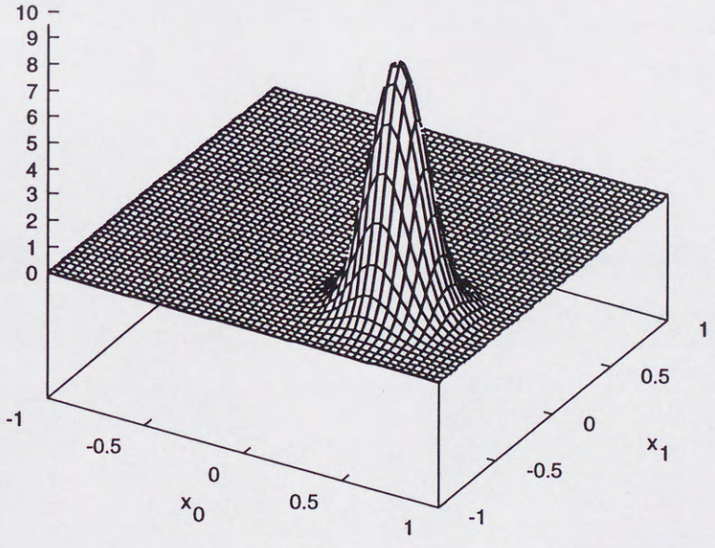
$$\text{(a):}\quad p_1(\boldsymbol{x}) = \frac{1}{2\pi/60} e^{-30\{(x_0+0.5)^2+(x_1-0.5)^2\}}$$

$$\text{(b):}\quad p_2(\boldsymbol{x}) = \frac{1}{2\pi/60} e^{-30\{(x_0-0.5)^2+(x_1+0.5)^2\}}$$

From $t = 0$ to $t = 4999$, PDF (a) appears. Inputs occurs around $(-0.5, 0.5)$. Then the PDF changes to (b) at $t = 5000$, and inputs occurs around $(0.5, -0.5)$.



(a)



(b)

Figure 2.1: An example of nonstationary PDF consists of two stationary PDFs.

2.3.2 Requirements for Approximating Nonstationary Probability Density Functions

To approximate nonstationary PDFs, NVQs should have the following properties, which have never been achieved by conventional NVQs.

1. fast convergence for quick response to changes in PDFs.
2. no time-decaying adaptation parameters.
3. nonstop(online) and nonperiodic adaptation.

The Kohonen learning and the conscience mechanism cannot quickly respond to changes in PDFs. The CSL assumes the finite set of inputs. This assumption determines the number of inputs to be used in a learning cycle, and therefore the selection is periodically performed by halting the network to calculate the distortions. To achieve nonstop and nonperiodic adaptation in the CSL, dynamic estimation of the distortions and dynamic node selection are essential. However, they cannot be achieved as long as the CSL deals with the finite set of inputs, which is a basic assumption. There are several other studies to improve the convergence speed of the Kohonen learning[40][41][42][43][44]. However, none of the networks in their studies is also suitable for approximating nonstationary PDFs, because these networks require decaying parameters which pre-define the total number of learning cycles of the networks, or need periodic adaptation, which have no relation to changes in PDFs.

2.4 A Law-of-the-Jungle Mechanism for Kohonen Learning

2.4.1 A Learning Procedure of the LOJ network

The purpose of the LOJ mechanism is to approximate nonstationary PDFs. The LOJ mechanism is based on the Kohonen learning. The Kohonen learning neither requires time-decaying parameters, nor periodic adaptation. Only the slow convergence problem remains to approximate nonstationary PDFs. This problem is caused by the reason that a node with the weight vector far from any input almost never wins, and therefore rarely learns. This thesis tries to solve the problem by increasing "strong" nodes and decreasing "weak" nodes according to their win probability. A new process is adapted to the Kohonen learning to realize this process. A winner history table is used to estimate a win probability of each node, and creation and deletion of nodes are performed according to the win probability of each node. The creation and deletion is paired and treated as an atomic operation to keep the number of nodes in the network constant. Table 2.1 shows the process of the LOJ mechanism with the Kohonen learning.

Let the number of nodes be N. In Step 1, a small random weight is given to each node. In Step 2, a new input generated according to some (unknown) PDF is presented to the network. In Step 3, distance d_i between the input and the weight vector of each node $i (i = 1, \ldots, \mathrm{N})$ are calculated by using Equation (2.4), and node j that has the shortest distance is selected as the winner node.

$$d_i^2 = \|\boldsymbol{w}_i - \boldsymbol{x}\|^2 \tag{2.4}$$

where $\boldsymbol{x}$ is the input and $\boldsymbol{w}_i$ is the weight vector of node i. Steps 1 through 3 of the LOJ mechanism are standard processes contained in the Kohonen learning.

In Step 4, the winner history table is updated. The winner history table retains up to M entries. If the number of entries in the table is less than M, the entry of winner node j is simply added to the winner history table. If the number of entries is equal to M, the oldest entry is deleted, and then the winner node's entry is added in a FIFO fashion. M is a parameter which decides how long the competition in the past affects the current win probability. In addition, M estimates the amount of change in win probability per competition. M must be selected so

Table 2.1: LOJ Mechanism.

Step 1.	Initialize weight of each node.
Step 2.	Present a new input.
Step 3.	Calculate distance from the input to each node, and then select the winner node.
Step 4.	Adjust the winner history table.
Step 5.	Deduce win probability.
Step 6.	Decide whether the creation and deletion process carried out. If the process is needed, goto Step 7. Otherwise goto Step 8.
Step 7.	Create a new node for the winner node, and delete the weakest node. Adjust the winner history table.
Step 8.	Adapt weight of the winner node. Goto Step 2.

that the random fluctuations in the input sequence do not affect network's behavior.

In Step 5, win probability p_i of each node is deduced from Eq. (2.5).

$$p_i = E_i / E_{all} \tag{2.5}$$

where E_i represents the number of entries of node i in the history table and E_{all} is the total number of entries in the table. In Step 6, whether creation and deletion of the node is performed or not is decided according to the following condition

$$(p_j \geq \mathrm{Cr/N}) \ \vee (\min_k(p_k) \leq \mathrm{De/N}) \wedge (p_j = \max_i(p_i))). \tag{2.6}$$

Equation 2.6 means that

if (the winner node is very strong)

or

(the node with the lowest win probability is very weak and the winner node is the strongest),

then creation and deletion is performed.

Cr is a threshold to create a new node for the winner node with high win probability. When the winner node creates a new node, the win probabilities of the two nodes are roughly divided into halves. In the LOJ, the win probability of each node is dynamically estimated whenever inputs are presented. Therefore, perturbations in the win probability would become quite large. Consequently, Cr should take rather a large value to keep the network stable. From the above reasons, the strength of the "strong winner node" is defined as follows. Even if the probability of the winner node is divided into halves, the winner node should be still stronger than the nodes which have the mean win probability$(1/N)$. Therefore, from the definition, Cr should take a value around 2. On the other hand, De is a threshold to delete the node which has very low win probability. The purpose of De is to delete nodes that exist in region where probability of input occurrence is extremely low; therefore De should take a value which is nearly 0.

In Step 7, creation and deletion of nodes are performed. Node creation is accomplished by dividing winner node j into two nodes j_{new} and j', and also dividing the win probabilities of the new nodes into halves. Fig. 2.2 shows the node creation scheme. In this figure, the first and

the second rows within each node show its win probability and weight, respectively. Since the region to which node j belongs needs more nodes, new nodes j_{new} and j' have the same weight vector as that of node j before division, and they exist in the same position.

After the division, nodes j_{new} and j' share the region that was previously occupied by node j. Based on this reasoning, these two nodes divides the win probability of node j by Eq. (2.7).

$$p_{j_{new}} = \lceil E_{all} p_j / 2 \rceil / E_{all}, p_{j'} = \lfloor E_{all} p_j / 2 \rfloor / E_{all} \tag{2.7}$$

Then, entries of node j in the winner history table are updated with the entries of nodes j_{new} and j' alternately. Deletion of a node is performed by deleting the weakest node k and also deleting the entries of node k from the winner history table.

In Step 8, the weight vector of the winner node is adjusted using Eq. (2.8).

$$\boldsymbol{w}_j^\star = \boldsymbol{w}_j + \alpha(\boldsymbol{x} - \boldsymbol{w}_j) \tag{2.8}$$

where $\boldsymbol{w}_j^\star$ is the weight vector of node j after adjustment, and α represents the learning rate for the weight adjustment. If node creation has been performed, only one node j_{new} moves toward the input. The learning continues by iterating Steps 2 through 8 while inputs are presented.

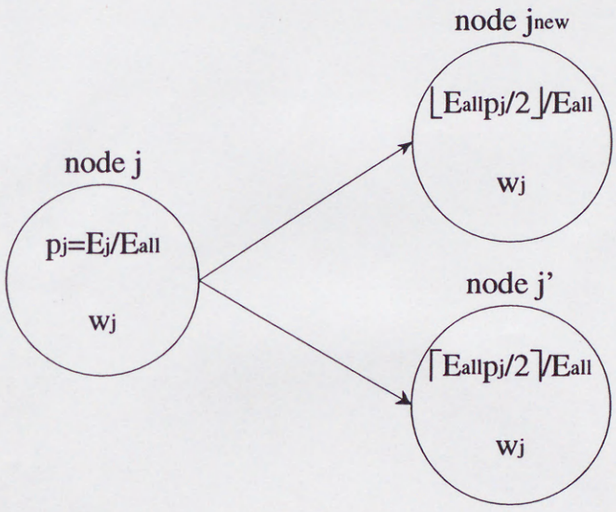


Figure 2.2: Node creation scheme.

2.4.2 Experimental results and Discussion

In this section, three experiments are performed to examine the performance of the LOJ network. Before two main experiments, a preliminary one to demonstrate the convergence speed of the LOJ mechanism is performed.

- **Convergence speed of the LOJ mechanism**

  First, the performance of the LOJ network is compared with that of the neural gas(NG)[44] in terms of convergence speed. The neural gas is known to be a learning method for an NVQ that converges very fast[13]. Within the NG, almost all nodes adapt to an input in the initial stage. Then the number of nodes participating in the adaptation gradually decrease, and the learning ends when no nodes adapt to inputs. While this "group participation" in adaptation has a great effect on fast convergence, the NG has "time-decaying parameters," which make the NG pre-define the total number of learning cycles of the network.

  In this experiment, a 2-dimensional stationary PDF shown in Figure 2.3 is used for input generation. The probability density function consists of several separated clusters, which is typical in practical applications[44]. The probability density is uniform inside the six white squares. On the other hand, the probability density of a shaded square is two times higher than that of white one. The probability density is zero outside the squares. First, the mean square quantization error(MSE) of the LOJ network after 5000 input presentations is measured. Then we examine how many input presentations are needed in the NG network to achieve the same MSE obtained by the LOJ network. The followings show parameters of the LOJ network. The number of nodes and learning ratio are N= 50 and $\alpha = 0.1$, respectively. The other parameters are set as $\mathrm{M} = 1000, \mathrm{Cr} = 2.0$, and $\mathrm{De} = 0.02$. In the case of learning in the LOJN, there is no winner history in the initial stage. In this situation, the win probability of each node is not stable and as a result, this makes network's behavior unstable. We avoid this problem by inhibiting creation and deletion until the history table is filled(from $T = 0$ to $T = 999$). Parameters of NG networks are set according to [44] with having the same number of nodes(N= 50).

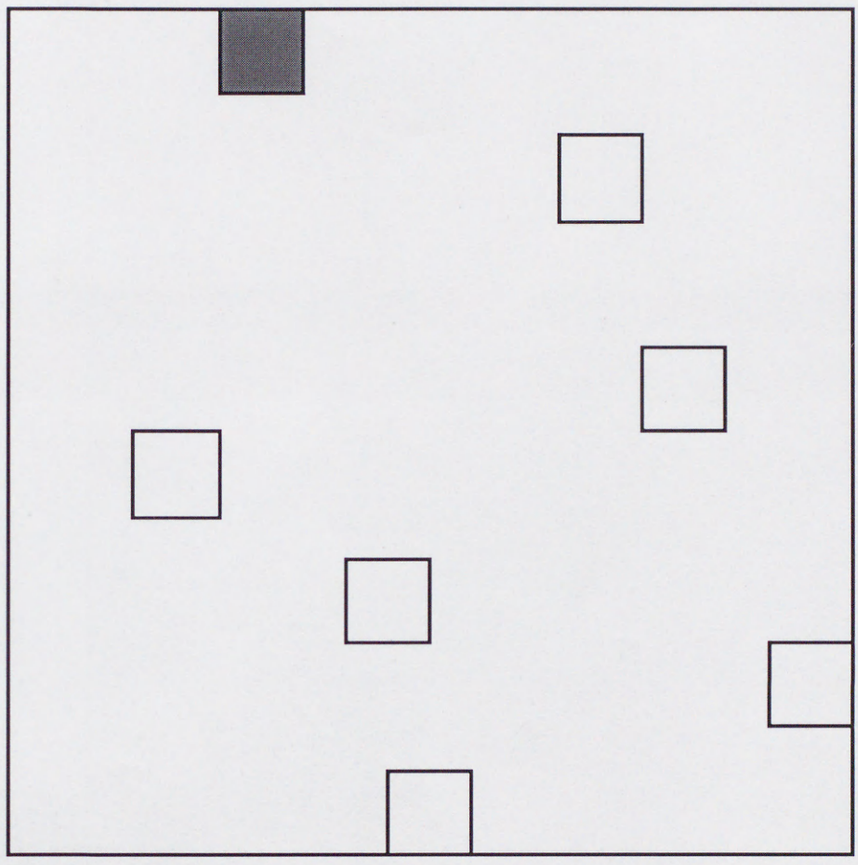


Figure 2.3: The stationary PDF used in experiments.

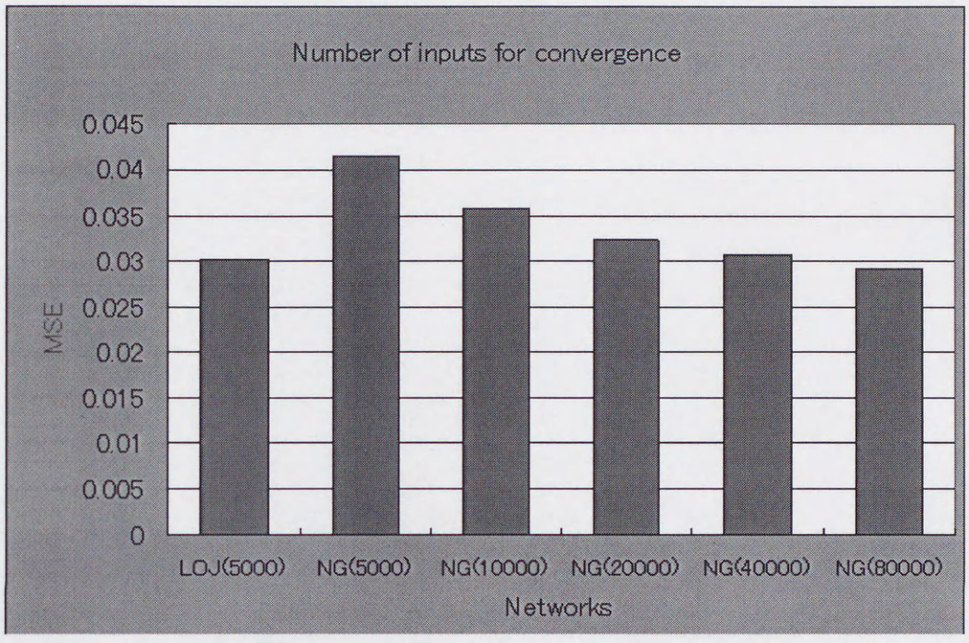


Figure 2.4: MSE vs. presented inputs.

Figure 2.4 shows the relationship between the MSE and the number of input presentations at termination of learnings. The MSEs are the mean values during ten trials. The MSE achieved by the LOJ network after 5000 presentations cannot be achieved by the NG network even after 40000 presentations. Though the NG is believed to converge rapidly, the LOJ needs only 1/10 of input presentations to achieve the same accuracy. This shows that creation and deletion of nodes in the LOJ network is quite effective for fast approximation of PDFs.

- **Approximation of Nonstationary Probability Density Functions**

  In the remains of this section, two experiments approximating nonstationary PDFs is performed to examine the performance of the LOJ mechanism. Figure 2.5 shows the PDFs used in the experiments. In the "simple" problem, the probability density is uniform within the bottom left hand square from $T = 0$ to $T = 19999$ and after $T = 20000$ input region moves to the top right hand square with the same density. Here, T is the number of inputs presented. The probability density is zero outside the squares. "Complex" is more practical where the probability distribution consists of several separated Gaussian clusters. In this problem, each Gaussian distribution corresponds to one cluster. The center and the radius of each circle represents the mean and the cluster boundary($3 \times \sigma$) of the Gaussian, respectively. Here, σ stands for the standard deviation of the Gaussian. The probability of input occurrence in the six white circles is the same. On the other hand, the occurrence probability within a shaded circle is two times higher than that within white one. The PDF also changes after $T = 20000$. An ideal noiseless environment and two kinds of noisy environments have been prepared for each problem.

  In the experiments, three kinds of Kohonen learning networks(KLNs) were investigated. They were a normal KLN(NKLN), a KLN with the conscience mechanism(CON), and a LOJ network(LOJN). However, the NKLN was used only under noiseless environments for comparison, because approximation accuracy of the NKLN was relatively very low. The weights of each KLN were initialized with small random values. The followings show parameters for the KLNs. The number of nodes and learning ratio in the all KLNs were the same, where N= 64 and $\alpha = 0.1$. The CON takes two more parameters β and γ. β estimates the amount of change in win probability and corresponds to the inverse of the maximum number of entries M in the history table in the LOJN. In this experiment, we set the parameter $\beta = \mathrm{M}^{-1} = 0.0008$ to equalize the amount of change in win probability between the LOJN and the CON. We set $\gamma = 10$ which decides the amount of handicap in the competition[34]. The parameters related to the LOJN were set as $\mathrm{M} = 1250, \mathrm{Cr} = 2.0$, and $\mathrm{De} = 0.02$. From $T = 0$ to $T = 1249$, creation and deletion of the LOJN is inhibited.

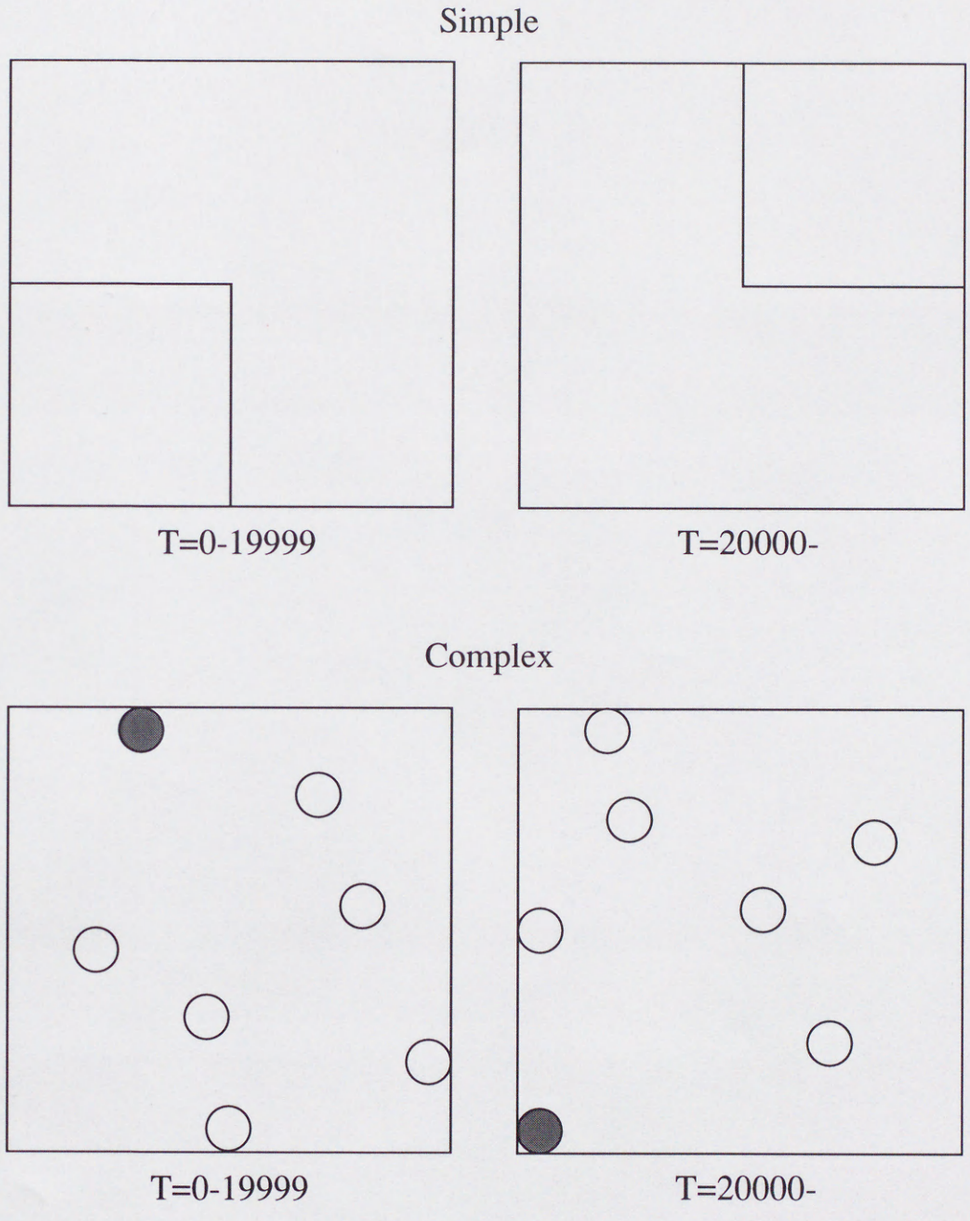


Figure 2.5: Nonstationary PDFs used in experiments.

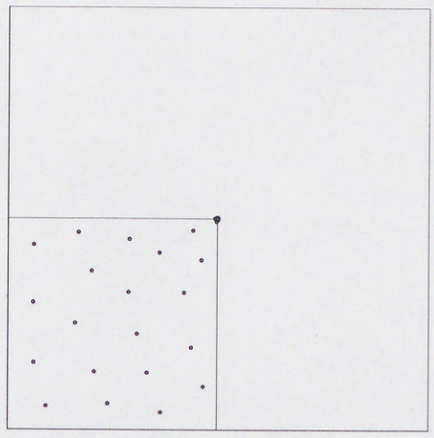
As a result, the LOJN works the same as the NKLN during this period.

1. **Noiseless Environment**

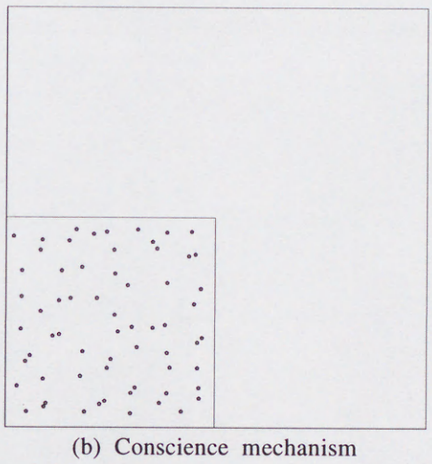
   We compare three networks from three points of views; the distribution of weight vectors, the standard deviation of win probability(SDW) and the MSE. The SDW is a metric of the equiprobability, and the MSE is a metric of the average distortion. The SDWs and the MSEs are the mean values during ten trials.

   First, we show the results of the simple problem. Figure 2.6 shows typical distributions of weight vectors at $T = 19999$ just before the PDF changes at this time, where about one third of nodes win in the case of the NKLN, and the other nodes are left at their initial positions. In the CON, all nodes move to the region where inputs occur, but the nodes have not been uniformly distributed yet. On the other hand, in the LOJN, all nodes move to the input region and are already distributed uniformly. Figure 2.7 shows the distribution of weight vectors at $T = 25000$, where 5000 presentations are performed after the PDF changes. In the NKLN, the nodes which win before the PDF changes are left in the old input region, and 13 nodes move to the new input region. In the CON, a number of nodes that are around their initial positions move to the new input region, because these nodes cannot win many times before the PDF changes and their win probabilities are satisfactory small. However, the nodes which win many times before the PDF changes will not be able to win until their win probabilities decreases. In the LOJN, no nodes remain in the old input region, and in addition, homogeneous distribution is already formed in the new input region.

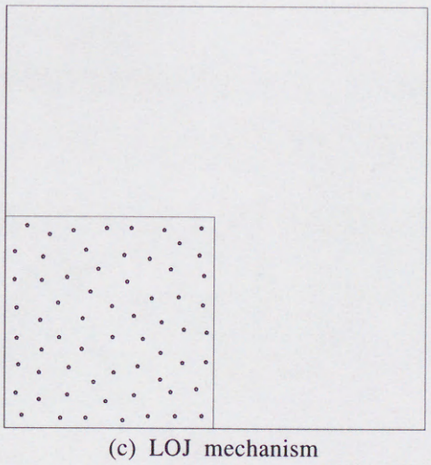
   Figures 2.8 and 2.9 show the SDW and the MSE as a function of the number of presentations. Both Figures show a similar tendency in the case of the simple problem. From $T = 0$ to 500, the SDW and the MSE of all networks decrease steeply because a few nodes move to the center of the input region. Before $T = 1250$, the behavior of the LOJN is similar to that of the NKLN, and their performance is inferior to that of the CON. The SDW and the MSE of the LOJN suddenly decrease



(a) Kohonen learning

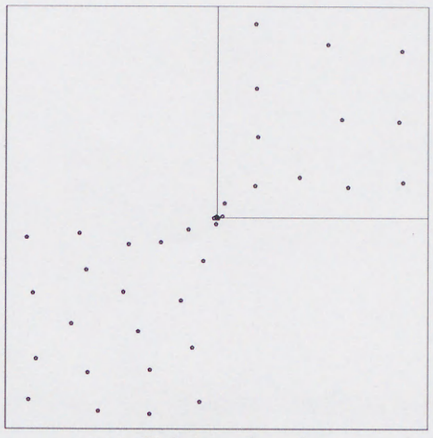


(b) Conscience mechanism

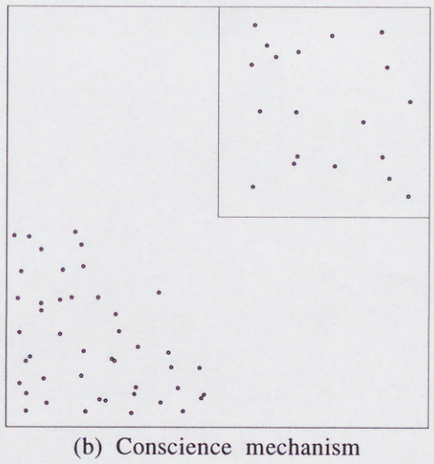


(c) LOJ mechanism

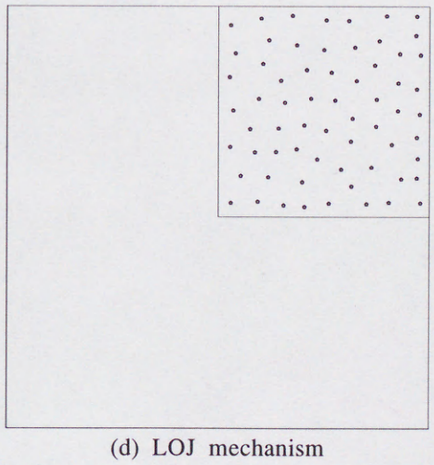
Figure 2.6: Weights distributions of the simple problem at T=19999.



(a) Kohonen learning



(b) Conscience mechanism



(d) LOJ mechanism

Figure 2.7: Weights distributions of the simple problem at T=25000.

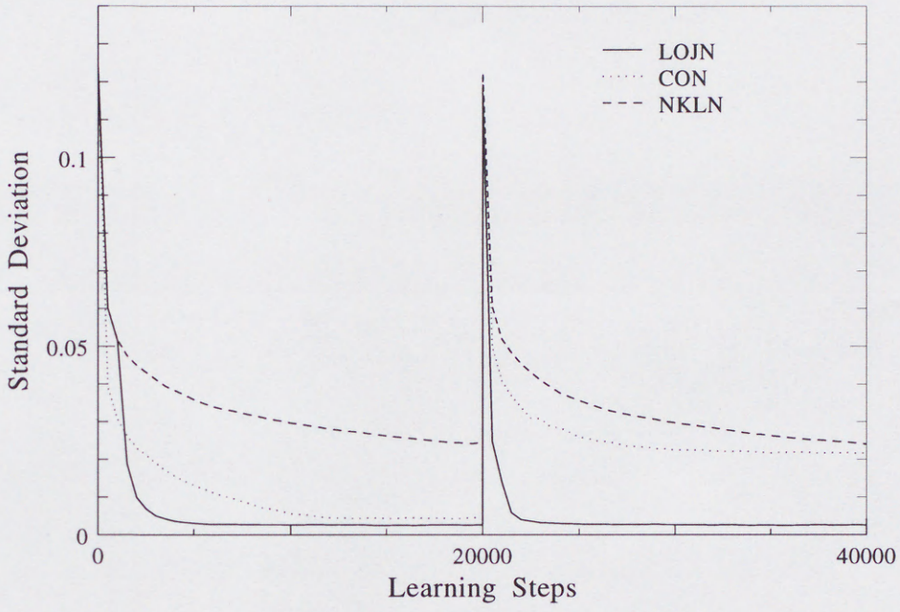


Figure 2.8: Standard deviation of win probability vs. number of learning steps of the simple problem.

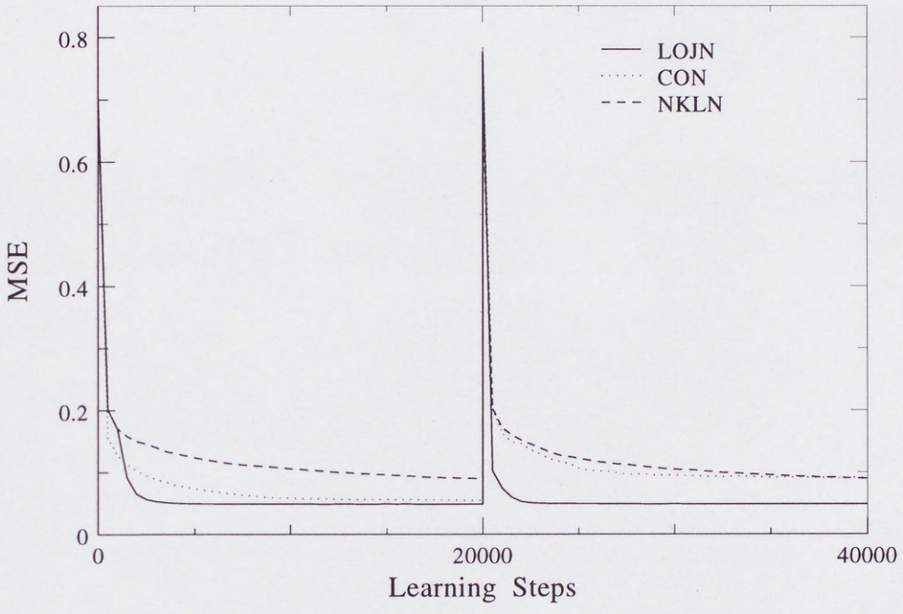


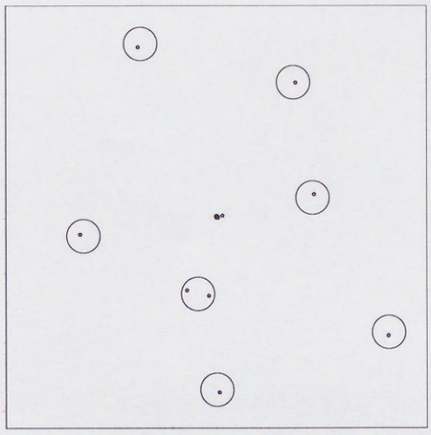
Figure 2.9: MSE vs. number of learning steps of the simple problem.

from $T = 1250$, which shows the effectiveness of creation and deletion. The LOJN already converges at around $T = 4000$. At $T = 20000$, the PDF changes, and the SDWs and the MSEs increase discontinuously. The MSE of the LOJN is the biggest at that time. This is because the LOJN adapts to the old PDF better than the other networks. At $T = 20500$, the LOJN already has smaller values of the SDW and the MSE than any other network, which means the rapid adaptation of the LOJN to the new environment. The LOJN converges at around $T = 23000$. The performance of the CON after $T = 5000$ is inferior to that before $T = 5000$, since the results of competitions in $T < 5000$ affect the behavior of the CON after $T = 5000$.

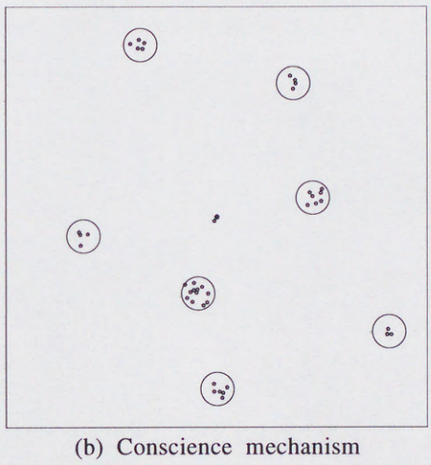
Second, we show the results of the complex problem. Figure 2.10 shows typical distributions of weight vectors at $T = 19999$. Only eight nodes win in the case of the NKLN, and the other nodes are left at their initial positions. In the CON, unlike the case of the simple problem, about one third of nodes cannot win the competitions and remain at their initial positions. On the other hand, in the LOJN, all nodes move to the input region and are already form the Gaussian distribution with reflecting the probability density. Figure 2.11 shows the distribution of weight vectors at $T = 25000$. In the NKLN, no node exists inside two top left clusters and a few nodes remain in the old input regions. In the CON, a lot of nodes move into the cluster close to their initial positions. Besides, several nodes remain in the old input regions. In the LOJN, no nodes are in the old input regions, and almost correct distribution is formed in the new input regions.

Figures 2.12 and 2.13 show the SDW and the MSE as a function of the number of presentations. In the complex problem, the performance disparity between the LOJN and the other KLNs clearly appears in comparison with the case of the simple problem.

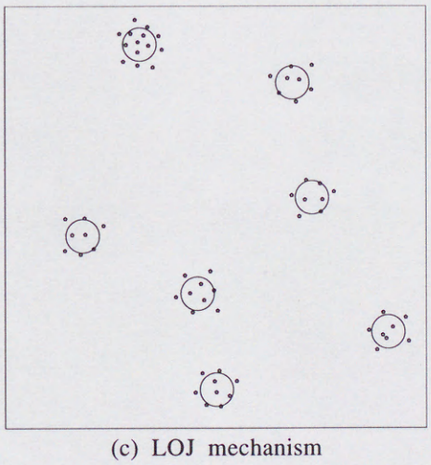
The LOJN requires about 4000 input presentations for adaptation to the new environment, whose behavior is almost the same as the case of the simple problem. On the contrary, the other KLNs show lack of adaptation capability. The performance



(a) Kohonen learning



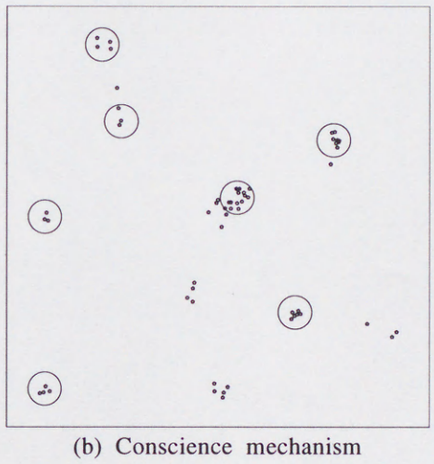
(b) Conscience mechanism



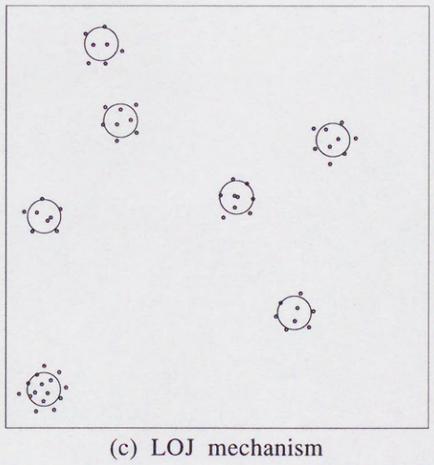
(c) LOJ mechanism

Figure 2.10: Weights distributions of the complex problem at T=19999.

(a) Kohonen learning



(b) Conscience mechanism



(c) LOJ mechanism

Figure 2.11: Weights distributions of the complex problem at T=25000.

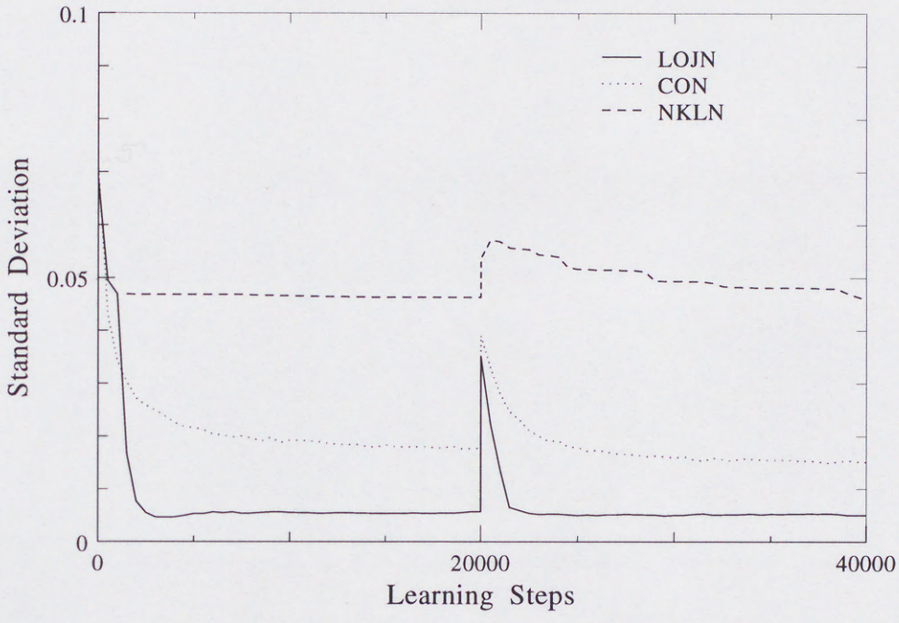


Figure 2.12: Standard deviation of win probability vs. learning steps of the complex problem.

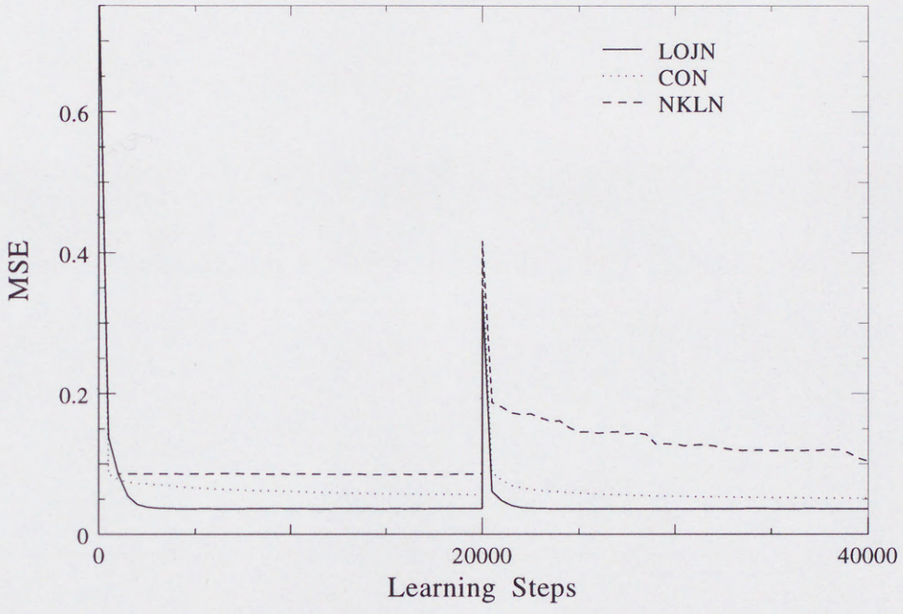


Figure 2.13: MSE vs. learning steps of the complex problem.

difference of the CON between the cases of $T \geq 20000$ and $T < 20000$ is not observed. This is due to the insufficient adaptation of the CON before the PDF changes.

2. **Noisy Environment**

In real applications, input data usually contain noise. Therefore, it is important to evaluate performance under some noisy environments. The CON and the LOJN are evaluated under two kinds of environments; they contain low white noise (S/N of power = 10), and high white noise (S/N of power = 2), respectively. We perform ten trials for each experiment, and the mean values of the SDWs and the MSEs are used for performance measurement.

First, we discuss the simple problem. Figures 2.14 and 2.15 show the SDW, and the MSE under the noisy environment as a function of the number of presentations. In the LOJN, the S/N ratio affects the convergence of the MSE and the SDW of each network, but does not affect convergence speed so much. The LOJN is stable even under noisy environments and its behavior does not differ from that under noiseless environments. In comparison with the CON under the same environment, the convergence of the MSEs in the LOJN is always achieved earlier, and the smallest MSEs are always obtained.

Next, we discuss the complex problem. Figures 2.16 and 2.17 show the SDW and the MSE under the noisy environment as a function of the number of presentations. The results differ from those of the simple problem. In the CON, the S/N ratio seems to slightly affect the SDWs and the MSEs. On the other hand, in the LOJN, the high S/N ratio badly affects the performance of the network, and the SDW is unstable especially in the high noise environment. In other words, in the case of low-noise environments, the LOJN is superior to CON, but under high-noise environments, the situation is reversed. These results show the superiority of the conscience mechanism under noisy environments. In the complex problem, probability of input occurrence is concentrated on the several small clusters. Therefore, the probability density of each cluster is extremely high. In the conscience mechanism, nodes with high win probability rarely participate in competitions, so they move very little. As a result, the nodes in the clusters are influenced little by noise. On the contrary, in the LOJN,

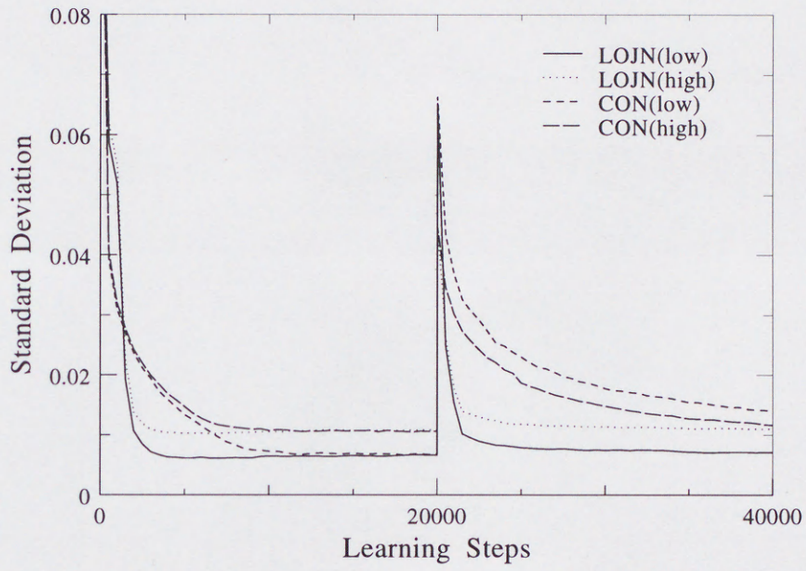


Figure 2.14: Standard deviation of win probability vs. learning steps under noisy environments (simple problem).

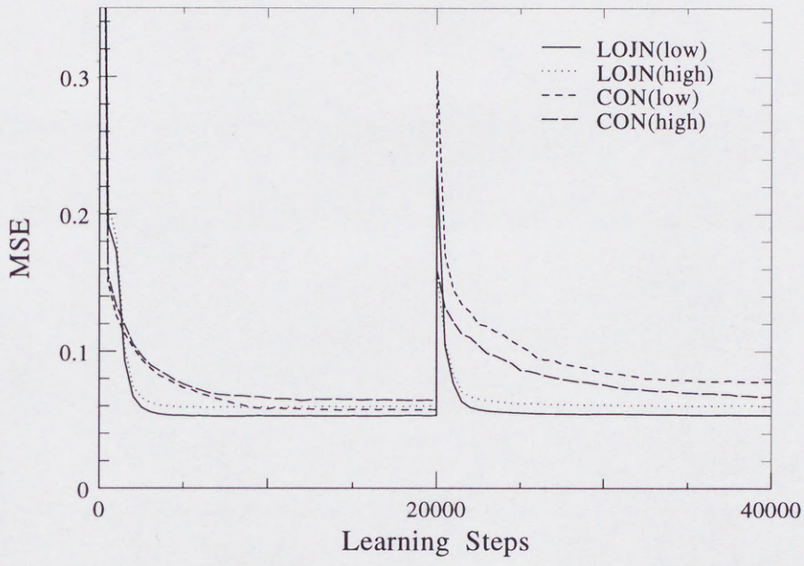


Figure 2.15: MSE vs. learning steps under noisy environments (simple problem).

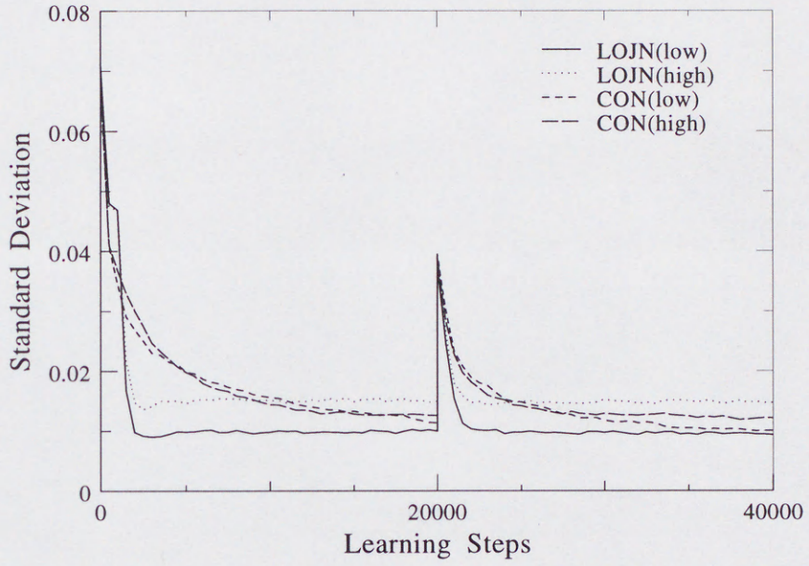


Figure 2.16: Standard deviation of win probability vs. learning steps under noisy environments (complex problem).

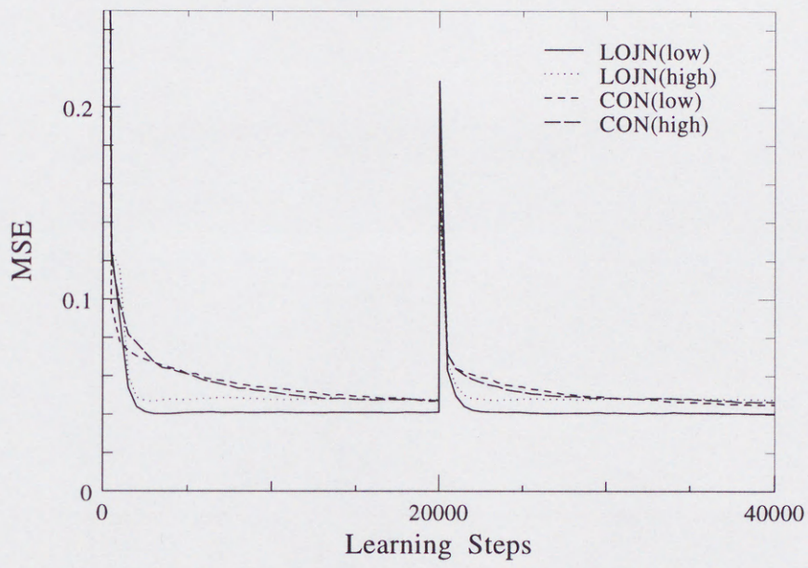


Figure 2.17: MSE vs. learning steps under noisy environments (complex problem).

there is no mechanism that inhibits strong nodes from participating in competitions, and therefore the LOJN is severely affected by noise. These suggest that LOJN is relatively not suitable for cluster separation problems under very noisy environments.

2.5 Dynamic Analysis of Convergence in the LOJ Network

As shown in the previous section, an LOJ network can approximate an unknown nonstationary PDF $p(\boldsymbol{x}, t)$. Approximation accuracy of the network is usually given by measuring the standard deviation or the variance of the win probability. However, when the standard deviation of the win probability(SDW) is measured, an LOJ network is required to stop its learning to calculate the win probability of each node with receiving test data generated by an "unknown" PDF. It would be required for many applications to evaluate the approximation accuracy in the LOJ network without stopping its learning. Therefore, this section discusses methods to evaluate approximation accuracy of the LOJ network during learning.

2.5.1 A Method of the On-line Decision

The LOJ network has the winner history table to decide necessity of creation or deletion of its nodes. In the history table, indices of winner nodes in the past M competitions are stored to estimate the win probability of each node. In this thesis win probability calculated during learning by using the winner history table is called "the internal win probability," and win probability calculated in the ordinary way by using test data is called "true win probability."

Under assumptions that inputs are produced by one of stationary PDFs forming a nonstationary PDF, the following two relationships as shown in Equations 2.9 and 2.10 hold[13].

$$\begin{array}{c}\text{(The network converges to (local) minimum)} \\ \Updownarrow \\ \text{(The mean position of each node is constant)}\end{array} \tag{2.9}$$

$$\begin{array}{c}\text{(The network converges to (local) minimum)} \\ \Updownarrow \\ \text{(The standard deviation of the true win probability)} = 0\end{array} \tag{2.10}$$

These two equations lead to a relationship shown in Equation 2.11

$$\begin{array}{c}\text{(The network converges to (local) minimum)} \\ \Updownarrow \\ \text{(The mean position of each node is constant)\&} \\ \text{(The standard deviation of the true win probability)} = 0\end{array} \tag{2.11}$$

Assuming that the number of entries in the winner history table $= \infty$, the following relationship

holds.

$$\begin{gathered}\text{(The mean position of each node is constant) \&} \\ \text{(The standard deviation of the true win probability)} = 0 \\ \Updownarrow \\ \text{(The standard deviation of the internal win probability)} = 0\end{gathered} \tag{2.12}$$

As a result, Equation 2.9 and 2.10 lead to,

$$\begin{gathered}\text{(The network converges to (local) minimum)} \\ \Updownarrow \\ \text{(The standard deviation of the internal win probability)} = 0\end{gathered} \tag{2.13}$$

As this thesis deals with a nonstationary PDF that consists of two or more stationary PDFs, it is assumed that the relationship in Equation 2.13 is approximately held in a group of data generated from a stationary pieces of the nonstationary PDF. From this assumption, this thesis proposes using standard deviation of the internal win probability(called internal SDW) to estimate approximation accuracy of the LOJ network during learning.

2.5.2 Experimental Results and Discussion

This section compares the internal SDW with the true SDW to examine the internal SDW for an estimation of an approximation accuracy of the LOJ network. Figure 2.18 shows a 2-dimensional nonstationary PDF used in this experiment. The nonstationary PDF consists of three stationary PDFs and changes at T =10001 and 20001. Here, T is the number of inputs presented. In the first piece of the nonstationary PDF, the probability density is uniform within the bottom left hand square. The second and the last pieces consist of seven separated clusters. The probability density is uniform inside each of the six white squares. On the other hand, the probability density of a shaded square is two times higher than that of white one. The probability density outside the squares is zero. The LOJ network has 50 nodes, the history table with 1250 entries, node creation parameter Cr= 2.0, and node deletion parameter De= 0.05. No creation and deletion are performed until the history table is filled(from $T = 0$ to $T = 1249$).

Figure 2.19 shows the internal and true SDWs as a function of the number of input presentation. The SDWs are the mean values during ten trials. Both graphs roughly have the similar tendency: SDWs rapidly increase just after changes in PDF and then gradually decrease. The internal SDW does not exceed a certain value except the initial stage because creations and deletions of nodes do not allow the existence of nodes that have too high or too low win probabilities. Taking a precise look at the graphs, there are some differences between them. Only the internal SDW has a minimal value after the PDF changes. Creations and Deletions of nodes frequently occur just after the PDF changes. Within the process of the node creation, the winner node creates a new node and gives the half of its win probability to the new node. However, in practical cases, the win probabilities of two nodes are not exactly the same because the probability of input occurrence in the space occupied by the two nodes is not always uniform. This difference make the internal SDW take excessively lower values than the true SDW. After the minimal values, the internal SDW slightly increases with correcting the difference during learning. Existence of minimal values seems to make it difficult to use the internal SDW for estimating the true SDW. However, this happens in the only case where node creations frequently occur when the network does not sufficiently approximate the PDF.

Figure 2.20 shows the relationships between frequency of node creations and the internal and true SDWs over the input presentations. The figure clearly illustrates that frequent creations cause a large difference between the internal and the true SDW. After about 1200 input presentations, which is approximately equal to the size of the winner history table M, the harmful effect of the frequent creations and deletions disappears. This makes the tendency of the internal SDW coincide with the true SDW. As a result, with calculating the internal SDW after M presentations from the last frequent creations, the internal SDW can be used as a measure of the approximation accuracy of the LOJ network.

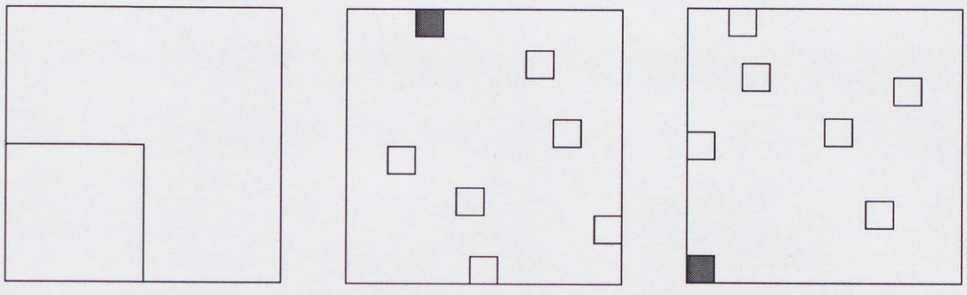


Figure 2.18: A nonstationary PDF used in the experiment.

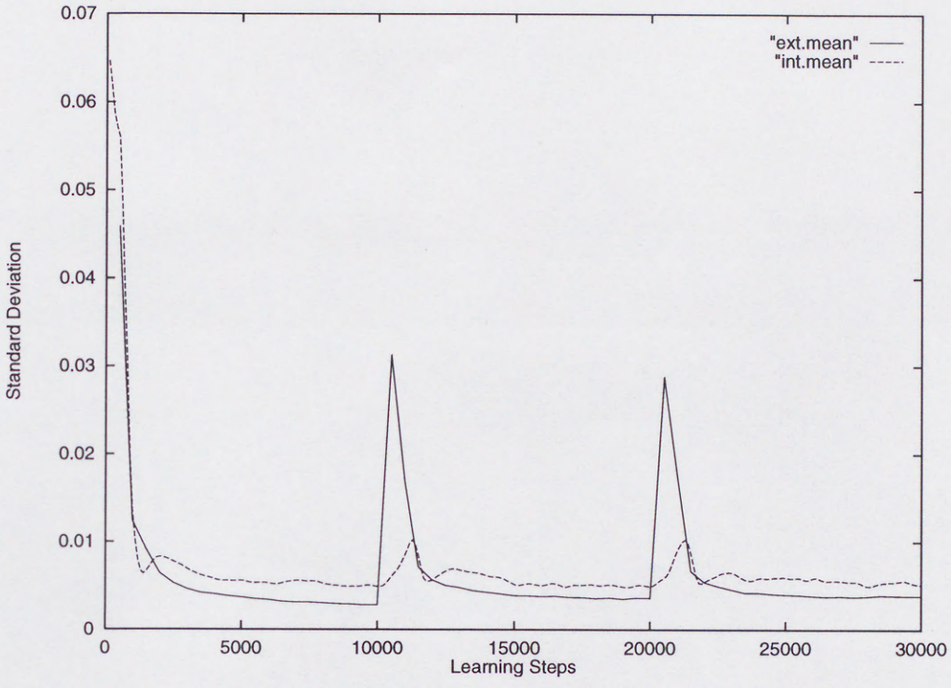


Figure 2.19: Standard deviations vs. learning steps.

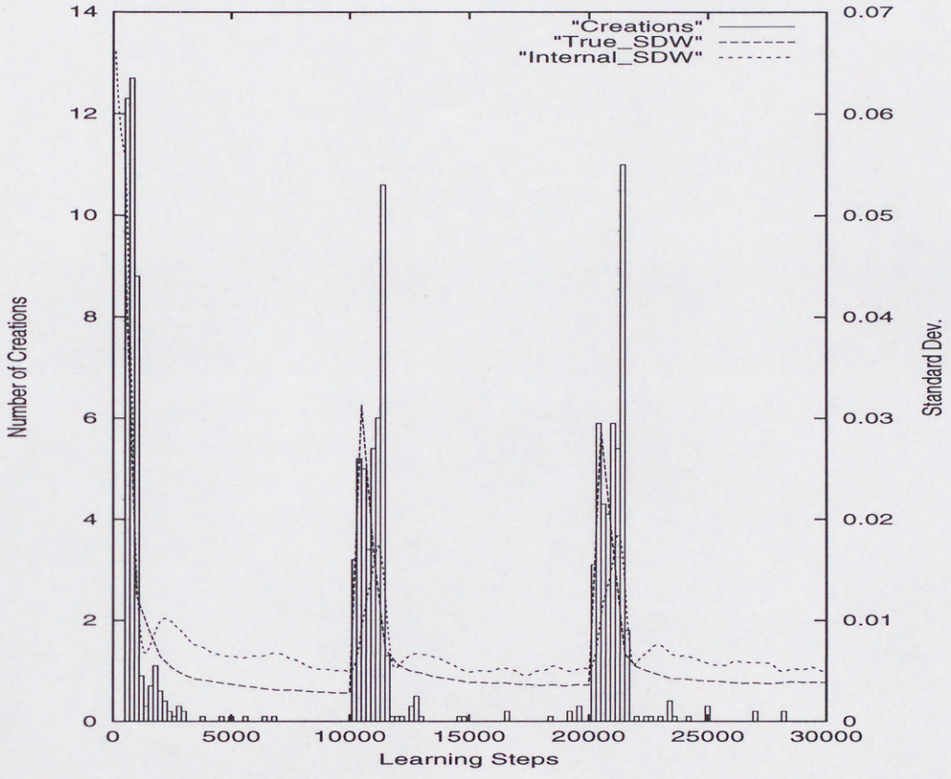


Figure 2.20: Standard deviation and number of creations vs. learning steps.

2.6 Conclusions

This section discusses an NVQ quantizing nonstationary PDFs. First, concept of stationary and nonstationary PDFs has been described. Second, a definition of a nonstationary PDF has been clarified, and some conditions to approximate nonstationary PDFs have been discussed. Third, the LOJ mechanism has been proposed, which is a learning method to quantize nonstationary PDFs. In the LOJ mechanism, the nodes which cannot win the competition are transferred directly from the region where inputs almost never occur to the region where inputs often occur. Therefore, the direct "jump" of weak nodes toward the region where inputs occurrence probability is higher in the LOJ will be effective for the fast convergence. In addition, the LOJ requires neither time-decaying parameters nor a constant cycle for network adaptation. These properties are suitable for approximating nonstationary probability distribution functions. By comparing the LOJ mechanism with the other Kohonen learning mechanisms through experiments, the networks with the LOJ mechanism have always shown a better performance for the approximation of nonstationary PDFs except for a complex problem under high noise environments.

Finally, we discuss a method to dynamically estimate the approximation accuracy of a LOJ network during learning. Experimental results have suggested that, excepting the period where the harmful influence of frequent node creations does not exist, the internal SDW can be used to dynamically measure the approximation accuracy of the LOJ network.

3 A Topology Preserving Neural Network

3.1 Introduction

A topology preserving map plays an important role in neural processing systems, and a wide variety of studies from theory to application have been performed[13]. Usually, an artificial neural network to form topology preserving maps(TPN) consists of nodes i, $i \in \{1, \ldots, N\}$ with lateral connections, each of which determines the neighborhood relationship between two nodes. These nodes and their connections form a graph G to represent a topological structure. Each node has a synaptic weight vector $\boldsymbol{w}_i \in \Re^D$ to indicate a point of a D-dimensional input space.

The TPN receives a series of input vectors $\boldsymbol{x} \in \Re^D$ where the probability of their occurrence obeys some unknown probability density function(PDF). Then, the TPN forms a map to project input vectors onto the nodes of the TPN where input vectors on the adjacent input regions are projected onto adjacent nodes, and vice versa. More precisely, the TPN forms a topology preserving map from the manifold $M \subseteq \Re^D$, to the graph G, and vice versa. Here, M consists of several regions where probability of input occurrence is non-zero.

Kohonen's self-organizing map(SOM) algorithm is the most popular learning algorithm to form a topology preserving map[18]. However, in the case where the topology of the SOM network is different from that of an input manifold, the SOM cannot generally form a topology preserving map because the SOM network keeps a fixed topology during learning.

Martinetz proposed the Competitive Hebbian learning rule(CH)[32] to form a perfectly topology-preserving map. The CH assumes that the density of weight vectors of an SONN is in proportion to the probability density of input occurrence over an input space. Thus, the ANN is considered a neural vector quantizer(NVQ) of an unknown PDF. For a given unknown PDF, a representation of the PDF using an NVQ is obtained first, and then the CH is used to generate an appropriate topology. Therefore, the CH and the NVQ are processed separately. For fast topology preservation learning, it would be effective to simultaneously apply both the CH and the NVQ to the formation of a topology preserving map. Several works are proposed

considering the combination of a neural NVQ and the CH[45][46][47], and successfully make TPNs form topology preserving maps during the approximations of PDFs.

However, all of the neural NVQs used in TPNs are not suitable for approximating nonstationary PDFs because their NVQs require time-decaying parameters that pre-define the total number of learning cycles, or need periodic adaptations that have no relation to changes in PDFs. Hence, the lack of ability for approximating nonstationary PDFs prevents their ANNs from forming topology preserving maps in nonstationary environments.

As nonstationary environments are more common than stationary environments in the real world, it is important for TPNs to form a map from nonstationary PDFs. Promising applications of TPNs for nonstationary PDFs would be real time feature extraction from video sequences, data analysis in nonstationary environments, and so on.

This section discusses the possibility of integration of the LOJ and the CH, and proposes an algorithm for ANNs that can form a topology preserving map from a manifold M whose topology may change. Section 3.2 briefly mentions the foundation of the CH, and discusses the possibility of integration of the LOJ and the CH. Section 3.3 proposes an algorithm for ANNs that can form a topology preserving map from a manifold M whose topology may change. Section 3.4 gives the experimental results and discussion. Section 3.5 gives conclusions of this section.

3.2 A Topology Preserving Network in the Nonstationary Environments

3.2.1 The Competitive Hebbian Learning Rule

The CH allows a neural NVQ to form a perfectly topology preserving map from an input manifold M. The CH assumes that an ANN consists of nodes $i \in \{1, \ldots, N\}$ with weight vectors $\boldsymbol{w}_i \in \Re^D$. It also assumes that the weight vectors are distributed to approximate the input manifold M, for any $\boldsymbol{w}_i \in M$. The CH successively inserts topological connections between nodes by using the following way:

> For each input $\boldsymbol{x}$, connect two nodes which have the nearest and second nearest weights measured by Euclidean distance.

This procedure is based on the idea that nodes with weight vectors having neighboring *masked Voronoi polygons* are connected[32]. Martinetz showed that the resulting graph G of the nodes with the lateral connections forms "induced Delaunay triangulation" of the weight vectors $\boldsymbol{w}_i$, and hence the CH can perfectly construct topology preserving maps of arbitrarily structured input manifolds[32]. There are several methods to simultaneously use the CH and the NVQ to achieve a fast topology learning as mentioned in the previous section. In these cases, they have to use techniques for removing obsolete lateral connections between nodes since the movement of the weight vectors in the NVQ may make the connections invalid.

Though these methods with the combination of the CH and the NVQs can form topology preserving maps during approximations of input PDFs, all of the NVQs are not suitable for approximating nonstationary PDFs, because these NVQs require time-decaying parameters that pre-define the total number of learning cycles, or need periodic adaptations that have no relation to changes in PDFs. Hence, the lack of ability for approximating nonstationary PDFs prevents their ANNs from forming topology preserving maps in nonstationary environments.

3.2.2 Integration of the LOJ and the CH

The LOJ is an NVQ to approximate nonstationary PDFs as shown in Section 2. The LOJ needs only 1/10 of input presentations to approximate a PDF compared with the neural gas[44], which is known to be an NVQ that converges very fast[13]. Moreover, the LOJ requires neither time-decaying parameters nor special periodic adaptations, where these properties are essential for dealing with nonstationary PDFs. Like ordinary NVQs, since the LOJ has no topological relationships among nodes, the ANN using the LOJ cannot form a topology preserving map. However, simultaneous use of the LOJ and the CH may lead us to a new ANN that can form topology preserving maps from nonstationary PDFs, which cannot be achieved by the conventional NVQs with the CH.

To consider the integration of the LOJ and the CH, we must discuss problems from two kinds of viewpoints:

1. Problems caused by the CH in the process of the LOJ to approximate a nonstationary PDF.
2. Problems caused by the LOJ in the process of the CH to form topological connections between nodes.

The CH only makes connections between nodes, which does not disturb the movement of weight vectors. Moreover, the CH requires neither time-decaying parameters nor special periodic adaptations, which both prevent the ANNs from dealing with nonstationary PDFs. As a result, the CH has no problem in the integration with the LOJ.

The problems of the LOJ is the movement of weight vectors, because the movement of weight vectors makes the topological connections between nodes invalid. Therefore, a method to eliminate obsolete connections caused by the movement of weight vectors should be discussed. Within the LOJ, weight vectors could move in two different ways. One is due to sequential adaptation toward inputs, which is also required in most NVQs. However, obsolete connections caused by the sequential adaptation can be eliminated with the *edge aging* scheme[45]. The other is due to the direct jumps of the weight vectors of weak nodes toward the weight vectors of strong nodes, which are peculiar to the LOJ. These direct jumps usually move the weights

drastically on the manifold. As a result, the jumps make topological relationships among nodes invalid. Therefore, the connections between a node and its neighbors should be removed when the node jumps. This removal may make topology learning slow down, but fortunately, the jumps would be performed only in the initial adaptation stage where win probabilities of nodes are quite different from each other. Therefore, this slowdown caused by the removal could be ignored.

3.3 LOJ/CH Network

This section describes a learning algorithm for an ANN to form a topology preserving map between a network G and a manifold M whose topology may change during learning. The algorithm is called the LOJ combined with the CH (LOJ/CH).

3.3.1 Learning Procedure of the LOJ/CH

The network of the LOJ/CH consists of nodes i, $i \in \{1, \ldots, N\}$, each of which has a synaptic weight vector $\boldsymbol{w}_i \in \Re^D$ to indicate a point of a D-dimensional input space. Any two of the nodes can form a lateral connection which determines the neighborhood relationship between the two nodes. These nodes with connections form a graph G to represent a topological structure. All the connections have ages to be used to delete invalid connections whose ages exceed some certain value. The network has the winner history table which can store up to M indices of the winner nodes.

Table 3.1 shows the learning procedure of the LOJ/CH.

In Step 1, a random weight is given to each node. In Step 2, a new input $\boldsymbol{x} \in \Re^D$ generated according to some unknown PDF is presented to the network. In Step 3, the winner node j that is the nearest node from the input measured by Euclidean distance and the second-winner node k that is the second-nearest node from the input are selected.

In Step 4, the age of each connection emanating from the winner node j is incremented. In Step 5, if nodes j and k have no connection between them, a new connection is created. The age of the connection between j and k is initially set to zero. In Step 6, connections whose ages are larger than a_{max} are removed.

Steps 7 through 9 are completely the same as the original LO discussed in Section 2.

Step 10 is performed only when creation and deletion are not performed. In this step, the weight vectors of the winner node and its neighbors are adjusted using Eq. (3.1).

$$\begin{aligned} \boldsymbol{w}_j^\star &= \boldsymbol{w}_j + \alpha_w(\boldsymbol{x} - \boldsymbol{w}_j), \\ \boldsymbol{w}_{n_i}^\star &= \boldsymbol{w}_{n_i} + \alpha_n(\boldsymbol{x} - \boldsymbol{w}_{n_i}), \\ & \quad i \in \{(\text{neighbors of } j)\}, \end{aligned} \tag{3.1}$$

Table 3.1: Learning procedure of the LOJ/CH.

Step 1.	Initialize weight of each node.
Step 2.	Present a new input.
Step 3.	Select the winner node j and the second-winner node k.
Step 4.	Increment ages of connections.
Step 5.	Adjust the connection between nodes j and k.
Step 6.	Delete old connections.
Step 7.	Adjust the winner history table.
Step 8.	Deduce win probability.
Step 9.	Decide whether creation and deletion is performed. If creation and deletion is needed, go to Step 11.
Step 10.	Adapt weights of the winner node and its neighbors, and then go to Step 2.
Step 11.	Create a new node for the winner node, and delete the weakest node. Adjust winner history table, and then go to Step 2.

where $\boldsymbol{w}_j^\star$ and $\boldsymbol{w}_{n_i}^\star$ are the weight vectors of the winner node and its neighbors, respectively. The learning rates for the weight adjustment of the winner node and its neighbors are α_w and α_n, respectively. This is a general process in sequential weight adaptation[46][47]. Then the learning goes back to Step 2.

In Step 11, creation and deletion of nodes are performed. Deletion of a node is performed by deleting the weakest node l and its connections. In addition, the entries of node l are removed from the winner history table. Creation of a node is described as follows. To clarify the explanation, let node j be the winner node before the creation, and node j' be the winner node after the creation. Node creation begins with creating a new node j_{new} whose weight vector points the position calculated by Eq. (3.2).

$$\boldsymbol{w}_{j_{new}} = \boldsymbol{w}_j + \alpha_w(\boldsymbol{x} - \boldsymbol{w}_j). \tag{3.2}$$

This equation means that a new node j_{new} is created in the same position of the winner node j' and then the new node moves toward the input instead of the winner node j'. The key point of the equation is that the jump and the adaptation of the node are simultaneously performed. To avoid pointing an identical position from two nodes, the winner node j' does not move in this process. The next process is to make proper connections between:

- node j_{new} and its neighbors,
- node j' and its neighbors, and
- nodes j_{new} and j'.

The process of the CH is to connect nodes whose weight vectors have neighboring *masked Voronoi polygons*[32]. In other words, two adjacent nodes on the connected input manifold M are connected. Thus, obeying this policy allows the network to connect these two adjacent nodes on M. We assume that two nodes j_{new} and j' are on the connected manifold M. First, a new connection between nodes j_{new} and j' with age "zero" is made. Next, if the following condition is satisfied, each connection emanating from node j' is changed to the connection emanating from node j_{new} with keeping the same age.

$$||\boldsymbol{w}_{j_{new}} - \boldsymbol{w}_{n_i}|| < ||\boldsymbol{w}_{j'} - \boldsymbol{w}_{n_i}||, \tag{3.3}$$

where the norm is Euclidean distance and nodes n_i are neighbors of j'. The last process of the creation is to replace the entries of node j in the winner history table with those of nodes j_{new} and j' alternately. This is because nodes j_{new} and j' share the space that was previously occupied by node j.

The following learning is accomplished by iterating Steps 2 through 11 while inputs are presented.

3.3.2 Experimental Results and Discussions

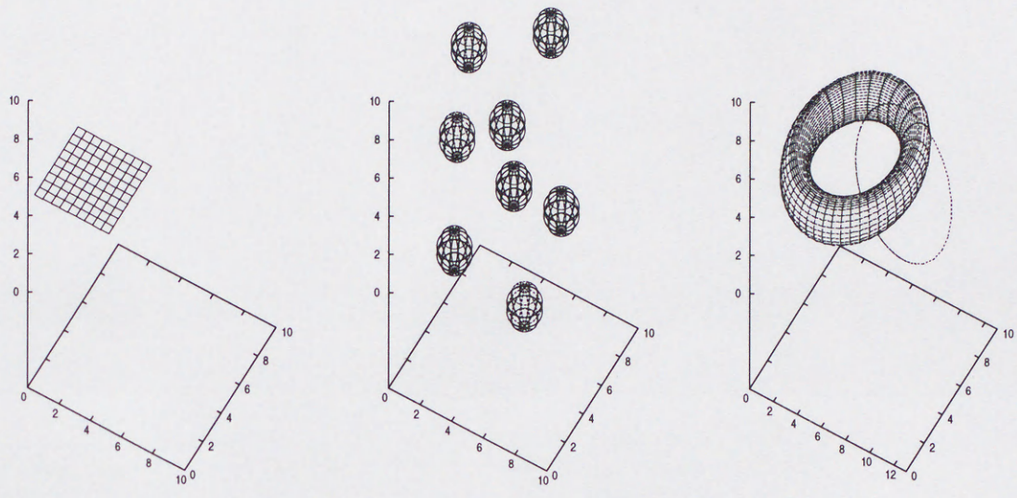
In this section, some experiments to examine the performance of the LOJ/CH are carried out. The first experiment uses artificial data to show an ability of the LOJ/CH. The second experiment is an image coding application, which uses real data.

Applying the LOJ/CH to Artificial Data

Figure 3.1 shows the input manifold $M \subseteq \Re^3$ used in the experiments.

The input sequence consists of three phases, i.e., the manifold changes twice at $T = 10001$ and $T = 20001$. Here, T is the number of input vectors presented(learning steps). In the first phase(from $T = 1$ to $T = 10000$) named "simple", the manifold is a simple 2-dimensional square, which is used to confirm the ability of ANNs to distribute weights and form a map. In the second phase(from $T = 10001$ to $T = 20000$) named "clusters", the manifold consists of several 3-dimensional balls for examination of the clustering ability of ANNs. In the last phase(from $T = 20001$ to $T = 30000$) named "complex", the manifold consists of a 2-dimensional torus and a 1-dimensional ring for examination of the ability of ANNs to represent a "complex" topology of the manifold M. For comparison, a conventional SOM is examined by using the same input manifold. Since the SOM is not designed to be used in nonstationary environments, a SOM network cannot be applied directly to these experiments. For the SOM network, therefore the input sequence is divided into three stationary phases, and the performance during each stationary phase is individually measured as three sub-experiments(each learning length = 10000 steps).

The SOM network has a 2-dimensional (fixed) topology, and the number of nodes N is 100(10×10). The initial learning rate for the winner node at $T = 1$ is 0.1 and linearly decays to 0 at $T = 10000$. The neighboring function is the "bubble" function[13] and the radius to determine the neighbor is 10 at $T = 1$ and linearly decays to 1 at $T = 10000$. In the network using LOJ/CH, following parameters are used: $N = 100$, $\alpha_w = 0.1$, $\alpha_n = 0.003$, $a_{max} = 50$, M = 1800, Cr = 3.5, De = 0.05. Within the learning process of the LOJ/CH, creation and deletion of the nodes are inhibited in the initial learning stage of $T = 1$ through 1000, because



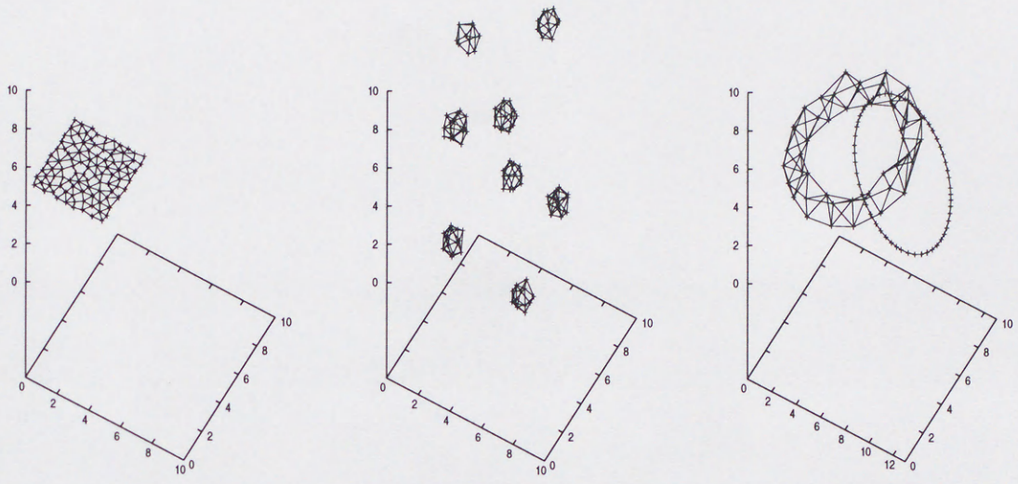
(a) Simple (b) Clusters (c) Complex

Figure 3.1: Input manifold.

the insufficient number of entries in the winner history table in the initial learning stage make the network behavior unstable.

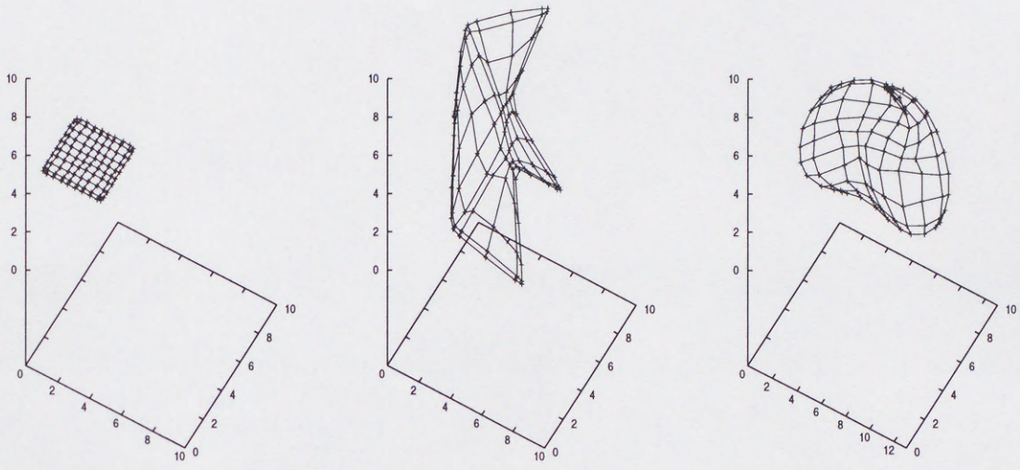
The LOJ/CH and the SOM are compared in terms of approximation of PDFs and quality of the topology preserving map. The mean square error(MSE) is used for a metric of approximating PDFs. In general, an appropriate metric of topology preservation consists of two parts. One is the degree of topology preservation of a map from a network G to an manifold M, and the other is the degree of preservation from M to G. Although a metric dealing with a network G having a variable topology is needed, there only exists a metric to calculate the degree of topology preservation from G to M[48]. Therefore, this thesis uses an imperfect metric TPG=$\Phi_G^M(-1)$ that is a part of a topological function proposed in [48]. The TPG calculates the number of connections which violate topology preservation in the map from G to M.

Tables 3.2 and 3.3 show the MSE and the TPG at the end of each phase, respectively. The results are mean values during ten trials. Figures 3.2 and 3.3 show a typical nodes distribution with connections of LOJ/CH and SOM at the end of each phase. In all phases, the MSEs of the LOJ/CH are superior to those of the SOM including the simple phase, even though the simple phase has the same topology of the SOM network. A remarkable difference between them in the clusters phase shows that the LOJ/CH would be favorable in clustering. Figure 3.2(b) also supports this observation: all the clusters are properly separated. On the other hand, as shown in Figure 3.3(b), it is difficult to identify several clusters from the map of the SOM network. Since the topologies of the input manifold in the simple phase and the SOM network are the same, the TPG of the SOM in the first phase is perfect. However, the TPGs of the SOM in the other phases are remarkably poor. This is especially shown in Figure 3.3(c): the representation of the SOM network does not form the torus and the ring. On the other hand, even though movement of weight vectors prevents the LOJ/CH from forming perfectly topology preserving maps, the TPGs of the LOJ/CH are quite smaller and more stable than those of the SOM in all phases. Especially, the TPG in the complex phase is almost the same as that in the simple phase. Figure 3.2(c) also shows that the LOJ/CH network can represent the complex topology in the complex phase. As a result, the ability of the LOJ/CH to represent topology



(a) Simple (b) Clusters (c) Complex

Figure 3.2: Node distributions(LOJ/CH).



(a) Simple (b) Clusters (c) Complex

Figure 3.3: Node distributions(SOM).

of input manifold would be useful for approximation problems with complex manifolds even if the topology changes.

Table 3.2: MSE of LOJ/CH and SOM in each phase.

ANN	Simple	Clusters	Complex
LOJ/CH	0.20	0.36	0.51
SOM	0.27	0.57	0.83

Table 3.3: TPG of LOJ/CH and SOM in each phase.

ANN	Simple	Clusters	Complex
LOJ/CH	2.6	5.3	3.0
SOM	0.0	87	87

Applying the LOJ/CH to a Binary Image Compression Problem

This thesis assumes a binary motion image captured from a white sheet where,

- drawings are written by a black pen on the white sheet.
- drawings consist of lines.
- drawings can be added or be erased.

Under these assumptions, this thesis applies the LOJ/CH to a simple binary motion image compression problem by using an ability of the LOJ/CH to represent the topology of the drawings.

Figure 3.3.2 shows the procedure to form a PDF from information on a white sheet. First, black drawings on the white sheet is captured by a CCD camera. Next, the captured image is thresholded to have two colors, such as black and white. Then, a stationary PDF is created from the image by giving uniform probability to the black pixels and zero probability to the white pixels. Inputs generated by the PDF is sent to the LOJ/CH network to form a topology preserving map.

The motion image sequence used in this experiment consists of 20 frames. There is letter "E" in the first four frame, and then "A" is gradually added from frames five through eight. The LOJ/CH network receives 2000 inputs per frame. In the network, following parameters are used: $N = 100$, $\alpha_w = 0.05$, $\alpha_n = 0.0006$, $a_{max} = 120$, M = 1250, Cr = 3.0, De = 0.05. Within the learning process, creation and deletion of the nodes are inhibited in the initial learning stage of $T = 1$ through 1250, because the insufficient number of entries in the winner history table in the initial learning stage make the network behavior unstable. In the experiments, a TPN using the simple Kohonen learning with the Competitive learning(K/CH) is also used for reference. Parameters of the K/CH network obey that of the LOJ/CH.

Figures 3.5, 3.6, 3.7 and 3.8 show original images and the corresponding maps obtained by the LOJ/CH and the K/CH on frame #1, #2, #8, and #9, respectively. In the results on frame #1, both TPNs remain nodes on the "white" regions, but the number of nodes in the LOJ/CH is smaller than that of the K/CH. Both also roughly represent shape "E." In frame #2, LOJ/CH already correctly represents the topology of the original image with having no

remaining nodes. On the other hand, the representation of K/CH does not change compared with frame #1. There is a pen writing "A" in frame #8, and gives a shade in the bottom-right hand part of the sheet. LOJ/CH represents the shade, the pen, and new letter "A" with old letter "E." On the contrast, K/CH correctly represents "E" only , and the other part does not represent the original image. In frame #9, letter "A" has been already written, and there are only letters on the sheet again. LOJ/CH rapidly follows the change in the original image, showing "E" and "A" correctly. Hence, the K/CH does not change its representation from the previous frame. Moreover, K/CH still remains nodes that cannot win in the competition. From the above result, it is shown that the LOJ/CH could be used for forming a map to represent simple image, which may change.

Next, the compression ratio of the LOJ/CH in this experiment is discussed. The size of the thresholded image is $311 \times 236 = 73396$(pixels), and it has 73396 bits of information. On the other hand, the LOJ/CH has N=100 of nodes, and $2 \times N - 3 = 197$ of edge at the maximum for 2-dimensional input space[45]. A node requires 17 bits of information to indicates coordinates in the input space, which have 73396 points. An edge requires $7 \times 2 = 14$ bits of information to store two indices of nodes. As a result, a map created by the LOJ/CH requires $100 \times 17 + 197 \times 14 = 4458$ at the maximum. Consequently, the compression ration of the LOJ/CH is larger than $73396/4458 = 16.5$.

In this experiment, the compression speed of the LOJ/CH is about 2 frames/sec on a PC with Celeron 415MHz. This would be a satisfactory speed for practical usage.

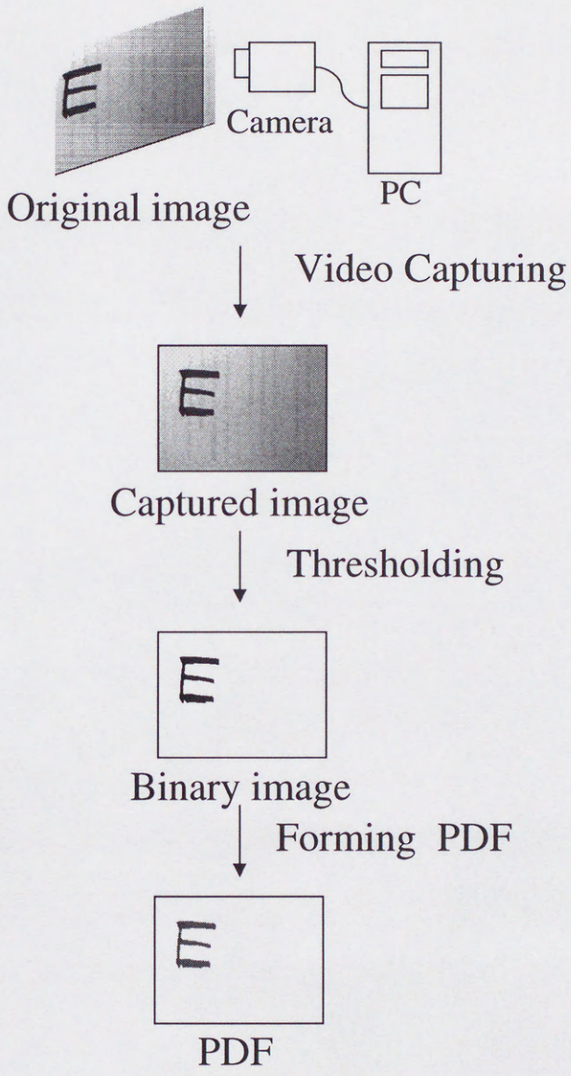
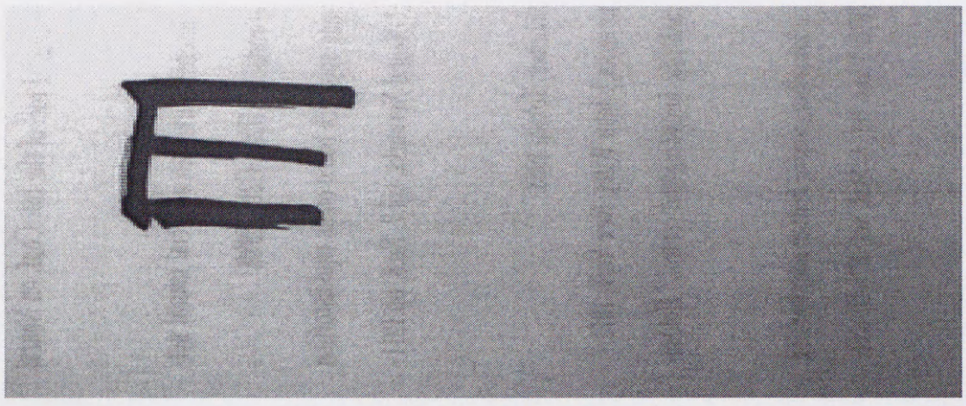
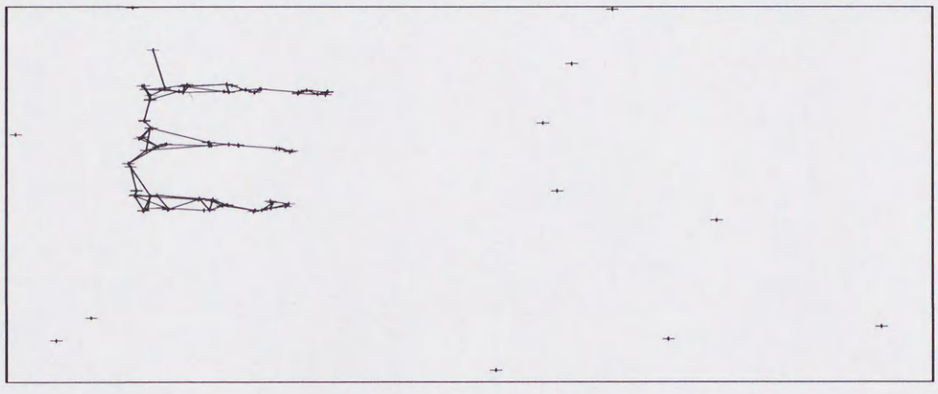


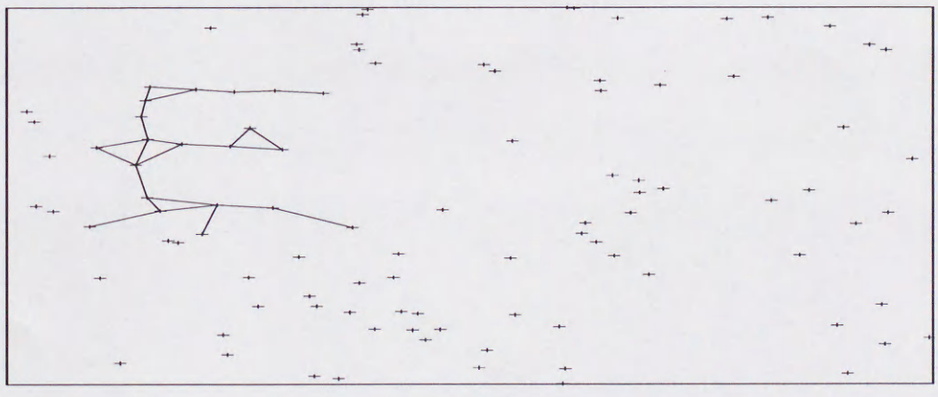
Figure 3.4: The process to form a PDF.



An original image.

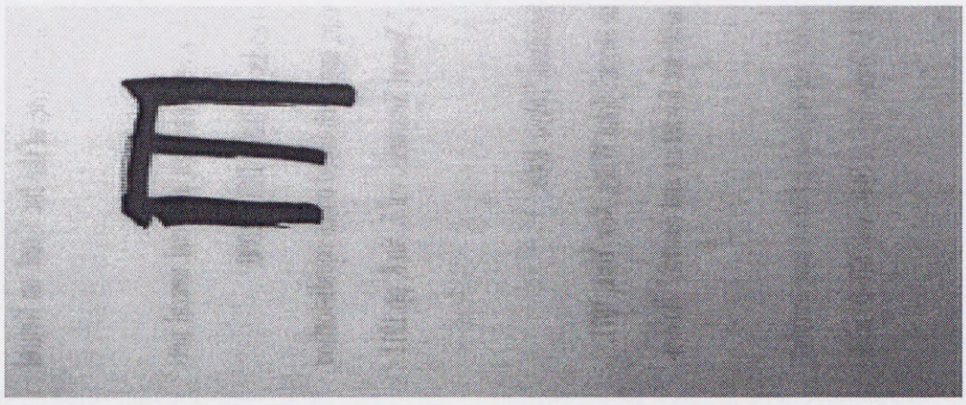


A map by LOJ/CH.

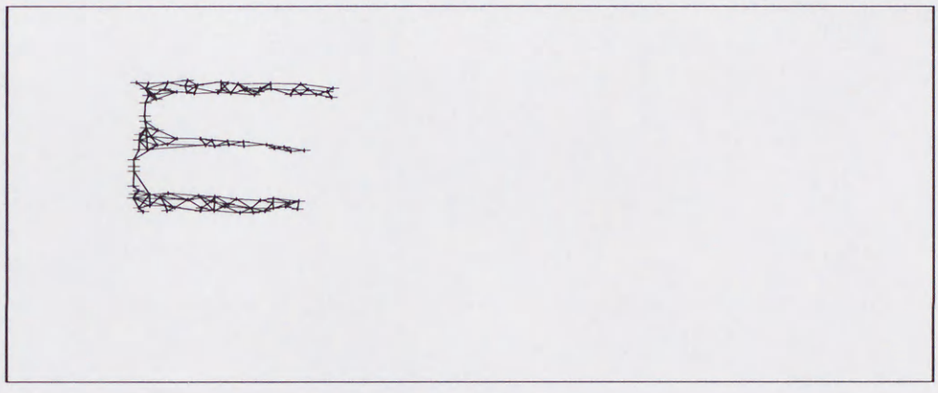


A map by K/CH

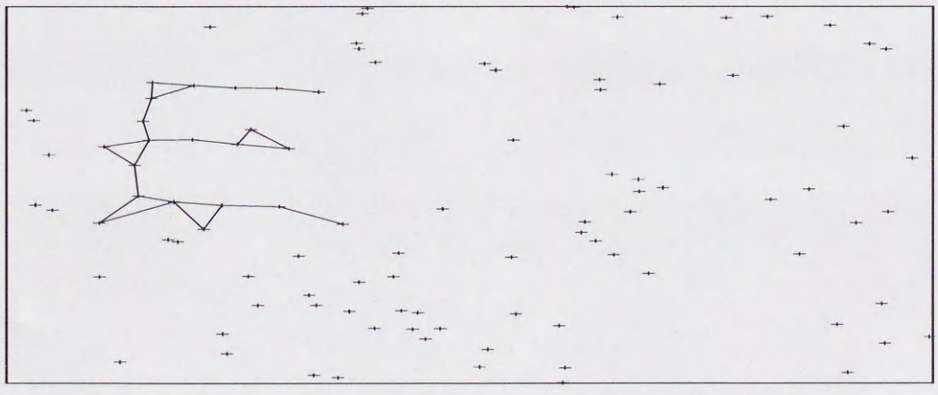
Figure 3.5: Results of frame #1.



An original image.

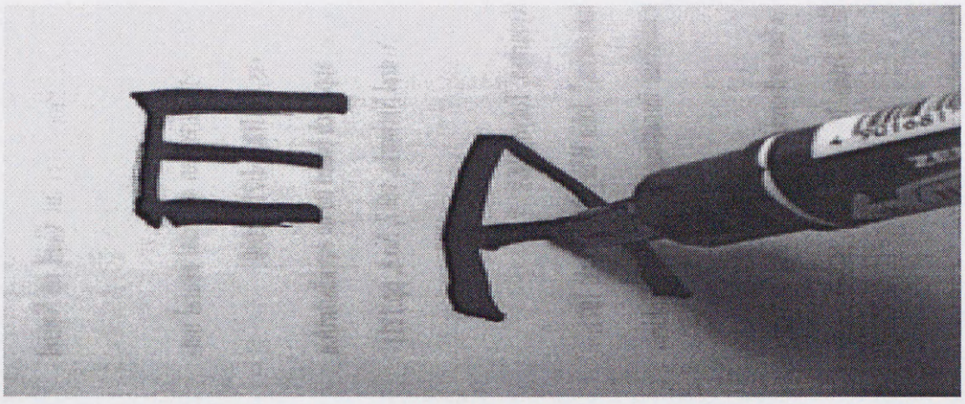


A map by LOJ/CH.

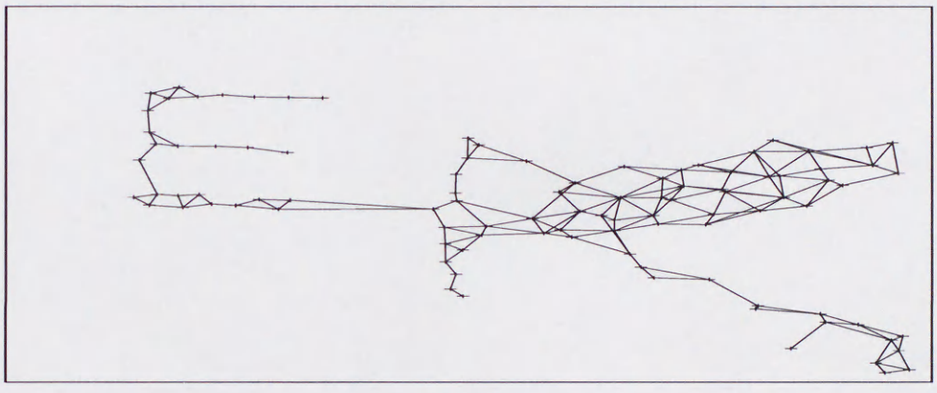


A map by K/CH.

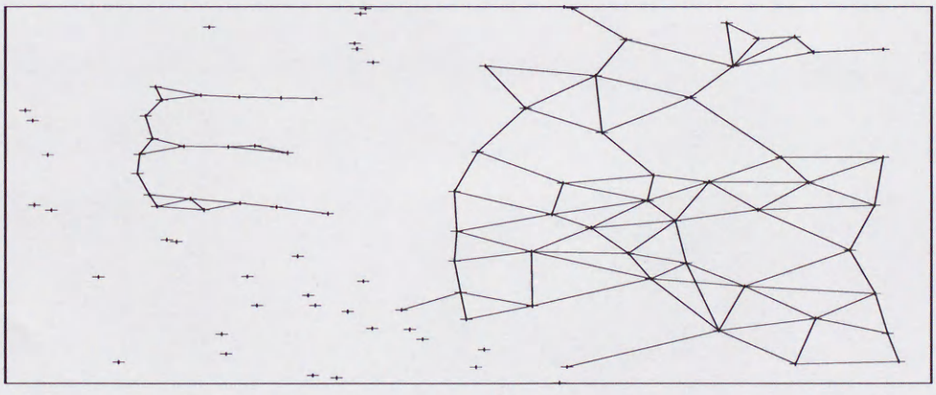
Figure 3.6: Results of frame #2.



An original image.

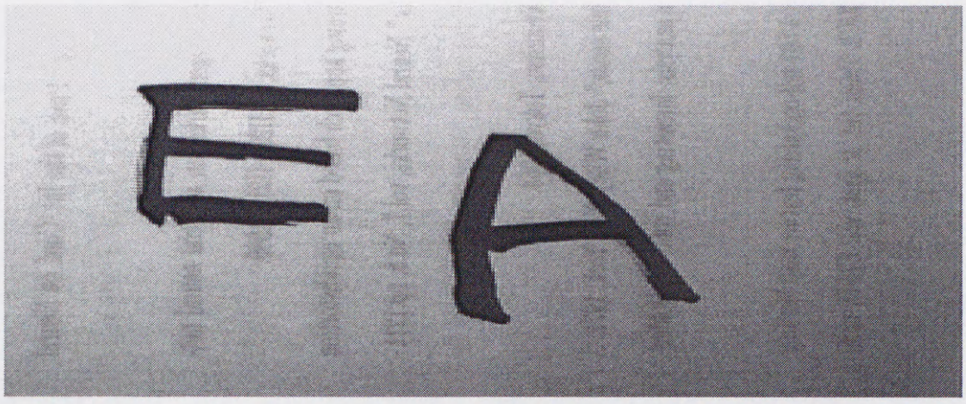


A map by LOJ/CH.

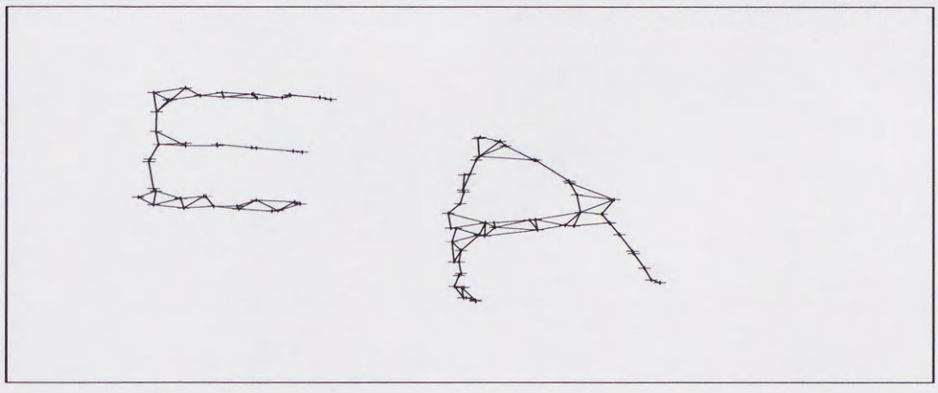


A map by K/CH.

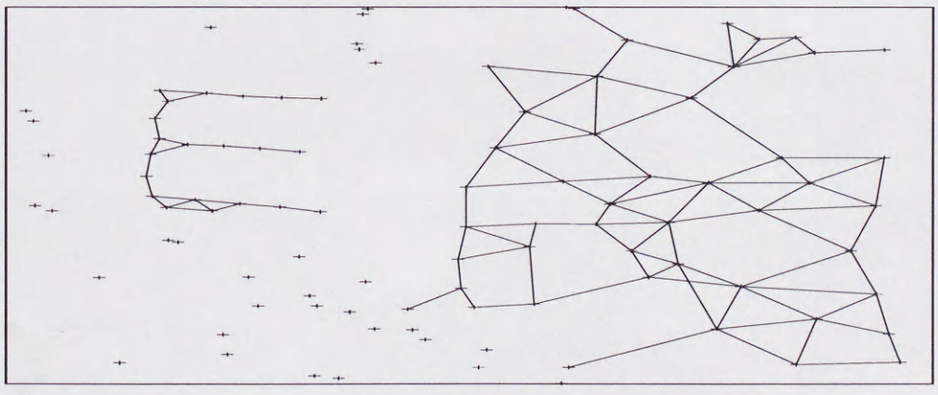
Figure 3.7: Results of frame #8.



An original image.



A map by LOJ/CH.



A map by K/CH.

Figure 3.8: Results of frame #9.

3.4 Conclusions

A learning algorithm named LOJ/CH have been proposed. The algorithm is suitable for an ANN to form a topology preserving map from an input manifold whose topology may change. The LOJ/CH is a integration of the "LOJ" mechanism as a NVQ algorithms, which can approximate nonstationary PDFs, and the topology generating algorithm "CH", which can form a perfectly topology preserving map from arbitrary input manifold. The results of the first experiment show that the LOJ/CH can rapidly form a topology preserving map even if the topology of input manifold changes. The LOJ/CH also has the ability to clusterize the input space in nonstationary environments. The results of the second experiment show that the LOJ/CH can be used to a simple motion image compression problems with a high compression ratio(over 15x).

4 A Neural Memory System

4.1 Introduction

As illustrated in the beginning of the introductory section, study on memory has been an attracting topic for many researchers. In the field of ANNs, several memory models have been proposed[5][6][50][51][52]. Especially, associative memory[18] is the most popular artificial neural memory system. The system memorize a map from a set of pattern vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ to the set of output vectors $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$. When associative memory receives a pattern vector $\boldsymbol{x}_k$, it produces the corresponding output $\boldsymbol{y}_k$.

Associative memory can be classified from the following three points of view[10]:

- feedforward or recurrent.
- autoassociative or heteroassociative.
- accretive or interpolative.

The first viewpoint is concerned with the system architecture. Feedforward associative memory systems[5] have no feedback paths and produce outputs as soon as the inputs received. Recurrent systems[51] have feedback paths. When an input received, a recurrent system activates the feedback paths to find a stable state, and the system produces the state as the output.

The second viewpoint is concerned with the relationship between inputs and outputs. In autoassociative memory systems, a set of inputs $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L\}$ is exactly the same as a set of outputs $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$, namely, $\boldsymbol{x}_1 = \boldsymbol{y}_1, \ldots, \boldsymbol{x}_L = \boldsymbol{y}_L$[5]. Though this seems to have no meaning, autoassociative memory is usually used to form a memory system that produce the correct output from an imperfect or a noisy input. On the other hand, heteroassociative system assumes that inputs are different from outputs[53].

The last viewpoint is concerned with the treatment of error in inputs. Accretive systems try to produce the same output $\boldsymbol{y}$ for noiseless input $\boldsymbol{x}$ and noisy input $\boldsymbol{x} + \boldsymbol{\epsilon}$[5]. Here, $\boldsymbol{\epsilon}$ is a sufficiently a small error vector. Interpolate systems produces $\boldsymbol{y} + \boldsymbol{\delta}$ for a noisy input. Here, $\boldsymbol{\epsilon}$ is also a sufficiently small error assuming that if $\boldsymbol{\epsilon} \to \mathbf{0}$, then $\boldsymbol{\delta} \to \mathbf{0}$[51].

As mentioned above, associative memory systems have been widely studied, and they can

further be classified from several viewpoints. However, associative memory systems commonly have several problems:

1. the system cannot add new items after the first memorization.
2. memory representation is a map from input to output.
3. whole information is given to the system at the same time.

Owing to the first problem, the system must memorize all the information at once. This is biologically unacceptable, and also inconvenient in engineering use. The second problem shows that the associative memory system does **not** keep information of the pattern, but keep information of the relationship between an input and its correct output. This means that without having the correct output, the system cannot memorize anything. The third problem makes the system inflexible because there are many cases in practical problems that produce insufficient information at a time.

This thesis proposes an approach to a neural memory system from quite a different way to solve these problems, that is, using LOJ networks which form approximated representation of an input PDF. This would be the first attempt to deal with nonstationary PDFs in an artificial neural memory system. In section 4.2, the possibility of using LOJ networks for a neural memory system is discussed. Section 4.3 proposes a memory system that forms memory from nonstationary PDFs. Section 4.4 gives experiments that evaluate memorization and recall ability of the proposed system. Section 4.5 concludes this section.

4.2 Application of the LOJ Network to a Neural Memory System

Memory systems are required to have three important functions that are, "memorization," "retention," and "recall[49]." Memorization is a function to store information from external inputs. Retention is a function to keep the stored information, and recall is to find information kept in memory.

An LOJ network has a group of nodes that approximates distribution formed by a group of inputs. From the viewpoint that the nodes memorize information of the inputs, the LOJ network regarded as a memory system, which memorizes input distribution. The LOJ network allows the system to deal with data, each of which has little information. In addition, the LOJ network memorizes input distribution itself, not a relationship between an input and an output. Besides, the LOJ network changes its node distribution by following changes in the nonstationary PDF. However, changing node distribution causes loss of memory, and this drawback prevents the LOJ network from having important functions as a memory system.

The LOJ network with online decision of convergence (called an LOJ/OD network) can form and retain memory from a nonstationary PDF without terminating its learning. However, in this case, the network can memorize only one stationary piece of the nonstationary PDF.

If an input sequence is generated from a nonstationary PDF that consists of several stationary PDFs, using several LOJ/OD networks can memorize the nonstationary PDF. This study proposes a neural system that consists of several LOJ/OD networks where each of the LOJ/OD networks keeps memory of a stationary piece of the nonstationary PDF. The problems of the single LOJ/OD network in memorization and retention, which are requirements of memory systems, are cleared by using multiple LOJ/OD networks. As the LOJ/OD network can evaluate how correctly the network adapt to the input sequence, recall of memory could be realized by the competition among LOJ/OD networks using their evaluation values.

4.3 A Neural System Forming Memory from Inputs Obeying a Non-stationary Probability Density Function

4.3.1 The System Architecture

The neural memory system discussed in this thesis is shown in Figure 4.1. Solid lines indicate current data flows and dotted lines possible data flows. The system includes several LOJ/OD networks having the following three exclusive states:

1. Reserved. (dot-lined boxes) Some networks are reserved for the future use. Reserved networks do not take part in input receiving and output producing.
2. Learning. (shaded box) Only one network has this state at a time. The network receives the input sequence and approximates the PDF by moving its nodes.
3. Recalling. (solid-lined box) Networks that have already completed their learning have this state. Each networks receives the input sequence and produces an estimated value of the standard deviation of the win probability. The standard deviation shows similarity between the input sequence and memorized time-invariant peace of the PDF.

The state transits in the $1 \rightarrow 2 \rightarrow 3$ order. Transition conditions are explained as follows:

$1 \rightarrow 2$ None of the "Learning" networks exist.

$2 \rightarrow 3$ The current "Learning" networks converges and no other "Recalling" networks match the current potion of the PDF.

Each LOJ/OD network has the winner history table to estimate the standard deviation of win probability. Convergence of the "Learning" LOJ/OD network is concretely given by Equation 4.1.

$$(\sigma < \theta_c) \quad \& \quad \text{(no creations occurs in past M times)} \tag{4.1}$$

Here, σ is the estimated win probability and θ_c is a threshold. M is the size of the winner history table. Matching of the "Recalling" network is given by Equation 4.2.

$$\sigma < \theta_m \tag{4.2}$$

Here, θ_m is a threshold that satisfies $\theta_m > \theta_c$.

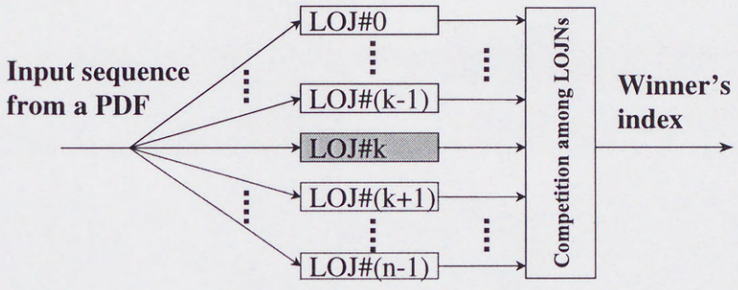


Figure 4.1: The proposed neural memory system.

4.3.2 The Procedure for Forming Memory

This section explains how the system forms and recalls memory. The explanation consists of four steps. In the explanation, a nonstationary PDF that consists of two stationary PDF is used as an example.

Figure 4.2 shows the initial stage of the memory system. The system has n LOJ/OD networks. Only one network(# 0) is in the "Learning" state and remaining are in the "Reserved" state. Inputs from a PDF(shown in left-hand side of the figure) are presented to the system. Network # 0 receives the inputs and starts adapting to the inputs through competitions among nodes.

Figure 4.3 shows the second stage. Network # 0 produces standard deviation of internal win probability σ_0, which estimates the accuracy of approximation. After about a few thousands of input presentation, Network # 0 converges with satisfying $\sigma_0 < \theta_c$. Then Network # 0 changes its state to "Recalling," which means the network correctly memorizes the current stationary piece of the nonstationary PDF. At the same time, Network # 1 changes its state from "Reserved" to "Learning" because no "Learning" networks exist in the system. Though "Recalling" Network # 0 terminates its learning, it keeps on receiving inputs, performing competitions among nodes, updating the winner history table, and producing σ_1. Produced σ_1 shows the degree of matching of the network to the current PDF. As only Network # 0 has the state "Recalling", the system produces the index # 0 as the output. In this stage, Network # 1 never changes its state because Network # 0 has already sufficiently approximated the PDF satisfying $\sigma_0 < \theta_m$.

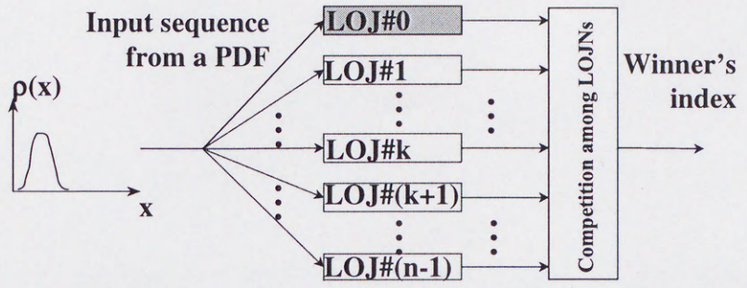


Figure 4.2: The initial stage.

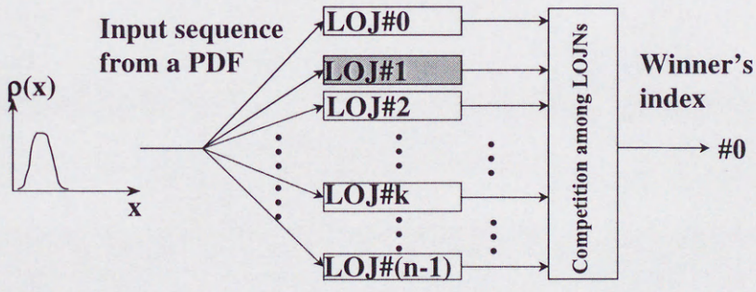


Figure 4.3: The second stage.

Figure 4.4 shows the third stage. In this stage, the PDF changes. As "Recalling" Network # 0 does not adapt to the changed PDF, σ_0 increases. In contrast, "Learning" Network # 1 decreases its σ_1 with receiving the input sequence produced by the changed PDF. However, the system keeps producing index #0 till Network # 1 converges because Network #0 is the only network having a state "Recalling."

Figure 4.5 shows the behavior in last stage. Network # 0 keeps on increasing its σ_0 by receiving inputs produced by a different stationary piece of the nonstationary PDF. On the other hand, Network #1 decreases its σ_1. After a while, Network # 0 does not match the current PDF, and Network # 1 converges by satisfying:

$$\sigma_0 < \theta_c \ \ \& \ \ \text{(no creations occurs in past M times for \# 0)} \ \ \& \ \ \sigma_1 \geq \theta_m$$

"Learning" Network # 1 changes its state to "Recalling" and starts storing σ_1 for competition among networks. New Network # 2 begins to learn. The system compares values $\sigma_i, i = 0, 1$ and produces the index # 1, which has the least value among σ_i, as the output.

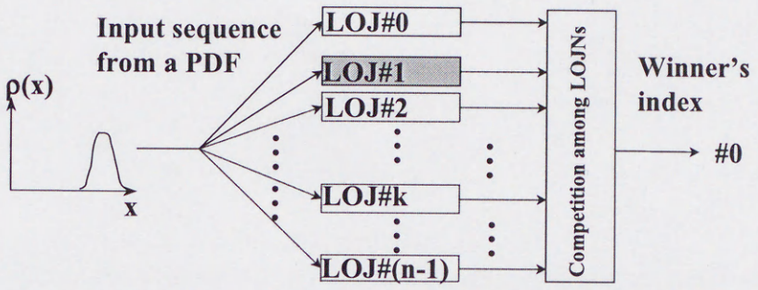


Figure 4.4: Behavior in the third stage.

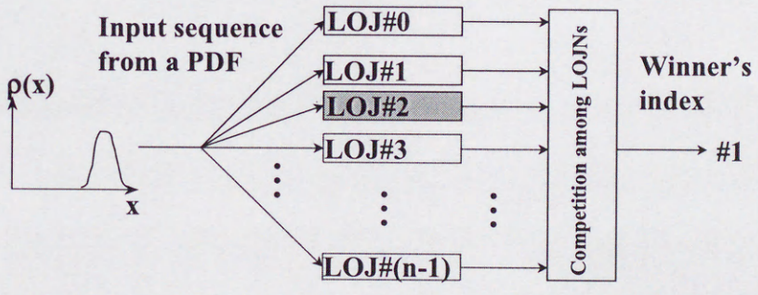


Figure 4.5: Behavior in the final stage.

4.4 Experimental Results and Discussion

For a demonstration of the system, some experiments are performed. As shown in Figure 4.6, a nonstationary PDF consisting of two kinds of stationary PDFs is used. The stationary PDFs have uniform distribution inside their squares. The probability density outside the squares is zero. The first piece of the nonstationary PDF is called (a) and the second one (b). In the experiments, the stationary pieces appear alternatively in order (a)→(b)→(a)→(b). Each of the pieces is presented 10000 times. All LOJ/OD networks have the same network parameters: the number of nodes N= 50, learning ratio $\alpha = 0.1$, node creation parameter Cr= 2.0, node deletion parameter De= 0.05, the size of the winner history table M= 1250, threshold for convergence $\theta_c = 6 \times 10^{-3}$, and threshold for matching $\theta_m = 10 \times \theta_c$. Creations and deletions of the nodes in the LOJ/OD networks are inhibited until the history table is filled. Ten trials are performed in each experiment, and the mean values of Ten trials are used for performance measurement.

Figure 4.7 shows typical node distribution of the networks having indices # 0 and # 1. The system correctly outputs the indices # 0 and # 1 as # 0→# 1→# 0→# 1. Index # 0 corresponds to (a), and # 1 to (b), respectively. Table 4.1 shows the input–output relations of the system.

Stationary PDF (a) starts generating inputs at $T = 1$, where T is the number of input presentations. After about 4000 presentations, Network # 0 in the memory system converges, and the system outputs index # 0 that corresponds to PDF (a). At $T = 10000$, the PDF changes its behavior to the second piece (b). After about 3500 presentations from the change, Network # 1 memorizes nonstationary PDF (b), and the system outputs its index. Network # 0 needs about 500 more presentations than Network # 1 because the inhabitation of creations and deletions of nodes in the initial stage makes the convergence slowdown. At $T = 20000$, the PDF again changes to the first piece (a). In this situation, the system already has memory of (a), therefore it takes only about 625 input presentations to output corresponding index # 0. At $T = 30000$, the PDF changes to the second piece (b), and after about 625 presentations, which is almost the same as the situation of piece (a), the system recalls corresponding index # 1.

The experiments shows that the system correctly memorizes and recalls stationary PDFs, which are pieces of a nonstationary PDF. This points out the possibility of using the system to memorize and recall nonstationary PDFs.

625 presentations, which are the required input presentations for recalling, is correspond to the half size the winner history table. In addition, no overlap region where input occurs exists in the two stationary PDFs. These suggest the size of history table strongly affects the required number of presentations in recalling. Therefore, further investigations would be needed to precisely evaluate the performance of the system.

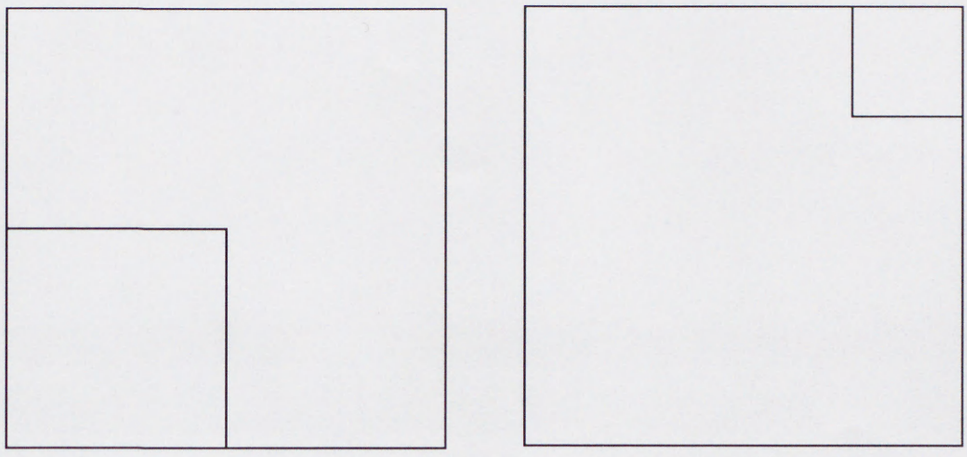


Figure 4.6: The pieces of nonstationary PDF used in the experiments.

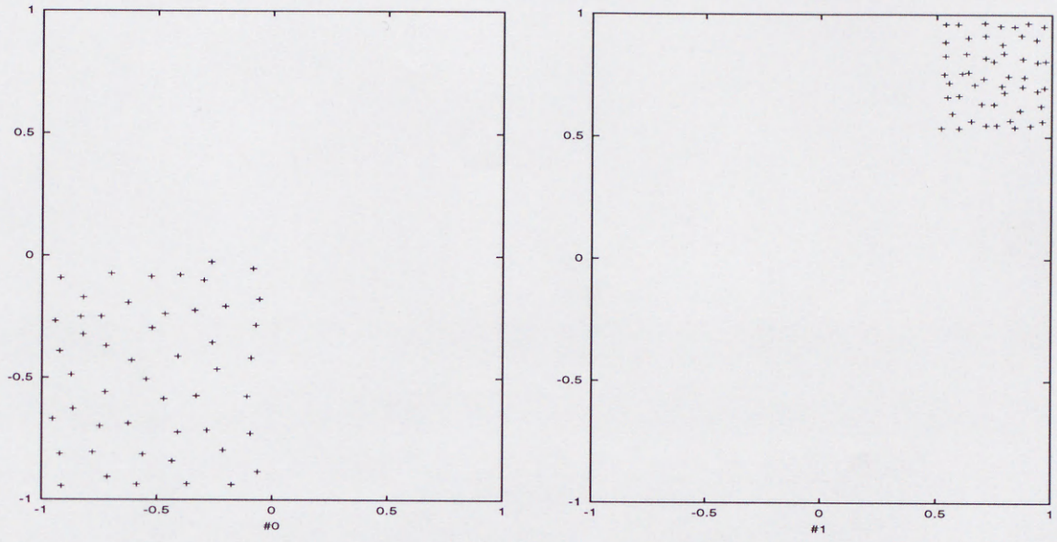


Figure 4.7: Node distribution.

Table 4.1: Input–output relationships of the system.

Distribution	a	b	a	b
PDF	1	10001	20001	30001
System outputs	4044	13501	20624	30626

4.5 Conclusions

We have proposed and demonstrated a neural system that forms a memory by receiving inputs obeying an unknown nonstationary PDF. This would be the first attempt to deal with nonstationary PDFs on an artificial neural memory system. The system keeps statistical information by distributing it equally over the system. Experimental results have shown the potential of the system to memorize and recall stationary pieces of nonstationary PDFs. The system can be used to model nonstationary phenomena. This ability is desirable for various applications, for example, process control, economical modeling, and so on.

5 Conclusions

So far only a few attempts have been made to solve the problem of approximating nonstationary PDFs in spite of its importance. This thesis discusses methods to approximate nonstationary PDFs by using self-organizing neural networks from the following three viewpoints.

- Quantizing of nonstationary PDFs
- Forming topology preserving maps from nonstationary PDFs
- Forming memory from nonstationary PDFs

New artificial neural networks for each problem have proposed and evaluated through experiments. Here, the summary of this thesis is presented as follows:

In Section 2, an NVQ quantizing nonstationary PDF has been discussed. A Concept and a definition of nonstationary PDFs discussed in this thesis have been given first. To approximate nonstationary PDF, a learning algorithm named a Law-of-the-jungle(LOJ) mechanism that

1. requires small number of inputs to converge,
2. has no time-decaying adaptation parameters, and
3. makes nonstop(online) and no periodic adaptation

has been proposed. Experimental results have shown that an ANN using the LOJ mechanism(LOJ network) required 1/10 of inputs needed by conventional ANNs to achieve a certain accuracy, and the LOJ network is able to approximate nonstationary PDFs.

In the case of approximating PDFs under the nonstationary environment, the network cannot stop learning to measure the approximation accuracy. To overcome the problem, a method to dynamically estimate accuracy of an LOJ network during learning has been proposed. The method uses the estimated win probability stored in the LOJ network to calculate the accuracy. Comparison of the accuracy obtained by a general method with the estimated accuracy has been examined. The experiments have illustrated the estimated win probability can be used to calculate the approximation accuracy.

Section 3 has discussed an ANN forming topology preserving maps from nonstationary PDFs. A learning algorithm named LOJ/CH for an ANN has been proposed to form a topology preserv-

ing map from an input manifold whose topology may change. The LOJ/CH is a combination of the "LOJ" as a NVQ algorithm, which can approximate nonstationary PDFs, and the topology generating algorithm "CH," which can form a perfectly topology preserving map from arbitrary input manifold. Experimental results have shown that the LOJ/CH can rapidly form a topology preserving map even if the topology of input manifold changes. It has also been shown that the LOJ/CH has the ability to clusterize the input space in nonstationary environments. Experimental results have also shown that the LOJ/CH can be applied to a simple motion image compression problems with a high compression ratio(over 15x).

Section 4 have discussed an artificial neural system that forms memory from nonstationary PDFs. Though many researches to form memory from patterns have been performed, no study on dealing with nonstationary PDFs has been found. The thesis has proposed a neural system using several LOJ networks that receives input from a nonstationary PDF, and memorizes its stationary pieces. Through some experiments, the system ability of memorizing and recalling stationary PDFs has been illustrated.

This thesis has discussed self-organizing neural networks. An LOJ network forms an approximated representation of a nonstationary PDF. This ability can be applied to problems like motion image compression, which needs adaptive coding under the nonstationary environment. LOJ/CH networks form a topology preserving map from nonstationary PDF. LOJ/CH can be applied to problems like feature extraction from motion image, which requires an ability to extract a data structure from nonstationary data distribution. The memory system proposed in this thesis memorizes and recalls stationary pieces from a nonstationary PDF. The system receives inputs each of which has little information, and forms a distributed represents of information with a large number of nodes. In addition, the nodes almost equally shares information. In these points of view, the system can be regarded as a biological memory model. Moreover, it can be applied to a problem like process control by taking advantage of the ability to model nonstationary phenomena. The following gives future research works:

- applying LOJ networks to the motion image compression.
- applying LOJ/CH networks to the feature extraction from the motion image.

- detailed analysis of the proposed memory system.

References

[1] R. Sorabji(trans.), "De memoria et reminiscentia," *Aristotle on Memory*, Providence, RI: Brown University Press, 1972.

[2] H. Ebbinghaus, "Memory:a contribution to experimental psychology(English translation)," Dover, 1964.

[3] M. Proust, K. Inoue(trans.), "A la recherche du temps perdu," Chikuma, Japan, 1992.

[4] D. Marr, "A theory for cerebral neocortex," Proc. Roy. Soc London Ser. 176, pp. 161–234, 1970.

[5] T. Kohonen, "Correlation matrix memories," IEEE Transaction on Computers, C-21, pp. 353–359, 1972.

[6] J. A. Anderson, "A simple neural netowak generating an interactive memory," Mathematical Biosciences, 14, pp. 197–220, 1972.

[7] M. A. Arbib, "The metaphorical brain 2," John Wiley & sons, New York, 1989.

[8] F. Click, "The astonishing hypothesis," Charles Scribner's Sons, New York, 1994.

[9] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity,", Bulletin of Mathematical Biophysics 5, pp. 115–133, 1943.

[10] H. Nielsen, "Neurocomputing," Addison-Wesley, Redwood City, 1990.

[11] S. Amari, "New development in neural networks(in Japanese)," Science Inc, Tokyo, 1993.

[12] K. Yana, and Y. Suzuki, "neuro infomation processing technology(in Japanese)," Kaibundo, Japan, 1992.

[13] T. Kohonen, "Self-Organizing Maps," Springer-Verlag, Berlin Heidelberg, 1995.

[14] R. Linsker, "From basic network principles to neural architecture," Proc. Natl. Acad. Sci. USA, pp. 7508–7512, 1986.

[15] S. Tanaka, "Theory of self-organization of cortical maps: mathematical framework," Neural Networks, 3, pp.625–640, 1990.

[16] H. D. Block, "The perceptron: amodel for brain functioning. I", Reviews of Modern Physics 34, pp. 123–135, 1962.

[17] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," Proc. of the National Academy of Sci., 81, pp.3088–3092, 1984.

[18] T. Kohonen, "Self-Organization and associative memory," Springer-Verlag, Berlin Heidelberg, 1989.

[19] D. E. Rumelhart, G. E. Hinton, R. J. williams, "Learning representations by back-propagating errors," Nature 323:pp.533–536, 1986

[20] S. Nakagawa, K. Shikano, Y. Toukura, "Speech sound, auditory sense, and neural network models," Ohm, Tokyo, 1990.

[21] H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "A method for improving classification capability of multilayer perceptrons, " The trans of the institute of electronics, information and communication engineers, D-II, Vol. J80-D-II, No.1, pp. 390–393, 1997.

[22] H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "A Systematic Training Data Generation Algorithm for Improving the Classification Accuracy of Multilayer Perceptrons," submitted to *Neural Computation.*

[23] H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "Acceleration Techniques for the Network Inversion Algorithm," to be published in *IEICE trans. on inf. & sys.*

[24] T. J. Sejnowski, and C. R. Rosenberg, "NETtalk: a parallel network that learns to read aloud," The Johns Hopkins Univ. Elect. Eng. and Comp. Sci. Tech. Rep. JHU/EECS-86/01, 1986.

[25] T. Nakajima, H. Takizawa, M. Simamura, H. Kobayasi, and T. Nakamura, "Facial expression recognition using neural networks capable of recognizing at an infant level," WAIMH 6th Congress, 1996.

[26] L. M. Stark, M. Okajima, and G. H. Whipple, "Computer pattern recognition techniques," Comm. of the ACM, 5, pp. 527–532, 1962.

[27] E. J. Hannan, "Time series analysis," Methuen, London, 1960.

[28] H. Ogura, "Introduction to stochastic process(in Japanese)," Morikita, Japan, 1998.

[29] J. R. McDonnell and D. Waagen, "Evolving Recurrent Perceptrons for Time-Series Mod-

eling," IEEE Transactions on Neural Networks, 5(1), pp. 24–38, 1993.

[30] D. O'Shaughnessy, "Speech communication," Addison-Wesley, 1987.

[31] T. Nakajima, H. Takizawa, H. Kobayashi, and T. Nakamura, "Kohonen learning with a mechanism, the Law of the Jungle, Capable of Dealing with Nonstationary Probability Distribution Functions," IEICE trans. on inf. & sys., Vol.E81–D, No. 6, pp. 584–591, 1998.

[32] T.M. Martinetz,"Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps," Proc. of the Int. Conf. on Artificial Neural Networks, pp.427–434, 1993.

[33] S.C. Ahalt, A.K. Krishnamurthy, P. Chen, and D.E. Melton,"Competitive learning algorithms for vector quantization," Neural Networks, Vol.3, pp.277–290, 1990.

[34] D. Sieno, "Adding a conscience to competitive learning," Proc. of the Int. Conf. on Neural Networks, I, pp.117–124, July 1988.

[35] D.E.V. Bout, and T.K. Miller, "TInMANN: the integer markovian artificial neural network," Proc. of the Int. Joint Conf. on Neural Networks, pp.II205–II211, 1989.

[36] N. Ueda and R. Nakano,"A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers," Neural Networks, vol.7, No.8, pp.1211–1227, 1994.

[37] K. Yano, "A tiny dictionary of Mathmatics," Kyouritsu, Tokyo, 1968.

[38] T. W. Anderson, "The statistical analysis of time series," John Wiley, New York, 1971.

[39] G. E. P. Box, and G. M. Jenkins, "Time series analysis: forcasting and control," Holden-Day, San Francisco, 1976.

[40] Y.P. Jun, H. yoon, and J. Wan,"L* Learning: a fast self-organizing feature map learning algorithm based on incremental ordering," IEICE Trans. Inf. & Syst. vol. E76-D, no.6, pp.698–706, June 1993.

[41] M. Hagiwara,"Self-organizing feature map with a momentum term," Proc. of 1993 Int. Conf. on Neural Networks, pp.467–470, 1993.

[42] N.R. Pal, J.C. Bezdek, E.C.K. Tsao,"Generalized clustering networks and Kohonen's self-organizing scheme," IEEE Trans. Neural Networks, vol.4, pp.549–556, July 1993.

[43] B. Fritzke,"Kohonen feature maps and growing cell structures – a performance compari-

son," Advances in Neural Info. proc. systems, 5, pp.115–122, 1993.

[44] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten,"Neural-gas network for vector quantization and its application to time-series prediction," IEEE Trans. Neural Networks, vol.4, pp.558–569, July 1993.

[45] T.M. Martinetz, and K.J. Schulten,"Topology representing networks," Neural Networks, vol.7, No.3, pp.507–522, 1994.

[46] B. Fritzke,"A Growing Neural Gas Network Learns Topologies," Advances in Neural Info. proc. systems, 7, pp.625–632, 1995.

[47] J. Bruske, and G. Sommer,"Dynamic Cell Structure Learns Perfectly Topology Preserving Map," Neural Computation, vol.7, pp.845–865, 1995.

[48] T. villmann, R. Der, M. Herrmann, and T. Martinetz,"Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement," IEEE Trans. on Neural Networks, vol.8, No.2, pp.256–266, 1997.

[49] M. Kuroda, Y. Tokuda, and S. Kimura, "General psychology from Applied point of view," Yachiyo, Tokyou, 1987.

[50] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory," Nature, 222, pp. 960–962, 1969.

[51] j. j. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci., 79, pp. 2554–2558, 1982.

[52] B. Kosko, "Bidirectional associative memories," IEEE Trans. Systems, Man & Cyber., 18(1), pp. 49–60, 1988.

[53] W. Wang, and D. Lee "A modified bidirectional decoding strategy based on the BAM structure," IEEE Trans. on Neural Networks, vol.4, No.4, pp. 710–717, 1994.

Authorized Paper List

1. Journal Papers

[Trans. on IEICE D-II (in Japanese)]

(1) H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "A method for improving classification capability of multilayer perceptrons, " The trans of the institute of electronics, information and communication engineers, D-II, Vol. J80-D-II, No.1, pp. 390–393, 1997.

[IEICE trans. on inf. & sys.]

(2) T. Nakajima, H. Takizawa, H. Kobayashi, and T. Nakamura, "Kohonen learning with a mechanism, the Law of the Jungle, Capable of Dealing with Nonstationary Probability Distribution Functions," IEICE trans. on inf. & sys., Vol.E81–D, No. 6, pp. 584–591, (1998).

(3) H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "Classification Accuracy of Multilayer Perceptrons," to be published in *IEICE trans. on inf. & sys.*

(4) T. Nakajima, H. Takizawa, H. Kobayashi, and T. Nakamura, "A Topology Preserving Neural Network for Nonstationary Distributions," submitted to *IEICE trans. on inf. & sys.*

[Submitted to Neural computation]

(5) H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "A Systematic Training Data Generation Algorithm for Improving the Classification Accuracy of Multilayer Perceptrons," submitted to *Neural Computation.*

2. International Conference Papers

[WAIMH 6th Congress, 1996]

(6) T. Nakajima, H. Takizawa, M. Simamura, H. Kobayashi, and T. Nakamura, "Facial expression recognition using neural networks capable of recognizing at an infant level," WAIMH 6th Congress, pp. 66, (1996).

[IJCNN99]

(7)Taira Nakajima, Hiroyuki Takizawa, Hiroaki Kobayashi, and Tadao Nakamura, "A self-organizing network system forming memory from nonstationary probability distributions", submitted to IJCNN99.

3. National Symposium

[IPSJ Regional Symposium in Tohoku]

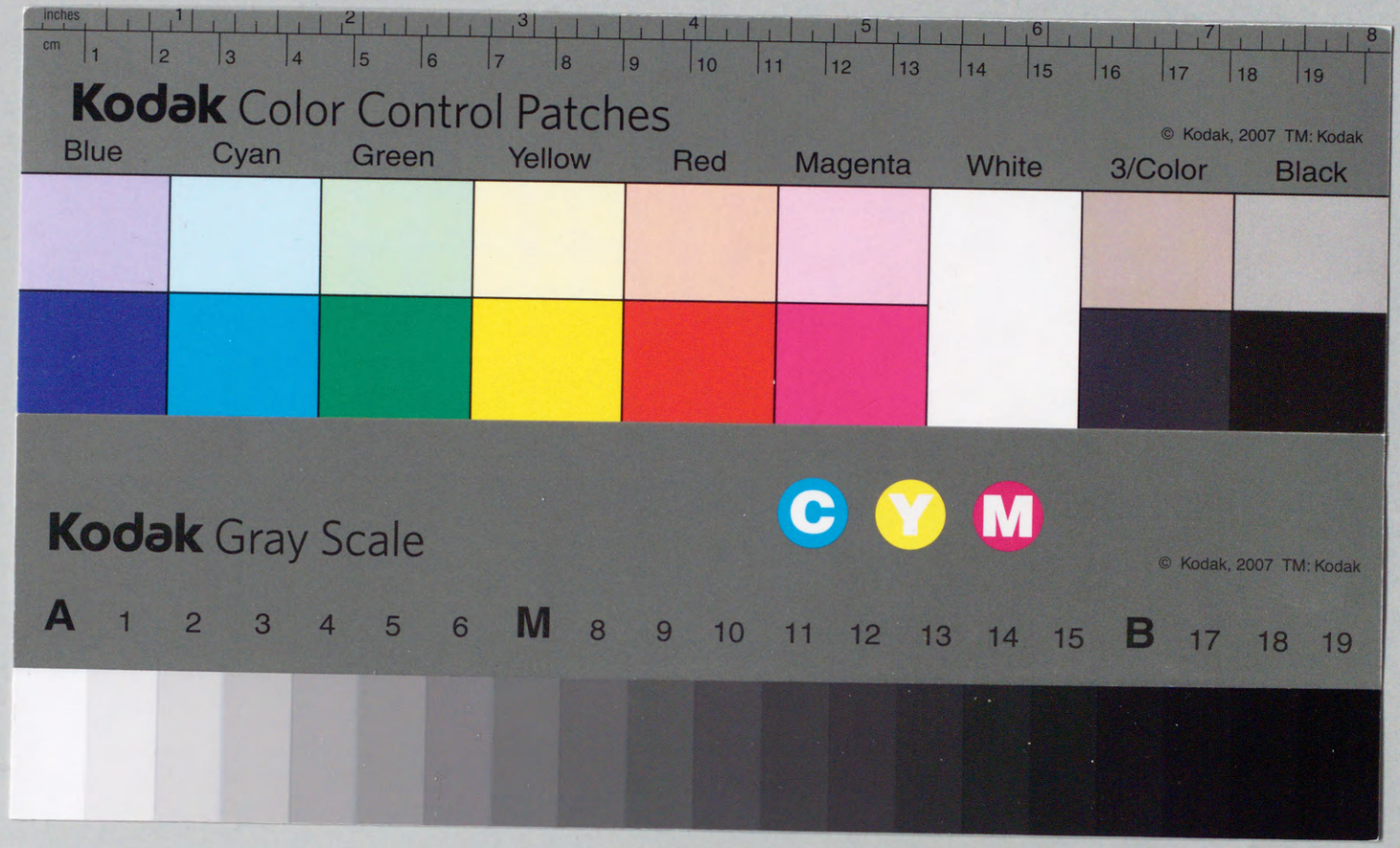
(8) H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura, "A study of optimal learning methods in neural networks," IPSJ Regional Symposium in Tohoku,(1996).

4. General Lecture Papers

[IEICE General Lecture Papers]

(9) Taira Nakajima, Hiroyuki Takizawa, Hiroaki Kobayashi, and Tadao Nakamura, "An automatic facial expression recognition system using neural networks," IEICE Technical Meetings, (1995).

(10) H. Takizawa, T. Nakajima, H. Kobayashi, and T. Nakamura,"Facial image recognition using neural networks," IEICE Technical Meetings, (1995).


inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19