論文 / 著書情報
Article / Book Information

題目(和文)	パターン認識系におけるモデルスイッチング
Title(English)	Model switching maneuvers in pattern recognition systems
著者(和文)	亀山啓輔
Author(English)	KEISUKE KAMEYAMA
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学大学院情報理工学研究科計算工学専攻, 報告番号:乙第3355号, 授与年月日:1999年9月30日, 学位の種別:論文博士, 審査員:小川英光
Citation(English)	Degree:Doctor of Engineering, Conferring organization: Tokyo Institute of Technology, Report number:乙第3355号, Conferred date:1999/9/30, Degree Type:Thesis doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Model Switching Maneuvers in Pattern Recognition Systems

Keisuke Kameyama

July 29, 1999

Acknowledgments

The author would like to express his sincerest gratitude to Professor Dr. Yukio Kosugi, for providing the circumstance and the opportunity for the work reported in this thesis. Most cordial thanks are due to Professor Dr. Hidemitsu Ogawa, who have been generous enough to be the main advisor of this thesis and have contributed so many precious ideas and advices. The author would like to thank Professor Dr. Kinji Mori, Professor Dr. Masayuki Nakajima, Professor Dr. Taisuke Sato and Associate Professor Dr. Itsuo Kumazawa who have served as advisors of this thesis. Without the help of the advisors this thesis would not have taken shape. Professor Emeritus Dr. Takuso Sato had been a great boss during the early days of research life of the author. Sincere thanks are extended to :

Mr. Tatsuo Okahashi and Mr. Morishi Izumita for their contribution in the development of the defect classification system for semiconductors in Chapter 6.

Ms. Kazuko Matsui for her ideas in developing the hybrid network in Chapter 4.

Mr. Kenzo Mori for his contribution in the development of the KM Net in Chapter 5.

Mr. Akira Hirabayashi for his tutorial on the generalization abilities of Optimally Generalizing Neural Networks.

Mrs. Toshie Asano for being a kind colleague.

Miharu for her patience, and Kaoru for her smiles.

Contents

Chapter 1
Introduction 1

1.1 Pattern recognition and supervised training 2
1.2 Layered neural networks for pattern recognition 4
1.3 The feedforward network and the backpropagation (BP) training 6
1.4 Model selection in BP trained layered neural networks for pattern recognition 7
1.4.1 Model selection and mapping ability
1.4.2 Model selection and the readiness of training by BP
1.4.3 Model selection and the cost of computation
1.5 Model alteration during training 11
1.5.1 Related works
1.5.2 Issue 1 : Computational cost and the occasion of model alteration
1.5.3 Issue 2 : Fitness of the initial map after MA
1.5.4 Aim of this work
1.6 Outline of the thesis 16

Chapter 2
Model switching 19

2.1 The definition of model switching 19
2.2 Model switching using model switching index 20
2.2.1 Stress index and the mode of switching
2.2.2 Switching operation and switching candidate
2.2.3 Fitness index
2.2.4 Model switching index
2.2.5 BP training involving MS by MS Index
2.3 Stress mode and the switching operation – scheduling of MS– 25
2.3.1 Unlimited model scheduling
2.3.2 Limited model scheduling
2.4 Selection of initial maps by Fitness Indices 28
2.4.1 Fitness by the degree of map inheritance
2.4.2 Fitness by the degree of map inheritance and the nature of class borders
2.5. Scheduling and fitness used in this thesis 29

2.6 Summary 30

Chapter 3
Model switching in BP training of Multilayer Perceptrons 31

3.1 Introduction 31
3.2 Map distance and fitness evaluation for map inheritance 32
3.2.1 Map distance
3.2.2 Map inheritance as fitness index
3.3 Model alteration by unit fusion, splitting and installation 33
3.3.1 Neural network pruning algorithms
3.3.2 The basic idea
3.3.3 Unit fusion by similarity evaluation
3.3.4 Formal analysis of unit fusion by similarity evaluation
3.3.5 Unit fusion by the variance evaluation
3.3.6 Unit splitting
3.3.7 Unit installation
3.3.8 The virtue of model switching by splitting in the BP training
3.4 Map distance by unit fusion, splitting and installation 45
3.4.1 Channel fusion revisited
3.4.2 Upper bounds of the map distance by unit fusion
3.4.3 Map distance by unit splitting and installation
3.5 Neural network pruning by unit fusion 49
3.5.1 Pruning examples
3.5.2 Discussion
3.6 BP training with MS controlled by the MS Index 56
3.6.1 Experiment – Training speed and generalization ability –
3.7 Initial map selection by unit fusion and generalization ability 61
3.7.1 Projection learning
3.7.2 Hidden layer basis functions limited by unit fusion
3.8 Smooth model switching controlled by the MS Index 65
3.8.1 Linear transformation and the modified training rule
3.8.2 Phase transition during the training
3.8.3 Experimental results
3.8.4 Discussion
3.9 Summary 73

Chapter 4
Model switching in Radial Basis Function networks and hybrid networks 74
4.1 Introduction 74
4.1.1 Pattern recognition using locally tuned kernel functions
4.1.2 MS in obtaining a reduced model with suitable class borders
4.2 MS in Hyperellipsoid clustering networks 76
4.2.1 Hyperellipsoid clustering network (HCN)
4.2.2 Fitness index for map preservation and rejection of unfamiliar inputs
4.2.3 Experiment
4.3 Model switching in hybrid layered networks 86
4.3.1 The hybrid network
4.3.2 Model switching from O2 unit to O1 unit
4.3.3 Fitness index for map preservation and selection of suitable class borders
4.3.4 Experiment
4.4 Summary 95

Chapter 5
Image Texture Segmentation Using Kernel Modifying Neural Networks 96
5.1. Introduction 96
5.2. Texture segmentation by multichannel filtering 98
5.3. The Kernel Modifying Neural Network 101
5.3.1 Architecture
5.3.2 Training
5.3.3 Test Phase
5.4. Experiments and results 105
5.4.1 Texture region extraction from a computer generated image
5.4.2 Segmentation of common textures
5.4.3 Tissue classification in an ultrasonic echo scan image
5.4.4. Discussion
5.5 Extraction of local phase information 113
5.5.1 Limitations of the Gabor filter feature extractors
5.5.2 Higher-order statistics for extraction of phase information
5.6 Bispectral Kernel Modifying Neural Network (BKM Net) 117
5.6.1 Architecture

5.6.2 Training rule
5.7 Application of the BKM Net and discussion 119
5.7.1 Texture region extraction - Comparison with the KM Net -
5.7.2 Segmentation of a computer generated image
5.7.3 Discussion
5.8 Model switching in the Gabor layer 122
5.8.1 The inefficiency in the Gabor layer training
5.8.2 Model switching by unit fusion in the limited model
5.8.3 Experiment
5.9 Summary 127

Chapter 6
Defect Classification in Visual Inspection of Semiconductors 129
6.1 Introduction 129
6.2 Automatic defect classifier system (ADC) 131
6.2.1 Defect classes
6.2.2 Detection of defects
6.2.3 Determination of the defect region
6.2.4 Feature extraction
6.2.5 Neural network classifiers
6.3 Experiment 139
6.3.1 Classification with a small training set
6.3.2 Classification with a larger training set
6.4 Discussion 144
6.5 Summary 145

Chapter 7
Conclusion 146

Appendix A
Modification of the Gabor kernel by the Kernel Modifying Network's training. 148
Appendix B
Modification of the Gabor kernel by the Bispectral Kernel Modifying Network's training. 149
References 152
Publications

Chapter 1
Introduction

In model-based pattern recognition, model selection has been one of the major issues in applying them to real world problems. Unsuitable selection of the model can result in a total failure of implementation of the pattern recognition system.

Since the 1980's, research on biologically inspired parallel computing systems generally known as neural networks have been widely attempted. Among them of special interest is the class of layered neural networks, which fundamentally is a vector to vector map, trainable with a set of ideal input-output pair samples. Due to this nature, use of layered neural networks for pattern recognition have been reported in many literature. The existence of the backpropagation (BP) training rule, which is in essence a gradient descent optimization of the network parameters, greatly contributed to this boom.

The issue of model selection, however, remains unsolved in the use of neural networks for pattern recognition, now embodying as the problem of choosing the best activation functions of the units, number of units and layers. As it will be shown in this chapter, the requirements from various aspects of learning, such as mapping ability, readiness of training by BP and computational cost, are often contradictory. This contradiction cannot be solved as long as the network, which defines the function space for searching and mapping, is limited to a single model. In order to solve this contradiction, method for Model Alteration (MA) during the stepwise learning process have been sought by many researchers. On alteration of the model, (1) determination of the *instant* of alteration, (2) selection of the *new model* and (3) selection of the *initial map* within the new model, are all important factors for a successful learning process. However, in the conventional training methods that involves MA, these factors were considered separately, thereby causing inefficiencies in the training process as a whole.

In this thesis, a novel scheme for simultaneous determination of the three factors listed above, named *Model Switching* (MS) will be introduced. Network training with MS can be considered to be a process of searching the model and the map simultaneously. Throughout this thesis, various aspects of BP training employing MS will be examined. It will be shown that BP training with MS is an efficient learning process for obtaining a trained network in a small model. This main feature of MS owes itself to the model alteration controlling by a factor named the *Model Switching Index* (MS Index). By using the MS Index, suitable instant for altering the model is determined in accordance to the progress of learning, for efficient training.

1.1 Pattern recognition and supervised training

Construction of a pattern recognition system by supervised learning can be formalized as an acquisition of a (true) mapping function $\boldsymbol{y} = f(\boldsymbol{x})$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and the output vectors. The supervised learning in doing this is often a parametric approach, which assumes a *model* represented by a function $\boldsymbol{o} = f_p(\boldsymbol{x}; \boldsymbol{w})$, and tries to approximate f by altering the parameter vector $\boldsymbol{w}$. Throughout this thesis, the word *model* will be used as being equivalent to the function space which can be realized by changing the parameter. Also, *model size* denotes the number of freely changable parameters. Model size will also denote the upper bound of the dimension of the model. In selecting the favorable setting of parameters, a set of ideal input-output pairs (training set) $\{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$ will be used. Basically, the requirement to the map consists of the calculation of the class probability density function (PDF) and application of a decision rule [3], but other means accomplishing equivalent results such as direct estimation of the decision border in the feature space, may be acceptable.

When a suitable function class which f belongs to is known in beforehand, a model F_p within the class can be selected, and with an appropriate selection of the training set, the best parameter vector may be obtained in a rigorous manner.

When the class of the true map f is unknown, an approximating function $f_p \in F_p$ must be selected from a class of functions that possibly include the true map f. If the ability of the selected function class is sufficient, generally the approximation map f_p can be tuned to an arbitrary closeness to the true map f measured by a certain distance metric, provided that a training set of sufficient size can be selected at will.

In most cases of real world applications, however, the function class which f belongs to is unknown, and the size and the selection of the training set are limited. In these cases, all we can do is to predetermine the model F_p, tune the parameter vector so that $f_p(\boldsymbol{x}; \boldsymbol{w}) \in F_p$ fulfills the mapping relation at the sample points in the training set. Naturally, the approximation function can differ in accordance to the model selection. As an imaginary example, Fig. 1.1 shows two different classification functions derived by a same training set and a different model selection.

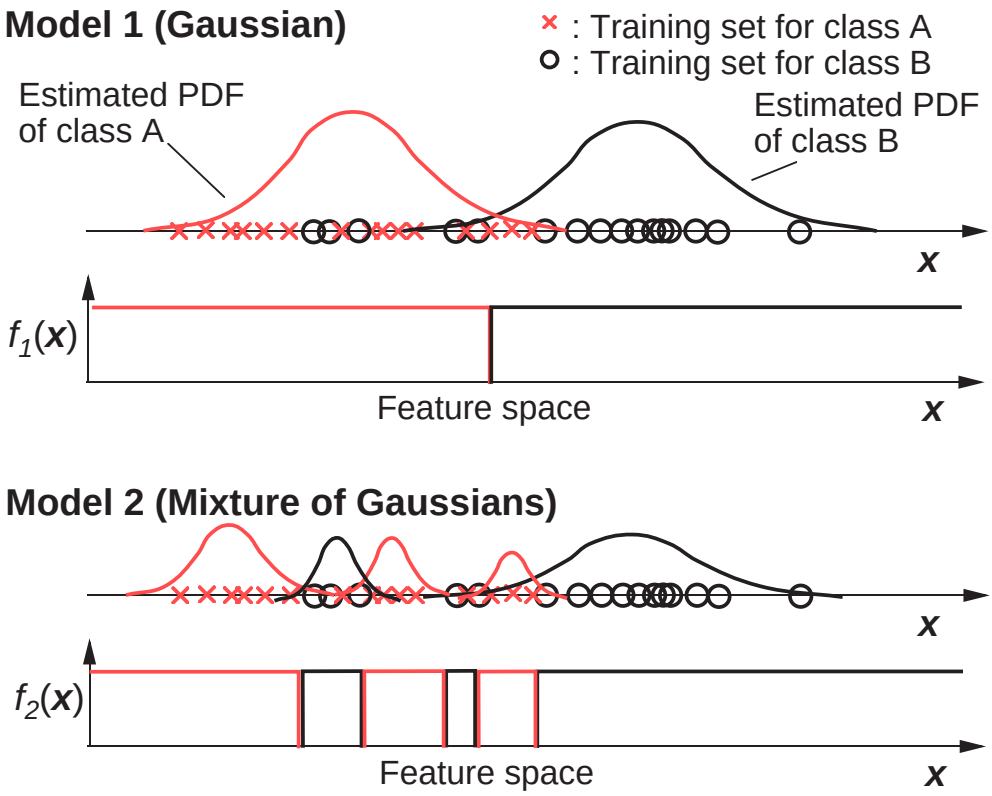


Fig. 1.1. An example of the differences in the estimated approximation function due to different model selection. Model 1 tries to model the distributions of each class with a single Gaussian distribution, whereas Model 2 tries to model them as mixtures of Gaussian distributions. The classification functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ differ especially in the marginal regions of the two classes.

Selection of the model can affect various aspects of the pattern recognition machine. First, the selection of the model is selecting the *mapping ability* of the possible set of functions. If the model size is set too small, there will be training sets that are not mappable by the function $f_p(\boldsymbol{x}; \boldsymbol{w})$. In the example of Fig. 1.1, the training set is not linearly separable, and therefore assuming one Gaussian distribution per class cannot map the training set relation.

Second, model selection can affect the *generalization ability* of the pattern recognition machine, namely a desired ability to correctly classify an unseen input. Even when a suitable model is intuitively determined, the parameter vector $\boldsymbol{w}$ which fulfils the error condition can be nonunique, and the inverse problem of obtaining the approximation function from the training set can become ill-posed.

In determining a particular function $f_p(\boldsymbol{x}; \boldsymbol{w})$, training methods that successively update the parameter vector $\boldsymbol{w}$ to minimize an *error function* of the training set are often used. The method is especially used when the estimation of the class PDFs or the class borders is not easy. This process

is also known as *supervised training*, as opposed to *unsupervised training* where true outputs are not provided by the oracle. The error function is defined as a discrepancy between the true function f and its approximation $f_p(\boldsymbol{x}; \boldsymbol{w})$. When no additional information for the true function f is available besides the training set $\{\boldsymbol{x}_m\}_{m=1}^{M}$, definition of,

$$E(\boldsymbol{w}) = \sum_{m=1}^{M} ||f(\boldsymbol{x}_m) - f_p(\boldsymbol{x}_m, \boldsymbol{w})||^2, \tag{1.1}$$

is often used as the error function.

A popular and very general formalism among the learning algorithms is the gradient descent method, which iteratively updates the parameter vector by a finite step as,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \frac{\partial E}{\partial \boldsymbol{w}}, \tag{1.2}$$

where t and η denote the time and the training gain constant, respectively. When the weights are not altered by the training after repeating the updating process of Eq. (1.2), or when the training error is smaller than a certain threshold, the training process will be stopped.

Employment of this stepwise training process introduces a third factor in model selection. It is known through experiments that the stepwise training process can fail by being caught in a so-called *local minima* of the error function. The rate of successful training can certainly vary with the model size, however, no general relation between the model size and the rate of successful stepwise training is known so far.

When implementing the pattern recognition system, the *computational cost* of training the system and the cost of using it will certainly be another concern. As the computational cost tend to grow with the size of the employed model, smaller models are desired from this aspect.

This thesis mainly focuses to the dynamic selection of the model to fulfill the requirements of mapping ability, computational cost and efficiency of the gradient-based training.

1.2 Layered neural networks for pattern recognition

Layered neural networks consist of several layers of units named neurons as shown in Fig. 1.2. Between every pair of units in the adjacent layers, there is a unidirectional link for the response of a unit to be transmitted to the other unit in the next layer. However, no intra-layer connections exist. It is also known as *feedforward neural networks* for there are no cyclic route for the signal [23][24].

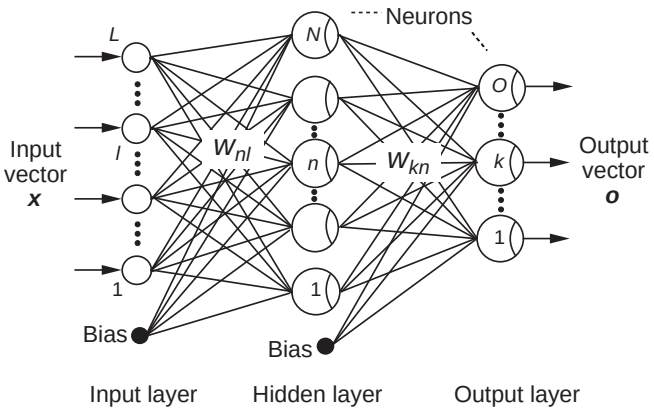


Fig. 1.2. A two layered perceptron network, commonly known as a three layered neural network.

The layered network can be regarded as implementing a vector function of $\boldsymbol{o} = f_N(\boldsymbol{x}; \boldsymbol{w})$ with $\boldsymbol{w}$ being the vector of all the tunable connection weights. By altering the weight vector $\boldsymbol{w}$, the network can implement various maps. It has been proved that several models of layered networks are universal approximators [7][8][20], meaning that the approximation function can be tuned to an arbitrary closeness to the target function f within a closed domain, under certain conditions. By using a training set as discussed above, the network can be trained to mimic the desired relation of $\boldsymbol{x}$ and $\boldsymbol{y}$ by selecting the suitable weight vector. The weight vector $\boldsymbol{w}$ can be determined by various means; one of the most common methods is the backpropagation (BP) training rule [5] which practically implements the gradient descent learning addressed above. Due to the tunable mapping nature and the existence of the training rule, the layered neural network is very commonly used as a trainable pattern recognition machine. In this thesis, layered neural networks trained with the BP rule for pattern recognition will be extensively discussed. The details of the BP training will be reviewed in Chapter 3.

The issue of model selection renders itself as the problem of choosing the number of layers, the number of units within each layer, and the type of unit activation functions. These factors all affect the mapping ability, the readiness of BP training, and the computational cost for training and using the network. The requirements from these conditions tend to be contradictory, which makes the system with predetermined function model sub-optimal at best. The main aim of this thesis is to construct a framework of efficient BP training of layered neural networks without being confined to a single function model. The relation between the network model and the three factors listed above will be discussed in the following sections.

1.3 The feedforward network and the backpropagation (BP) training [5]

Here, the backpropagation (BP) training rule for layered neural networks will be briefly reviewed for the clarity of the definitions.

The network configuration used in this chapter is a so-called three layered network, which consists of two layers of perceptrons [2][5] and an input latch layer as illustrated in Fig. 1.2. A unit in a layer has a full unidirectional connection to the units in the neighboring layer, however, no connections exist within the layer, or between the units of non-adjacent layers. Each connection link has an attribute called the *weight* which is a multiplied factor to the unit output transmitted to the connected units.

In an i-th network unit (neuron), the input signals from the other units are combined as a weighted sum as

$$u_i = \sum_{connected\ j} w_{ij} o_j , \tag{1.3}$$

where u_i, w_{ij} and o_j denote the unit *potential*, the weight of the link from unit j to i, and the output of unit j, respectively. Unit i is activated through a unipolar sigmoid step function with (0, 1) output range to produce an output of,

$$o_i = s(u_i) = \{1 + exp(-u_i)\}^{-1} . \tag{1.4}$$

Sometimes, a function with a bipolar range of (–1, 1), such as,

$$s(u_i) = \tanh(u_i) , \tag{1.5}$$

is used.

The input vector $\boldsymbol{x}$ presented at the input layer will be fed forward to obtain the network output $\boldsymbol{o}$, thereby enabling to use the network as a vector-to-vector map.

The desired map will be obtained through training the network, by presenting the training patterns and calibrating the connection weights according to the BP rule. A set of ideal input-output pairs $\{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$ are used as the training patterns, where M denotes the number of training pairs. The training *error* E is defined as

$$E = \frac{1}{M} \sum_{m=1}^{M} E_m = \frac{1}{2MO} \sum_{m=1}^{M} \sum_{k=1}^{O} (y_{km} - o_{km})^2, \tag{1.6}$$

which is a sum of the errors for each training set E_m. In Eq. (1.6), o_{km} and O denote the response of the k-th unit to the m-th training input, and the number of the output units, respectively.

A training *epoch* is a single round of presentations of all the training sets to the network. Throughout this work, updating of the weights occur only once per epoch, which is a method typically known as the *batch updating* [5]. The weight update strategy of the BP rule is equivalent to the gradient descent optimization method. Therefore, the update measure for each epoch can be written as,

$$\Delta w_{ij} = \sum_{m=1}^{M} \Delta w_{ijm} = -\eta \sum_{m=1}^{M} \frac{\partial E_m}{\partial w_{ij}} = -\eta \sum_{m=1}^{M} (\delta_{im} o_{jm}), \quad (1.7)$$

where, η is the training gain constant and δ_{im} is the *backpropagated error* defined as,

$$\delta_{im} = (y_{im} - o_{im}) s'_i(u_{im}), \quad (1.8)$$

when the unit *i* is the output unit with *s'* being the derivative of the transfer function and,

$$\delta_{im} = s'_i(u_{im}) \sum_k \delta_{km} w_{ki}, \quad (1.9)$$

when the unit *i* is a hidden layer unit. The summation in the right hand side. of Eq. (1.9) is extended over all the units in the next layer, that are linked with the unit *i*.

1.4 Model selection in BP trained layered neural networks for pattern recognition

In this section, the relations between the network model and several factors in building and using the network for pattern recognition will be discussed in detail. First, the notation of the model and the model size will be formally defined.

Definition (Model)

Let us put the function of the network as , $\boldsymbol{o} = f_N(\boldsymbol{x}; \boldsymbol{w})$, where $\boldsymbol{x}$, $\boldsymbol{o}$ and $\boldsymbol{w}$ denote the input, output and the parameter vectors, respectively. The *model* of the network signifies the space of functions F_N achievable by altering the parameter vector $\boldsymbol{w}$.

Definition (Model Size)

The *model size* denotes the number of independently changable parameters in the network. If all the parameters in parameter vector $\boldsymbol{w}$ are independent, model size will be identical to the dimension of $\boldsymbol{w}$.

1.4.1 Model selection and mapping ability

In a layered network intended to be used for pattern recognition, the number of input and output layer units will be determined by the problem. The number of hidden layers, the number of units within each layer, and the types of functions has to be determined in beforehand, which is equivalent to selecting the function model. In the following discussion, the number of hidden layer is assumed to be one for simplicity.

The set of maps $\{f_N(\boldsymbol{x}; \boldsymbol{w})\}$ which the network can implement is determined by the network model. A network with insufficient number of hidden units cannot implement the map to satisfy the training error criterion. This nature is irrelevant with the method of training. A well known example is the XOR problem, which cannot be implemented with a single perceptron, or equivalently, a three-

layered network with only one hidden layer unit employing a linear discriminant [2]. The abilities of the so-called three-layered networks with sigmoidal activation functions have been investigated in [7] and [8]. In a three-layered network with sigmoidal activation functions, the issue of model selection will reduce to the determination of the number of hidden layer units. There, it has been proven that the class of networks can approximate any continuous map to arbitrary accuracy, when sufficient number of hidden layer neurons are available. Generally, it can be said that the model size (and thus the ability) will be expanded as more hidden layer units are used.

1.4.2 Model selection and the readiness of training by BP

A. Local minima

Since the basic BP training depends on the local gradient for modification of the weight vectors, it can get captured in a local minimum of the error potential function. The issue of being captured in local minima is reported to be avoidable by using batch-mode BP training in most real world problems [5]. However, opposite claims exist for the case when the structure of the error potential function becomes complicated [2][24]. The selection of the model can affect the readiness of training by BP. As the network size grows, so will the combinatoric permutations of the weights that makes the network to act identically [23]. Therefore, the number of the global minima will increase, and it may improve the possibility of reaching one of them by training. However, as the error surface will be formed by more basins attracting the training to their minima, it may increase the number of local minima as well, as a side effect. The general relation between the size of the model and the existence of the local minima is still unknown.

Baldi and Hornik have proved that three-layered networks with units employing linear transfer functions can have minima in the error potential function only at points where the training error is zero [10]. They showed that other points in the weight space that have zero gradient are all saddle points.

Gori and Tesi have reported in [11], a more general investigation for the networks with nonlinear transfer functions which applies to the case of this work. One of their results show the conditions for the layered network configuration and the training set so that the gradient based training can be free of local minima. The conditions will be briefly presented below. Although they discuss for the networks with arbitrary number of layers, it will be presented for the case of three-layered networks employing a unipolar sigmoid function of (0, 1) range.

Notations :

L, N, O	: Numbers of input, hidden and output layer units, respectively.
M	: Number of training patterns.
$Y \in R^M \times R^O$	: Training output matrix $[y_{mo}]$ ($m = 1,2, \ldots, M$, $o = 1,2, \ldots, O$) .
$W_0 \in R^N \times R^{L+1}$	: Connection weight matrix between the input and the output layers.

$W_1 \in R^O \times R^{N+1}$	: Connection weight matrix between the hidden and the output layers.
$X_0 \in R^M \times R^{L+1}$	: Matrix whose row vectors are the training input including the bias channel (input layer trace matrix).
$X_1 \in R^M \times R^{N+1}$	: Matrix whose row vectors are the hidden layer responses including the bias channel (hidden layer response trace matrix).
$X_2 \in R^M \times R^O$	: Matrix whose row vectors are the responses of the output layer units (output layer response trace matrix).
$D_1 \in R^M \times R^N$	: Matrix $[\partial E_m / \partial u_n(m)]$ (m = 1,2, ..., M, n = 1,2, ..., N).
$D_2 \in R^M \times R^O$	: Matrix $[\partial E_m / \partial u_o(m)]$ (m = 1,2, ..., M, o = 1,2, ..., O).
E_m	: The output error for the m-th training pattern.
$u_n(m)$	: The unit potential of the n-th hidden unit for the m-th training pattern.
$u_o(m)$	: The unit potential of the o-th output unit for the m-th training pattern.
S^{D1}	: Set of all possible D_1 generated by varying the weights in weight space.

Theorem [11]

Gradient descent leads to the global minimum if the layered network and its training set satisfy the following conditions :

(1) $N \leq H$ (Pyramidal hypothesis).

(2) Matrix W_1 is a full row rank matrix.

(3) $N(X_0) \cap S^{D_1} = \{0\}$.

(4) $y_{mn} \in \{0, 1\}$ (m = 1, ..., M, n = 1, ..., N)

The proof will be omitted here.

The authors of [11] show that in particular networks and the training sets satisfying the four conditions above, all the points that have zero gradient are saddle points, except for the global minimum. It is clear however, that the conditions in the above Theorem is often violated. Many applications employ so-called *bottleneck networks*; namely networks with a hidden layer which is smaller than the output layer. Networks with bottlenecks trained for autoassociative mappings are especially important for their abilities of compression and both linear and nonlinear principal component analyses of the training data by the BP learning [10][12]. Since the weight matrix can be considered to be changing in random, the second condition usually holds. The third condition is required in essence to prevent the gradient matrix G_0 calculated as,

$$G_0^T = D_1^T X_0 , \tag{1.10}$$

from becoming a zero matrix albeit D_1 is a nonzero matrix. Here, T denotes the transpose of the matrix. The obvious cure to this situation is to keep the input patterns linearly independent by setting $L \geq M$ [11], however this requirement is not practical, since in most cases there are more training patterns (M) than the number of input channels (L).

B. The herd effect (Inefficiency of feature extraction)

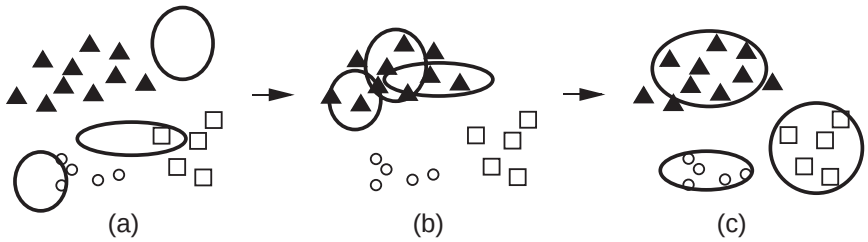


Fig. 1.3. The herd effect during BP training. Small legends denote the three-classes of training sets in the input space. The large ovals denote the local regions that the hidden layer basis functions are tuned to. (a) Initial state before BP training. (b) When the BP training is started, all the hidden layer units are tuned to one significant class, possibly falling into a local minima of the error function. (Herd effect) (c) Accomplishment of the training.

The inefficiency named the *herd effect* in the BP training process has been pointed out in [44]. During the learning process, it would be ideal if the hidden layer units could divide the whole feature extraction task automatically, and take up each part for the accomplishment of the whole. However, since there are no lateral connections between the units in the hidden layer, the error information which each hidden layer unit receives is more or less the same. Consequently, especially at the initial stage of training, it often happens that all the hidden layer units try to capture the most significant feature; acting as a herd of sheep. Later on, the error function target will move due to this initial modifications to the weights, and the herd will split up as several among them will change to capture the remaining feature. This process is usually very time consuming compared to the case when there are lateral connections between the hidden layer units for avoiding this inefficiency [45].

When there are only minimal number of feature extractors, e.g. two features *A* and *B* to be extracted by two hidden units *i* and *j*, the herd effect can even cause the training to be captured in a local minima. After both units *i* and *j* have tuned themselves to feature *A*, either of them can be converted to capture the remaining feature *B*. However, the conversion can cause a temporal increase in the error, because feature *A* is jointly took upon by the two units at this moment. Even when either of the units could be converted to feature *B*, the process of the conversion is very time consuming, especially when it requires a large amount of change in the weight values. This type of failure and inefficiency can be avoided by using several redundant hidden units in the course of training.

In turn, when the number of hidden layer units are increased to make the network far redundant than the minimal model for learning the map relation of the training sets, the training will slow down again. This time, the error will not converge easily since many feature extractors change at once, causing in a drastic change in the landscape of the error function. This phenomenon is mainly due to the limited step-size of the weight modification [44].

1.4.3 Model selection and the cost of computation

In view of the computational cost of using the network, a minimal model that can accomplish the desired mapping is favored. The cost for both training and using the network will be in the order of $O(n)$, where n denotes the number of the hidden layer units in a three layered network with full connections between the layers.

Among other approaches for determining the model are use of Genetic Algorithms (GA) [47]-[49]. In these approaches, the structure of the network is encoded in the gene, and at each generation, the networks are trained from random state to evaluate their fitnesses. Therefore these approaches require a vast number of trial and error to find the best model in their own criteria, and the majority of the training attempts will simply be discarded in the process of obtaining the best model. In the framework of Neurogenetic Learning by Kitano [49], an improvement has been made to boost the training speed by inheriting the distribution of the random weights. However, the actual maps of the trained networks are not inherited through the generations.

1.5 Model alteration during training

The relations between the model size and the factors discussed in the former section, are summarized in Fig. 1.4. For satisfying the requirements, network training by BP should be conducted in an easily trained model, and always maintaining a small model for reducing the computational cost whenever possible. Further, in order to solve the unpredictable local minima problem, it would be insufficient to merely reduce the model size. Therefore, a method for reducing and increasing the model size without altering the acquired map will be necessary, together with its controlling method.

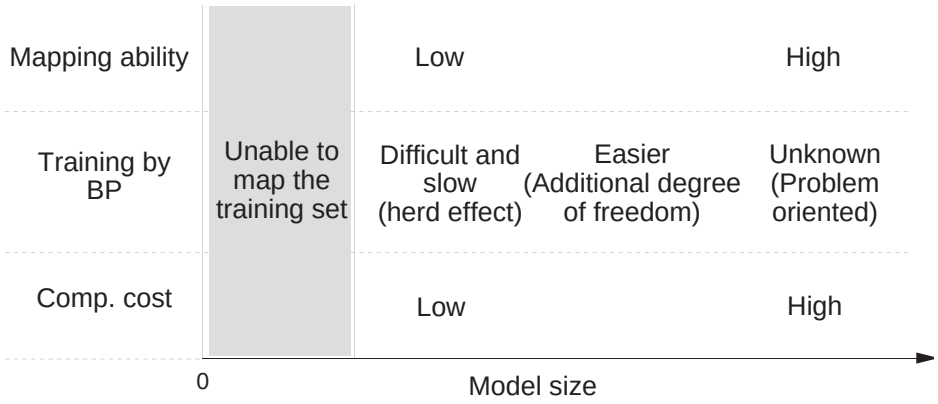


Fig. 1.4. The characteristics of the network models viewed from the readiness of training by BP, and the computational cost of training and using them.

1.5.1 Related works

In the literature, there are several different approaches for solving the conflicting requirements.

On reducing the model size of a network, an idea generally called *pruning* [30]-[38] have been pursued. After training a large network, pruning tries to remove the irrelevant links, units and layers in the network according to several detection criteria. Although the features of several pruning methods will be discussed in Chapter 3, the basic philosophy of pruning is to remove the unessential factors from the network.

There are methods that *grow* the network to gain the required ability while training. In [43] and [44], an architecture called the Cascade-Correlation Network is introduced whose units are installed one by one, connected in a cascade manner to form a deep structure. An attempt to connect the perceptron in a decision making tree structure appears in [46] for application to medical image segmentation. Increasing the basis function unit in the Radial Basis function (RBF) nets are tried for clustering problems in [21]. However, simple growing of networks can sometimes result in excessively large structures.

Hirose et al. report a BP training algorithm which adds a hidden layer unit by detecting the situation while the training is caught in a local minimum [39]. After the convergence of the error, the hidden layer units are removed in the last-come-first-go order. Such an approach which alters the function model in both directions is very attractive, in that the function model itself can be learned by an adaptive procedure during the learning.

1.5.2 Issue 1 : Computational cost and the occasion of model alteration

Training speed and necessary resource are always major concern in training and using neural networks, because most applications are implemented in a conventional computer simulating the parallel action of the network. The reported BP training methods that involve MA, has several factors that can be improved, with regards the training speed and the required computational resource such as memory.

All the MA methods in the literature so far, evokes the MA procedure when the error converges to a minimum of the error potential function. Namely, model size reduction is attempted after the error convergence, and expansion is attempted when the training is caught in a local minima of the error function. However, this restriction can make the whole training process a computationally costly one.

It is clear that it will be beneficial when MA with map inheritance is allowed in broader occasions during the BP training. For example, when training error of the network of a decent model size is making a steep decrease of the error, it can be foreseen that the error will eventually converge, and model size reduction will take place. In such a case, alteration of the model to a smaller one should take place before the convergence of the error. On the other hand, when the decrease of the error is very slow, the training can often be accelerated by extending the model size.

On using the trained network a smaller model is always favored as noted in the former

section. In Fig. 1.5, a schematic of error convergence and the change in the model size are compared for the conventional BP training with MA and the aimed method. On allowing MA at various occasions of training, the occasion should be carefully selected in accordance to the nature of the initial map which is the starting point of further training in the new model. Otherwise, the alteration may even be harmful to the whole training process.

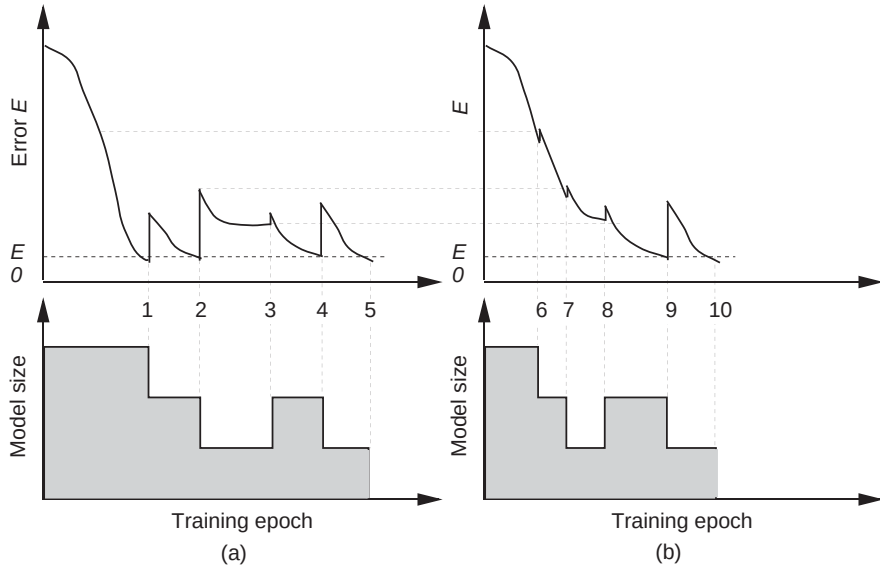


Fig. 1.5. The training error convergence and the change in the model size for (a) conventional BP training with MA and (b) the aimed method. 1 and 2: Error converges to the target value E_0 and model size is reduced. 3: The training is caught in a local minima, and the model is re-expanded. 4: training further proceeds by detouring the local minima. 5: final convergence to the global minimum in the small network model. 6 and 7: Model size is reduced by detecting a fast convergence. 8: slowdown due to possible local minima is detected and the model is expanded. 9 and 10: escape from the local minima and final convergence to the global minimum in the small network model. The area of the shaded region reflects the computational cost of the whole training process.

1.5.3 Issue 2 : Fitness of the initial map after MA

As noted in Issue 1, the occasion of MA should be carefully selected by the nature of the initial maps among the possible candidates. There are two reasons for this. The first is because the time that the whole training process will take, should be kept minimal. One good practice that allows an efficient MA is the inheritance of the acquired map at the point of MA, by the new model. The second reason is because there are other desired natures to the new map. Even when there is only one candidate of the new model, there can be several candidates of the initial maps within the new model, and they may be rated differently from the additional requirements. These factors of the new initial map including the inheritance of the map and other natures, will be referred to as the *fitness*.

When the fitness of the new map is to be evaluated by how the new map inherits the map of

the immediate model, a model and a map which perserves the map most will be selected. When the discrepancies of the maps at each MA can be ignored, the training process can be illlustrated as in Fig. 1.6. In this case, the whole training process can be made very efficient, because no additional compensation (e.g. training) for the loss of the learned information will be necessary.

Extension of the model size is a trivial operation from the viewpoint of map preservation, because a larger model often includes the smaller one. In other words, it is usually possible to add a weight or a unit without affecting the map. This operation which add a hidden unit will be referred to as *unit installation.*

On reduction of the model size, many methods that involve pruning of connection weights and units stand in a similar view of map preservation. Among these are, pruning by evaluating the sensitivity of weight removal by Karnin [31], Optimal Brain Damage (OBD) by Le Cun et al.[35], and Optimal Brain Surgeon (OBS) by Hassibi et al. [36]. Selection of removable elements by sensitivity assessment tries to suppress the increase of the error due to the model alteration, however, no rigorous compensation measures are taken to the remaining network for map preservation. Also, the sensitivity measure does not always reflect the importance of the removed factor in the network map. For example, a unit with constant response to all the training inputs, is not essential when there is a bias unit in the same layer. However, such a unit will not always be selected as a candidate of removal in sensitivity analysis. OBD and OBS both trying to approximate the minima in the error potential function with a simplified quadratic function, are known to be effective for drastic reduction in the model size of a large network. However, the methods are only applicable after the training converges to one of the local the minima of the error potential function, which greatly limits the occasion which the model can be altered.

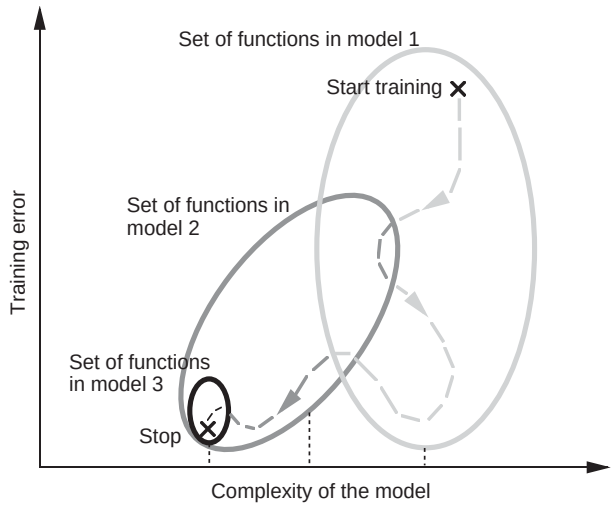


Fig. 1.6. The scheme of training and model switching. The training is started in model 1 which is large, and reaches the final function of a smaller model 3 via model 2. Although the models are changed during the progress of training, the path is continuous in the function space.

There can be various other possibilities of the desired natures of the new model and the initial map that can be used as the fitness. Among them are, the generalization ability, the smoothness of the map and the types of classification borders that the network will form.

In this work, novel methods for MA named *unit fusion* and *unit splitting* for reducing and expanding the model size will be introduced in chapter 3. The methods enable a map-preserving reduction and addition of hidden layer units that are applicable at any instant during BP training. In chapter 4, other types of fitnesses that evaluate the type of classification borders of the new model have been tried for altering the network to model the given distribution of the class clusters adequetely and efficiently.

1.5.4 Aim of this work

As noted above, when extending the occasions of model alteration as suggested above, determination of the occasion of model alteration, the new model, and the initial map in the new model will be especially important. In fact these three factors are mutually dependent and closely related because the pair of suitable model and initial map will change in accordance to the progress of training and the existence of possible initial map candidate at each moment of BP training.

The aim of this work is to build a framework of BP training involving model alteration at arbitrary occasions observing the fitness of the target model and map, for the efficiency of the whole training process and higher fitness of the final network model and map.

1.6 Outline of the thesis

This thesis consists of seven chapters. The structure is shown in Fig. 1.7.

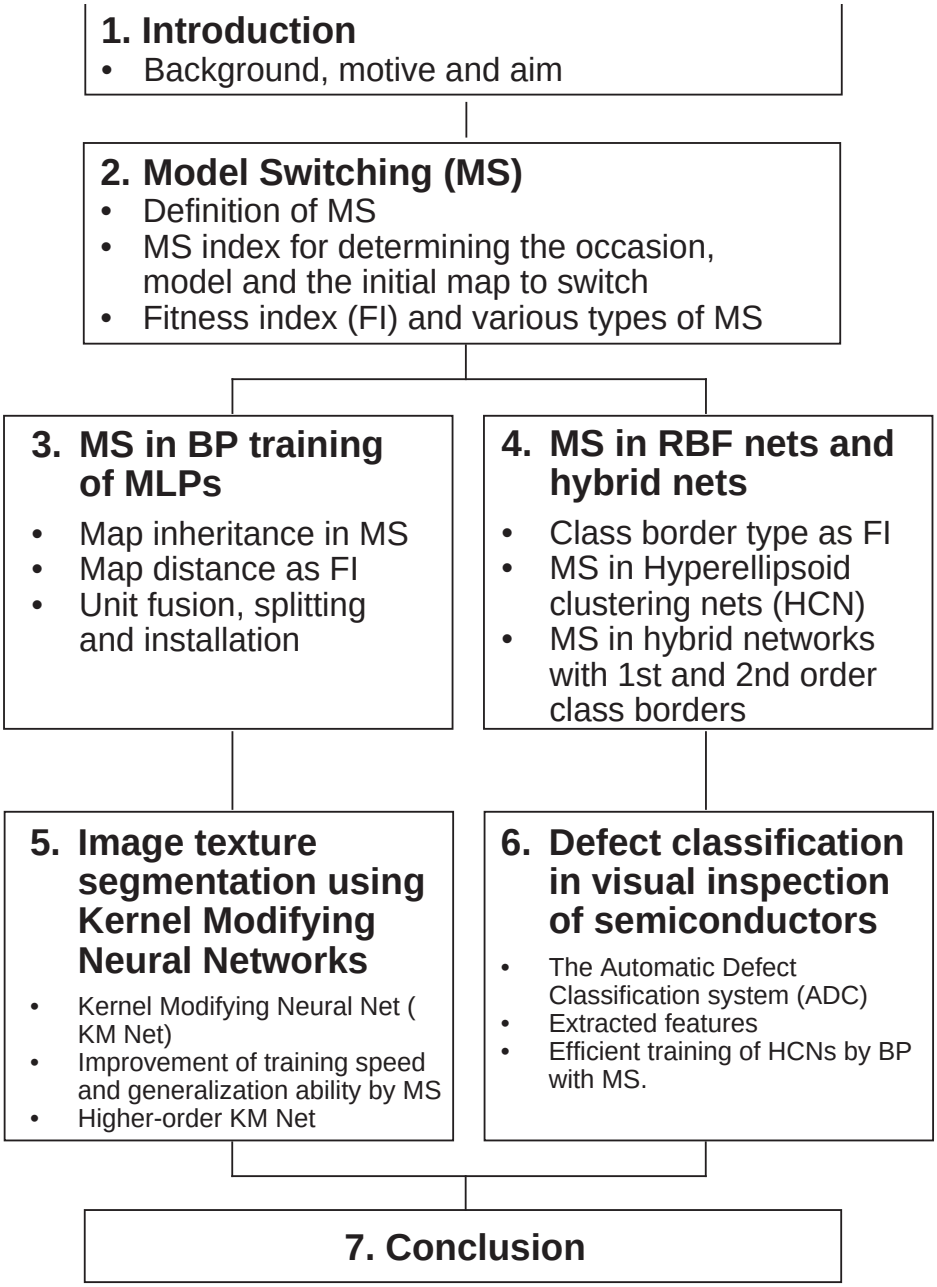


Fig. 1.7. The structure of this thesis.

In Chapter 2 “Model Switching”, the general idea of an operation named *Model Switching* (MS) for enabling the simultaneous determination of the occasion, the model and the initial map will be introduced. After giving a strict definition of MS, the actual method of MS consulting an index named the *Model Switching Index* (MS Index) will be introduced. This MS Index is a function of the training error, the time derivative of the training error and a measure named *fitness index* which reflects the suitability of the new initial map candidate at each moment of training. It will be discussed that there are various types of possible fitness indices. As it will be shown in the following chapters,

MS of different nature will be possible in accordance with the selection of the fitness indices.

In Chapter 3 “Model switching in BP training of Multilayer Perceptrons”, a type of MS which uses the *map distance* between the two maps before and after switching as the fitness index, will be introduced. This fitness index of this type guarantees that the map acquired to the point of switching will be inherited by the new model. In order to implement this type of MS to Multilayer Perceptrons (MLPs), *fusion*, *splitting* and *installation* of the hidden layer units will be introduced as methods for determining the new model and the initial map within the new model. Further, theorems that enable to obtain approximated map distances with less calculation will be given, and the BP training method employing MS with approximated map distances set as fitness indices will be defined. By applying the training method to an encoder problem, the importance of map inheritance in MS will be shown. Also, it will be shown that a small trained network with high generalization ability can be obtained by using this training method, by applying to a character recognition problem.

In Chapter 4 “Model switching in Radial Basis Function networks and hybrid networks”, MS using another type of fitness index will be introduced. Here, in addition to the map distance, the fitness index also evaluates the nature of the class discrimination borders which will be determined by the initial map in the new model. In the first half of this chapter, BP training with MS will be defined for the Hyperellipsoid Clustering Network (HCN), which is a member of networks employing Radial Basis Functions (RBFs). It will be shown that a classifier can be constructed, which points out the inputs that are distant from known training sets to be originating from an unknown class, in a small model having far less hidden layer units in comparison to the size of the training set. In the latter half of this chapter, BP training with MS will be defined for *hybrid* networks having different types of hidden layer units making first and second order discrimination borders. It will be shown that the hybrid networks trained by the proposed training method can achieve higher generalization, when compared with networks having homogeneous hidden layers.

In Chapter 5 “Image Texture Segmentation Using Kernel Modifying Neural Networks”, the BP training with MS introduced in Chapter 3 will be applied to networks for image texture classification. First, the *Kernel Modifying Neural Network* model which unifies the feature extraction filter array and the classification MLP in a single network for texture classification will be introduced. Then it will be shown that the KM Net can automatically extract the features that are necessary for texture classification. By applying the BP training with MS introduced in chapter 3, it will be experimentally shown that the network can be trained more efficiently when compared with the case without MS. Also, it will be shown that the generalization ability can be improved by use of MS. Next, the *Higher-order KM Nets* which can make use of higher-order statistical features of the texture will be introduced. It will be shown that the *Bispectral KM Net* using third-order features can improve the classification ability of the KM Net by extracting the phase relation among the spatial frequency components.

In Chapter 6 “Defect Classification in Visual Inspection of Semiconductors”, the HCN

introduced in Chapter 4 trained by BP with MS will be applied to a defect inspection system for semiconductors. First, the details of the effective features will be defined, and it will be deducted from the distribution of the clusters in the feature space, that the HCN is suitable as the classifier. By using the BP training with MS, it will be shown that the training in the multidimensional feature space can be conducted efficiently. On classification of the actual defect images, it was found that a classification rate comparative to those of the human experts could be achieved, which makes the system usable in the fab line.

In Chapter 7 “Conclusion”, the results of this work will be summarized.

Chapter 2
Model switching

In this chapter, the general idea of an operation named *Model Switching* (MS) for enabling the simultaneous determination of the occasion, the model and the initial map will be introduced. After giving the definition of MS, the actual method of MS consulting an index named the *Model Switching Index* (MS Index) will be introduced. This MS Index is a function of the training error, the time derivative of the training error and a measure named *fitness index* which reflects the suitability of the new initial map candidate at each moment of training. It will be discussed that there are various types of possible fitness indices, and MS of different nature can be made possible in accordance with the selection of the fitness indices.

2.1 The definition of model switching

Definition (Model Switching)

On altering the neural network model, methods which determine the *moment* or the *occasion* of model alteration, by taking into account the two factors in the following :

1. The nature and fitness of the new model and the initial map by the new model.
2. The status of the immediate model and map.

will be referred to as *Model Switching*.

In Table 2.1, the feature and the novelty of MS among the training methods that involve MA is shown. The most significant difference in MS-based methods is that the fitnesses of the switchable candidates of the new model and the initial map are also taken into account on determining the instance of MA. In the conventional methods, only the status and the progress of the immediate network was evaluated for determining when to alter the model.

Various measures are conceivable as fitnesses of the new model and map. Among them, a method which rates the destination of switching by the degree of map inheritance will be used in Chapter 3. Another criterion which incorporates the degree of map inheritance and the suitability of the class discriminating border types will be used in the BP training with MS of Radial Basis Function (RBF) and hybrid type networks in Chapter 4.

Table 2.1. The occasions of MA in conventional and proposed methods, with the possible fitness measures for the initial map in the new model.

Fitness of the switching target / **Occasion** to swtich determined by ...	Random	Map preservation	Map preservation + types of class borders	…
training progress of the **immediate model**	GA search, try and err	Sens. analysis + pruning, OBD, OBS		…
training progress of the **immediate model** + fitness of the switchable model-map **candidate**		Chapter 3 Chapter 5	Chapter 4 Chapter 6	…

Model Alteration Model Switching

2.2 Model switching using model switching index

In this section, MS driven by a quantity named the Model Switching Index (MS Index) will be introduced. The MS Index consists of factors named Stress Index and Fitness Index, each reflecting the progress of training and the suitability of the candidate, respectively.

2.2.1 Stress index and the mode of switching

The Stress Index denoted by I_S, is a measure originating from an attempt to empirically model the approach of humans when trying to accomplish a given task. Roughly speaking, the Stress Index reflects the “difficulty” of the training progress, advising the model to be expanded when the value is positive, and to be reduced when it is negative.

Let us imagine a case when a human is assigned to accomplish a certain task. On initial trial , one will try to solve it using a method at hand. Since the task is yet to be accomplished, the effort will be mainly weighted to the accomplishment of the task itself, regardless of the cost. During this period, the performer will go through a repetition of selection or modification of the method and their application. The method tends to start from a simple and general one, gradually growing complicated to adapt to the necessities of the task.

In some tasks, it is sufficient to a merely accomplish it, requiring no further effort. However, the method tends to go through further modifications preferring fairness, simplicity, speed and lower cost in many cases. It can be seen that there is an implicit tendency to push this process forward, which may be modelled as an optimization of a certain composite cost C as shown in Fig. 2.1.

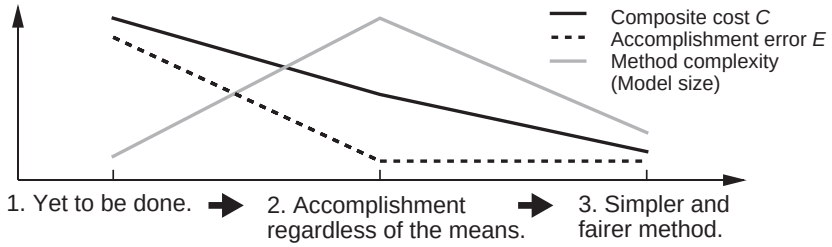


Fig. 2.1. A three-stage reduction of the composite cost found in human problem solving.

If the task is to be replaced to the learning process in neural networks, there are reports in the literature which tries to model the minimization of the composite cost *C* by adding penalty terms to the training error for neural network model simplification by weight decay [18]. However, since the three-stage search route roughly expressed in Fig. 2.1 should include expedient detours to escape from local minima, it is not easy to encode the whole process as a mere minimization of a scalar cost in a gradient descent way. Therefore, the training error minimization and the MS should be used as two separate operations acting on the network in parallel.

In problem solving by humans, two different *modes* exist when seeking alternative methods. Alternative methods are sought in the following occasions. (1) The achievement is yet to be improved, but not much gain is observed for a certain period of time (*positive mode*). (2) The task is accomplished satisfactorily, and the successful method seems to be simplifiable (*negative mode*). The first situation matches to the network during its training process being caught in a local minima, or when the model does not have sufficient degree of freedom. In this case the network should be switched to a larger model. The second situation corresponds to a trained network with reducible factors in the model. In case of humans, this situation is even predicted before actual accomplishment; one can foresee that the task will surely be done at any time, when the requirements are met quickly.

Table 2.2. MS operations being a joint function of the training error E and its time derivative. Values E_0 and I_0 denote the error to attain by training, and a small positive threshold for the evaluation of the error convergence speed. Case (C) does not take place when E_0 is sufficiently small.

E \ $\partial E / \partial t$	(–) ◄– $-I_0$ –► 0	(+)
Not satisfactory (E_0 ↑)	(A) Switch to a smaller model	(B) Switch to a larger model
Satisfactory (0 ↓)	(C) *	(D) Switch to a smaller model

The two modes pointed out above can be detected as a joint condition of the values of the training error and its partial derivative with respect to time, as shown in Table 2.2. The desired switching operation can be controlled using a measure which we name the *Stress Index* I_S, basically defined as a product of the error factor and its time derivative factor as,

$$I_S(E, \frac{\partial E}{\partial t}) = \beta(E - E_0)\cdot\gamma(\frac{\partial E}{\partial t} + I_0)\,, \tag{2.1}$$

using the following notations :

E : Training error.
E_0 : Target error value.
t : Time.
I_0 : Small positive constant.
β , γ : Monotonically increasing real functions satisfying $\beta(0) = \gamma(0) = 0$.

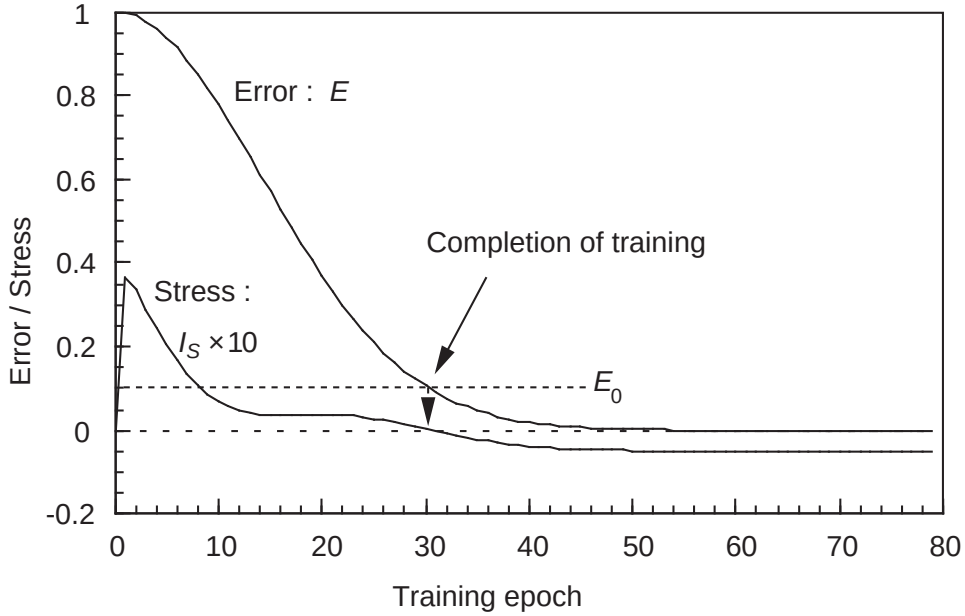


Fig. 2.2. The change of stress index I_S defined in Eq. (2.1) plotted for a typical convergence of error E of a Gaussian function. Functions $\beta(x) = x$ and $\gamma(x) = x$ were used. Parameters were set as, $E_0 = 0.1$ and $I_0 = 0.05$.

2.2.2 Switching operation and switching candidate

As seen from the way of MA in human problem solving in Table 2.2, the direction of MA will be different according to the modes, or the sign of I_S. When I_S is calculated, a set of candidates in possible models and initial maps denoted by $C_{MS} = \{f'_{Ni}\}$ will be determined. This set may be expressed as C^+_{MS} or C^-_{MS} to explicitly signify the sign of I_S.

Each member f'_{Ni} in C_{MS} will typically have a one-to-one relation with the *switching operation* S_i, applied to the immediate network. If the function of the immediate network is f_N, switching operation from f_N to f'_{Ni} will be written as,

$$f'_{Ni} = S_i(f_N) . \tag{2.2}$$

Further, the set of possible switching operations will be expressed as, $O_{MS} = \{S_i\}_{i=1}^{|C_{MS}|}$.

2.2.3 Fitness index

For all elements in $\{f'_{Ni}\}_{i=1}^{|C_{MS}|}$, Fitness Index I_F will be evaluated as

$$I_F = \alpha(f_N, f'_{Ni}) . \tag{2.3}$$

Function $\alpha(f_N, f'_{Ni})$ $(0 \le \alpha \le 1)$ is to take a positive value growing larger as the candidate f'_{Ni} becomes suitable for switching. Some examples of the definitions of the Fitness Index will be suggested in Sec. 2.4.

2.2.4 Model switching index

By using the Stress Index and the Fitness Index, the Model Switching Index (MS Index) will be defined as,

$$I_{MS}(f_N, f'_{Ni}, t) = I_F I_S = \alpha(f_N, f'_{Ni}) \cdot \beta(E(f_N, t) - E_0) \cdot \gamma\left(\frac{\partial E(f_N, t)}{\partial t} + I_0\right), \quad (2.4)$$

where $E(f_N, t)$ explicitly signifies the training error of the network f_N at time t. MS Index defined in Eq. (2.4) is the Stress Index modified by the Fitness of the candidate map to switch to. The MS Index will be monitored throughout the BP training process. Upon monitoring the MS Index, MS will be evoked according to the following rule.

$$\begin{aligned}
&\text{if } ((I_{MS} \geq 0) \text{ and } (\max\{I_{MS}(f_N, f'_{Ni}, t)\} > I^{+}_{MS})) \text{ then} \\
&\qquad \text{switch } f_N \rightarrow f'_{Nk} \text{ where } k = \underset{i}{\operatorname{argmax}}\{I_{MS}(f_N, f'_{Ni}, t)\} \\
&\text{else if } ((I_{MS} < 0) \text{ and } (\min\{I_{MS}(f_N, f'_{Ni}, t)\} < I^{-}_{MS})) \text{ then} \\
&\qquad \text{switch } f_N \rightarrow f'_{Nk} \text{ where } k = \underset{i}{\operatorname{argmin}}\{I_{MS}(f_N, f'_{Ni}, t)\} \\
&\text{else do not switch.}
\end{aligned} \quad (2.5)$$

The threshold values I^{+}_{MS} and I^{-}_{MS} are to be predetermined.

2.2.5 BP training involving MS by MS Index

The flow of BP training incorporating Model Switching is shown in Fig. 2.3. Evaluation of the MS Index can be done at every training epoch, or once in a certain interval. The whole training process is to be terminated when no switching of model occurred and the training error has converged to a sufficiently small value E_0.

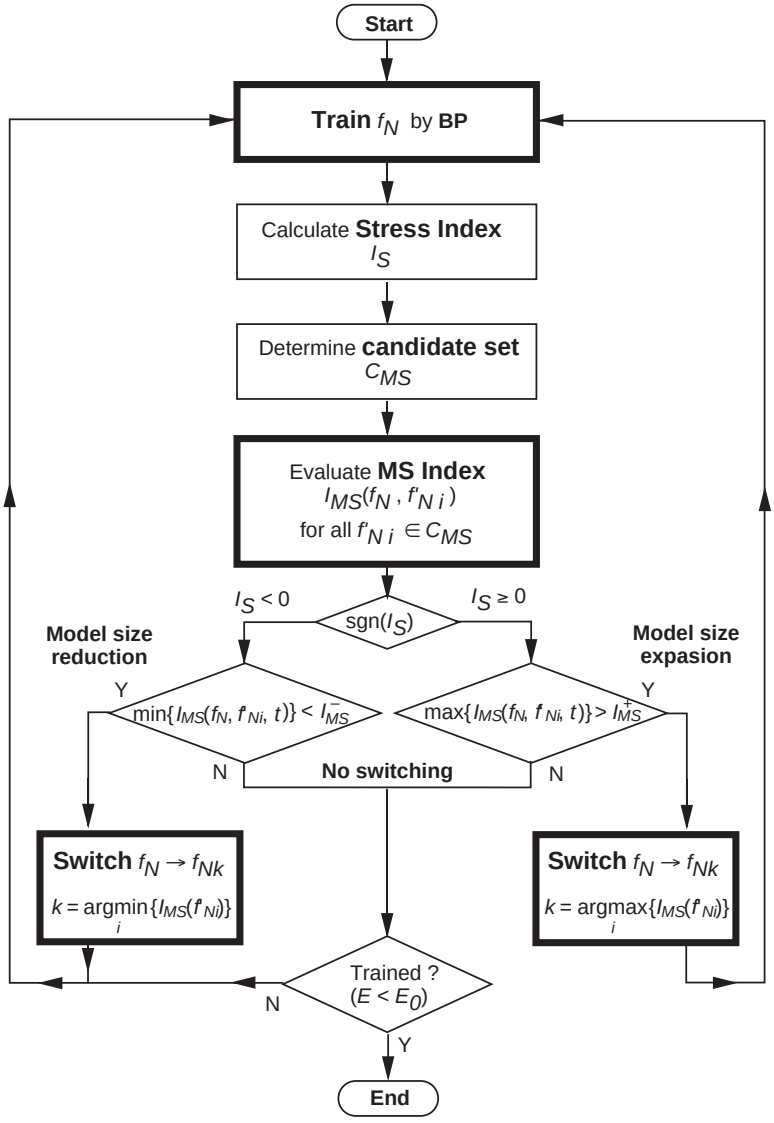


Fig. 2.3. Flow of BP training with MS, controlled by MS Index.

2.3 Stress mode and the switching operation – scheduling of MS–

Various types of Model Switching *scheduling* of the whole training process will become possible by employing different switching operators *S* assigned to the positive and negative modes of Stress Index.

2.3.1 Unlimited model scheduling

This is the most basic scheduling of BP training with MS. When the mode is positive with positive Stress Index, the model size will be expanded. Otherwise the model size will be reduced. The assignment of the switching operators in each mode are as shown in Table 2.3.

Table 2.3. Assignment of MS operators in the unlimited model scheduling.

Mode : sgn(I_S)	Operation : S	Model size : m
Positive	Switch to f'^{+}_{N} in a larger model	$m(f'^{+}_{N}) > m(f_N)$
Negative	Switch to f'^{-}_{N} in a smaller model	$m(f'^{-}_{N}) < m(f_N)$

The typical change of model size in the whole training process illustrated in relation with the training error E will be as shown in Fig. 2.4.

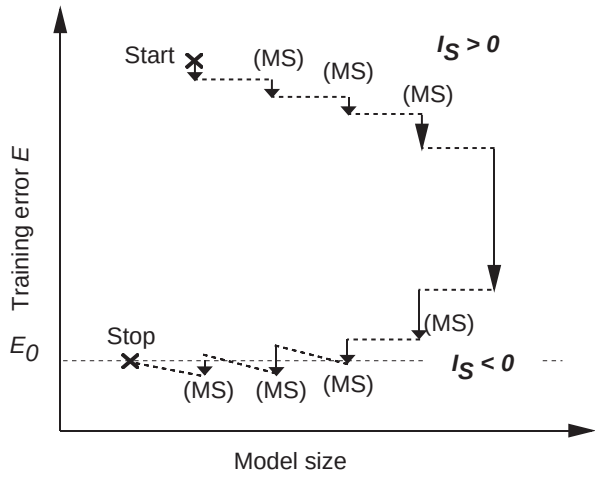


Fig. 2.4. The change of the model in the BP training with MS in unlimited model scheduling.

2.3.2 Limited model scheduling

When applying the training with MS to real world problems, preparing a system which allows various models can be costly. Also, there are cases that the preferred range of model is predetermined due to external assessments for generalization ability etc. In such cases, efficient training method within the limited model must be sought.

As discussed at the introduction of the Stress Index, positive Stress mode switching will be executed for accelerating the training. In limited model scheduling, the switching operation will be a composite one, to switch to a larger model and again to switch to the former model to a different map. When the training was caught in a local minima of the error potential function, the new map should enable to escape from it, having the highest Fitness among the candidates, such as the degree of map preservation etc. In case of negative Stress mode switching, the composite switching operation will be to switch to a smaller model and again to switch to a different map within the former model. This operation can be effective when there are redundancies in the extracted features during fast error

convergence. By switching to a small model to remove the redundancy, and again switching to the original model by adding degrees of freedom may contribute to extraction of novel features for robust pattern recognition. This operation may also contribute to further acceleration of the training, not by reducing the model as discussed in Sec. 1.4 (see Fig. 1.5), but by allowing to capture new features by the added factor of the model. The assignment of the switching operators in each mode are as shown in Table 2.4.

Table 2.4. Assignment of MS operators in the limited model scheduling.

Mode : sgn(I_S)	Operation : S	Model size : m
Positive	Switch to f'^{+}_{N} in the same model via a larger model	$m(f'^{+}_{N}) = m(f_N)$
Negative	Switch to f'^{-}_{N} in a the same model via a smaller model	$m(f'^{-}_{N}) = m(f_N)$

The change of the map by MS in the limited model scheduling is illustrated in Fig. 2.4.

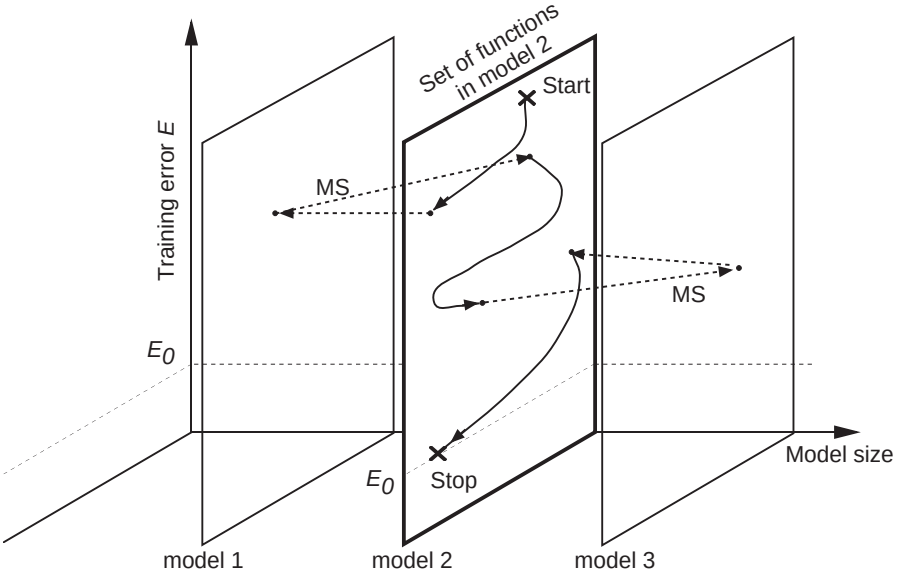


Fig. 2.5. The scheme of the change of the model and the map in the BP training with MS employing the limited model scheduling.

2.4 Selection of initial maps by Fitness Indices

In the framework of MS controlled by MS Index, definition of the Fitness Index will determine the attributes of the maps that will be selected as starting points in the new model. The most commonly accepted nature that are considered necessary in initial maps are the inheritance of the information acquired by training up to the point of MA. By setting a framework which scores the attributes of the initial maps, various natures including map inheritance will become ratable. In the following, two among various definitions of Fitness Indices will be addressed.

2.4.1 Fitness by the degree of map inheritance

This is a common way of selecting the initial map after MA. Many network pruning methods [30][31][35][36] use a method generally known as *sensitivity analysis* to evaluate the factors to be removed in the pruning operation.

In this work, the degree of map inheritance is measured by another means named *map distance* $D(f_N, f'_N)$, which is defined as the norm between the immediate map f_N and the map after MA f'_N. See Chapter 3 for details.

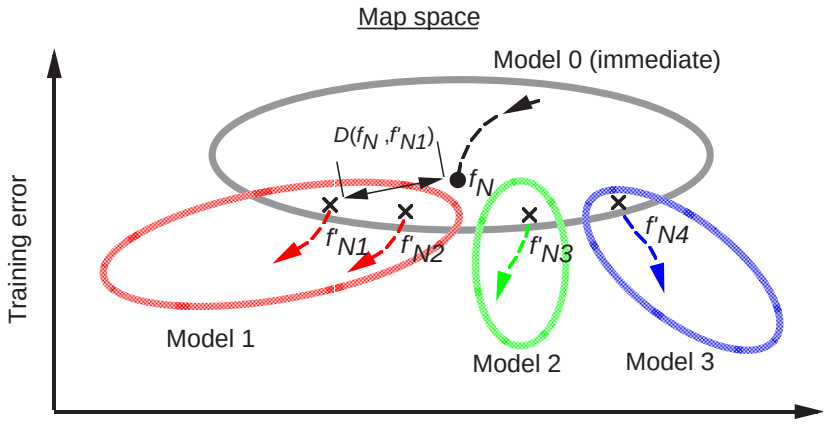


Fig. 2.6. Evaluation of the switchable candidates $f'_{N1} \sim f'_{N4}$ by the map distance criterion. The candidate which gives the smallest map distance between the current map f_N will have the highest fitness.

2.4.2 Fitness by the degree of map inheritance and the nature of class borders

Another conceivable factor which does not reflect its nature to the training error is the type of classification border that the new map will employ. This type of alternative will be available for example, when multiple types of classification borders are choosable, in a network such that more than two types of neurons are co-residing in the hidden layer. In Fig. 2.7, the Radial Basis Function (RBF) type unit which is characterizing the linear class border is to be replaced by units of other type. Among the two candidates, f'_{N2} is more fit to the nature of the class border made by the

training sets.

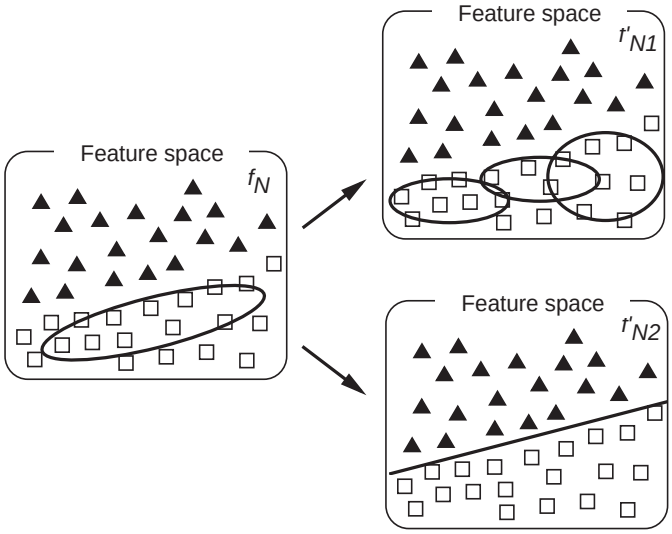


Fig. 2.7. Two switching alternatives in a network that can use locally tuned RBF type units and linear type units.

2.5. Scheduling and fitness used in this thesis

So far, two types of MS scheduling and three types of Fitness Indices have been introduced. In this thesis, combinations of the scheduling and Fitnesses will be evaluated in the chapters as shown in Table. 2.5.

Table 2.5. Combinations of the scheduling and Fitness evaluations.

Fitness / Scheduling	Map perservation	Map perservation + border type
Unlimited model	Chapter 3	Chapter 4
Limited model	Chapter 5	Chapter 6

2.6 Summary

In this chapter, the general idea of an operation named *Model Switching* (MS) for enabling the simultaneous determination of the occasion, the model and the initial map was introduced. After giving the definition of MS, a method of MS consulting an index named the *Model Switching Index* (MS Index) was introduced. The MS Index is defined as a function of the Stress Index reflecting the training progress of the immediate model, and the Fitness Index reflecting the suitability of the new initial map candidate at each moment of training. Then the variations in the scheduling of MS, and the types of Fitness Indices were introduced.

Chapter 3
Model switching in BP training of Multilayer Perceptrons

3.1 Introduction

As discussed in the former chapter, training a neural network is a process of obtaining an approximation map of a desired input-output relation $\boldsymbol{y} = f(\boldsymbol{x})$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and the output vectors, respectively. In case of layered neural network training, the task is to find the set of weights $\boldsymbol{w}$ in an approximating function $\boldsymbol{o} = f_N(\boldsymbol{x}; \boldsymbol{w})$, optimizing a certain criterion. On training the network, a set of ideal input-output pairs $\{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$ are used, and the training criterion is typically a function of this set. The input vector, commonly called the feature vector, is a preprocessed form of a raw observation signal $\boldsymbol{z}$. Thus we can write $\boldsymbol{y} = f(\boldsymbol{x}(\boldsymbol{z})) = g(\boldsymbol{z})$. In some cases, the process of feature selection can also be incorporated to this intra-model optimization task. Neural network models and pattern recognition systems of the kind that optimize both the feature extraction and the classification phases for the sake of better classification abilities have been reported [69][71][72]. The Kernel Modifying Neural Network which will be introduced in Chapter 5 is also a network of this type. In these systems, the selection of the feature set will also be optimized. We can generalize these operations to find the best set of parameters $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{x})$ in a function $f_N(\boldsymbol{x}(\boldsymbol{z}); \boldsymbol{w}) = g_N(\boldsymbol{z}; \boldsymbol{\theta})$.

In both cases of approximation functions f_N and g_N, after determining the function model, training is a task to find an optimal set of parameters $\boldsymbol{w}$ or $\boldsymbol{\theta}$. In the course of optimization, the function model is not altered usually. However, the search for the best approximating function should not be confined within a single function model. Joint search of the function model and the parameter sets will be possible by using model switching and the BP training in parallel, as addressed in the former chapters.

This chapter will be devoted to implementing the Model Switching (MS) controlled by MS Index, in which the Fitness Index is defined as the degree of map inheritance. By using the fitness evaluation of this type, efficient training process involving MS will become possible. This is because the MS method by consulting the MS Index will allow to select the necessary instance of switching, as opposed to the conventional methods where there were no such freedom of choice.

In Sec. 3.2, a method of measuring the degree of map inheritance, named *map distance* will be defined. Further, the Fitness Index using the *map distance* measure will be defined for use in the network training with MS. Map distance evaluates the norm between the immediate function and the candidate after switching. By selecting the candidate which gives the minimum map distance, inheritance of the acquired map will be guaranteed.

In Sec. 3.3, newly proposed methods for MA, named *unit fusion*, *splitting* and *installation* will be introduced. Unit fusion tries to detect the redundancies in the extracted feature among the hidden layer units, by evaluating the similarity and the variance of the unit responses to the training inputs.

In Sec. 3.4, an approximate method for estimating the map distance caused by the unit fusion will be introduced using the theorems giving the upper bounds of the map distances. By using this theorem, the task of map distance calculation in case of unit fusion will be simplified.

In Sec. 3.5, the proposed methods for MA will be applied to network pruning, which many other methods have been discussed in the literature. The unit fusion method will be compared with several well known pruning methods. In the experiments, the superiority of the unit fusion method in map preservation will be shown.

In Sec. 3.6, MS according to MS Index preferring map preserving candidates will be applied to the BP training. There, it will be shown that the whole training process of obtaining a small trained network, will be much efficient when MS by MS Index is used, in comparison with the conventional MA strategies.

In Sec. 3.7, the relation between the generalization ability of the network and the selection of the initial map in the new model using the proposed MA methods and map distance, will be discussed. There, it will be shown that altering the model by unit fusion has a positive effect to improving the generalization ability of the network.

In Sec. 3.8, a variant of the training method governed by the MS Index, which allows a smooth change in the model by controlling the linear independence of the responses of the hidden layer units will be introduced.

3.2 Map distance and fitness evaluation for map inheritance

3.2.1 Map distance

When inheritance of the map in alteration of the model (or the map) is concerned, and when there exist multiple switching candidates, a common measure to define the degree of map inheritance will be in need. In this section, a criterion for the decision making named *map distance* will be introduced, which is defined as a sampled norm between two maps.

An epoch in a training process can be considered to be a switching from one map to the other. It is guaranteed by the continuity of the model that the small change in the parameter vector $\boldsymbol{\theta}$ will only cause a small change in the map $\boldsymbol{g}$. Any transfer from one map to the other can be appropriate as long as an effective metric between the two models are not too large. The candidates of transfer should include a map implemented with a different function model, as long as the continuity and the necessary distance criterion is satisfied.

Since the suitability of a map for a problem is evaluated only with the training sets during the

training process, the metric will be a function of the training sets as well. The following metric will be defined as the measure of distance between two models $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$.

Definition

The *map distance* between two mapping vector functions $\boldsymbol{g}_1(\boldsymbol{x}) = (g_{11}(\boldsymbol{x}) \ldots g_{1O}(\boldsymbol{x}))$ and $\boldsymbol{g}_2(\boldsymbol{x}) = (g_{21}(\boldsymbol{x}) \ldots g_{2O}(\boldsymbol{x}))$ trained with the training vector set $\{(\boldsymbol{x}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$ is defined as,

$$D(\boldsymbol{g}_1, \boldsymbol{g}_2) = \frac{1}{MO} \sum_{m=1}^{M} \sum_{k=1}^{O} \{g_{1k}(\boldsymbol{x}_m) - g_{2k}(\boldsymbol{x}_m)\}^2, \tag{3.1}$$

where O and M are the dimension of the vector functions and the number of training pairs, respectively.

When selecting the map to switch to among the candidates, the candidate map giving the minimum map distance should be selected. Generally, it is not easy to find a model and a function which gives a small map distance from a network in an arbitrary state. However, as it will be shown later in this chapter, there are several special cases when switching with minimal map distance is possible.

3.2.2 Map inheritance as fitness index

On using the map distance defined in Eq. (3.1) in the Fitness Index factor of the MS Index, the fitness should be larger when the map distance between the immediate map f_N and the target map f'_N is smaller. Therefore, a definition of

$$I_F(f_N, f'_N) = \begin{cases} \{D_{max} - D(f_N, f'_N)\} / D_{max}, & \text{if } \{D(f_N, f'_N) < D_{max}\} \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

will be used as the Fitness Index. In Eq. (3.2), D_{max} denotes the maximum value which the map distance can take. When the output elements o_{km} and the training output elements y_{km} in Eq. (1.6) are bounded (such as in the case of sigmoid units), D_{max} can be set to unity. If the output elements are not bounded (as in the case of linear units), there is no upper bound D_{max} in the strict sense. However, in actual usage of the networks, selecting a sufficiently large D_{max} will be sufficient.

3.3 Model alteration by unit fusion, splitting and installation

3.3.1 Neural network pruning algorithms

Neural network pruning is popularly used as a means to alter the model to a smaller trained network. In many pruning algorithms, however, the procedures are not very careful in letting the new model to inherit the mapping obtained by the training, or are not successful in doing so [38]. Thus, with some methods that detect and simply remove units in the hidden layer, MA can result in a severe loss of the trained map, requiring a long additional training. Here, several known algorithms for neural network

pruning that are applicable during training will be surveyed, and the necessity of an algorithm for feature dimension reduction which tries to preserve the information accumulated into the network, will be emphasized.

For adjusting the hidden layer size for better balance of mapping ability and the simplicity of the model, several works have been reported [30][31][32]. Most of the approaches are to train a network starting with a redundant set of hidden layer units, and to remove (prune) the unimportant ones: Mozer and Smolensky [30], and Karnin [31] have tried to detect the removable connections by calculating the sensitivity of the error function to the pruning of each connection. On pruning, the link with a minimum sensitivity was chosen. We have tried their method and it produced good subset networks. The *damage* due to the pruning was also small. However, the remaining network could not always easily compensate the removed unit's function in further learning. Sietsma and Dow [32] have formulated a pruning method by analyzing the responses of hidden layer units for all learning sets. They state :

1) If the outputs of two units are synchronous throughout the learning patterns or are alternating, one of the two can be omitted.
2) If the output of a unit is always the same for all the learning sets, the unit can be pruned.
3) If the pruning of a unit does not affect the logical separability of the patterns formed by the hidden layer units, the unit can be omitted.

It is true that no lack of information occur by any of these operations, but again, the effect of pruning to the error is not taken into account. Generally, the increase of error after these operations were not small; the network needed more re-learning iterations to regain the complete function, when compared with the Mozer and Smolensky's method. In these works, not much discussion has been done on the cost of further learning, to eliminate the increase of error caused by the pruning. When implementing a neural network through pruning procedures, a fully functioning subset network is in need, rather than a impaired subset (even though the damage may be trivial). Thus, reducing both the damage caused by each pruning operation and the total cost of computation to obtain a compact network should be the major aims. It will be shown that the damage could be substantially reduced when pruning is done by introducing the *fusing* method.

The technique of MA introduced in this section resembles the Sietsma and Dow's first rule, in that synchronously or alternately firing pair of hidden layer units are the candidates to be pruned. However, instead of merely removing one of the two, the connection weights to the remaining unit is altered so that the map distance of pruning remains minimal, allowing the cost of further learning to be reduced. By applying this operation to two units responding similarly or alternately, it is possible to replace the two with one unit without losing much of either unit's features. Thus we call this operation *fusion*.

3.3.2 The basic idea

In contrast with the weight removal strategies, a more drastic reduction in the number of parameters is

possible by changing the number of units. It may seem that such a drastic change in the model may cause a larger gap in the functional map, however, with additional weight compensations in the remaining network, the gap can be suppressed.

A layered network manifests itself as serially connected vector functions for feature extraction as illustrated in Fig. 3.1.

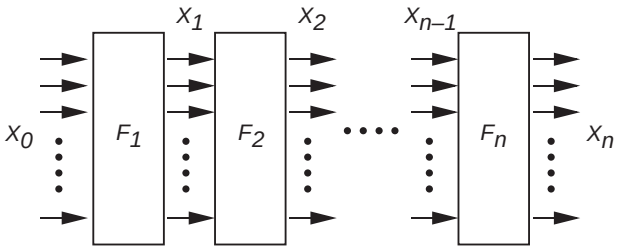


Fig. 3.1. The layered network as a cascade of vector-to-vector functions F_1, F_2, ..., F_n. Matrices X_0, X_1, ..., X_n denote the layer response trace for all the training sets, in the style of the notations defined above.

The reduction of the dimensionality in each layer will be possible by evaluating the linear independence of the rows of the layer response trace matrices X_0, X_1, ..., X_n, each of which column consists of the layer response against each training pattern. When a channel is dependent to the other, the channel pair can be replaced by a single channel, thereby reducing the model size without causing any change in the map. Since this operation ensures a smooth switching from a larger model to a smaller one, this operation will be named *unit fusion* and will be used extensively in this work. When the operation is to be exercised in a neural network, manipulations to the weights connected to the remaining unit will be necessary. The details of unit fusion in a layered neural network will be explained in the next subsection.

The growing procedure for a unit will be a relatively trivial one. It is possible by either installing a unit with zero effect to the preceding layer, or to do the inverse operation of unit fusion, namely to install a unit whose response is dependent to another unit. Both operations will be used in this work, and will be called *unit installation* and *unit splitting*, respectively.

3.3.3 Unit fusion by similarity evaluation

Here, the proposed fusion method will be introduced. The discussion will be focused to the hidden layer units in an MLP, consisting of units whose unit potential employ linear discriminant functions.

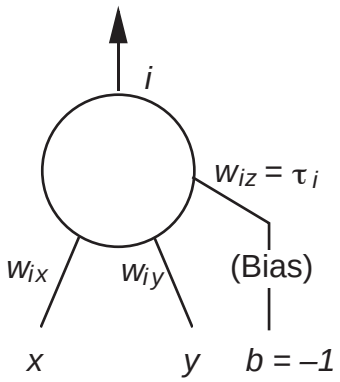


Fig. 3.2. A neuron with two inputs and a bias input.

First, a neuron which has two inputs and one bias input as shown in Fig. 3.2, will be considered. The activation function of a unit may be a step function or a sigmoid function. If we name the two inputs as x and y, and the bias input as b (which is always –1), the link weights as w_{ix}, w_{iy} and w_{iz} (=threshold τ_i), respectively, the role of this neuron i is to dichotomize the three dimensional space spanned by orthogonal bases $(x,y,b) = (0,0,1),(0,1,0)$ and $(1,0,0)$. The separating hyperplane H_i can be written as,

$$H_i \ : \ w_{ix}x + w_{iy}y + w_{iz}b = 0 \ . \tag{3.3}$$

This hyperplane always includes the origin of the coordinate system, so it can be characterized with a vector

$$\boldsymbol{w}_i = (w_{ix}, \ w_{iy}, \ w_{iz}) \, , \tag{3.4}$$

which is perpendicular to H_i .

A. Synchronous pairs

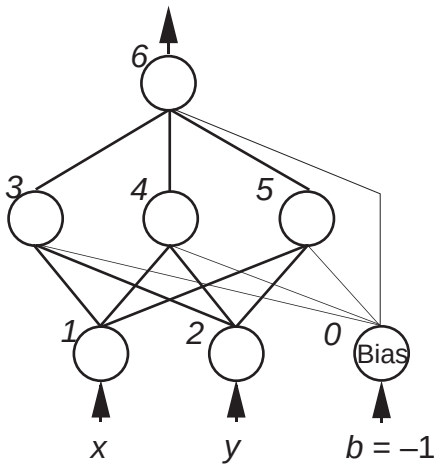


Fig. 3.3. A small three-layered network.

Let us assume that a small network illustrated in Fig. 3.3 was trained with M patterns, and that units 3 and 4 responded similarly to all the training patterns. The similarity of units i and j defined as r_{ij} is evaluated by the linear correlation coefficient formula as,

$$r_{ij} = \sum_{m=1}^{M} (h_{im} - \overline{h_i})(h_{jm} - \overline{h_j}) \Big/ \left[\sum_{m=1}^{M} (h_{im} - \overline{h_i})^2 \sum_{m=1}^{M} (h_{jm} - \overline{h_j})^2 \right]^{1/2} . \tag{3.5}$$

Here, h and $\overline{h}$ denote the output and the mean output of a unit, respectively. Similarity r_{ij} falls between –1 and 1. If h_{im} and h_{jm} are similar over all the learning sets (synchronous), r_{ij} approaches unity. If they are nearly opposite (alternate), $r_{ij} \cong -1$. Hence, in our case assumed in the above, $r_{34} \cong 1$.

On analyzing the hidden layer activities, the similarity of the units may be evaluated by the distance between the input weight vector directions, using a subspace distance metric [14]. However, since the distribution of the class vectors can be very complicated and uneven, it is safer to use the training set oriented measure as defined in Eq. (3.5).

Let us try to replace these two units by a unit for dimensionality reduction. The new neuron replacing the two will be called neuron a. We will first look into the link weights from the input layer to unit a, then to the weights from unit a to the output layer.

As stated at the beginning of this section, units 3 and 4 in the hidden layer separate the xyb feature space with hyperplanes H_3 and H_4. These hyperplanes are characterized by vectors $\boldsymbol{w}_3 = (w_{31},\ w_{32},\ w_{30})$ and $\boldsymbol{w}_4 = (w_{41},\ w_{42},\ w_{40})$. Since both units respond similarly to many input patterns, we can suppose that hyperplanes H_3 and H_4 are more or less the same thing. In other words, vectors $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$ intersect at the origin O with an acute angle. If a new hyperplane for unit a is determined so that vector $\boldsymbol{w}_a$ (perpendicular to hyperplane H_a) would have the bisecting direction of the two vectors $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$, this H_a should be a good replacement of the original two hyperplanes as seen in Fig. 3.4. The length of $\boldsymbol{w}_a$, as long as they are not too small, will not remarkably change the output h_a even if the activation function of a neuron is a sigmoid function. However, it will certainly affect the readiness of further learning. It is well known that the link weights that have grown too large through many learning iterations, are hard to be changed by further backpropagation learning. Considering this fact, the length of $\boldsymbol{w}_a$ was initially set to a very small value, but this caused more unsteady changes in the direction of $\boldsymbol{w}_a$ than desirable modifications in the post-fusion training. Hence, the length of $\boldsymbol{w}_a$ was set to be the average length of the two vectors $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$. This resulted in a much steadier learning after the fusion. Now, vector $\boldsymbol{w}_a$ is formally defined as,

$$\boldsymbol{w}_a = \frac{|\boldsymbol{w}_3| + |\boldsymbol{w}_4|}{2} \frac{\boldsymbol{e}'_a}{|\boldsymbol{e}'_a|} \tag{3.6}$$

where

$$\boldsymbol{e}'_a = \frac{\boldsymbol{w}_3}{|\boldsymbol{w}_3|} + \frac{\boldsymbol{w}_4}{|\boldsymbol{w}_4|} \ . \tag{3.7}$$

If vector $\boldsymbol{w}_a$ is set according to Eq. (3.6), the output of unit a will replicate the outputs of the former

units 3 and 4 over all the training patterns. Therefore we can write

$$h_{3m} \cong h_{4m} \cong h_{am}\,, \qquad (m = 1,2,\ \ldots,\ M)\,. \tag{3.8}$$

As for the link weight from unit a to the output unit 6, it should be set so as to minimize the effect of the fusion to the weighted sum of inputs for unit 6 (u_6). If the approximation in Eq. (3.8) holds, we can write

$$\begin{aligned} u_6 &= w_{63}h_3 + w_{64}h_4 + w_{65}h_5 - w_{60} \\ &\quad \text{(before fusion)} \\ &\cong (w_{63} + w_{64})h_a + w_{65}h_5 - w_{60} = w_{6a}h_a + w_{65}h_5 - w_{60}\,, \\ &\quad \text{(after fusion)} \end{aligned} \tag{3.9}$$

where the new link weight w_{6a} is defined as

$$w_{6a} = w_{63} + w_{64}\,. \tag{3.10}$$

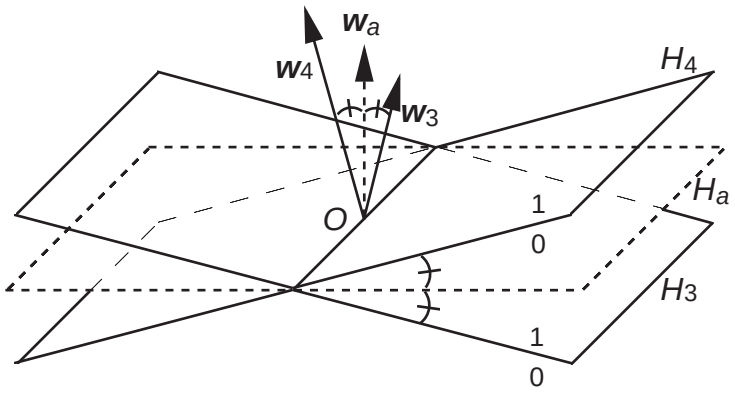


Fig. 3.4. Feature space separating hyperplanes and weight vectors for the synchronous unit pair. The hyperplane and the weight vector for the fused unit is shown in dashed lines.

B. Alternately firing pairs

If the output of units are alternating over all the learning sets, the pair of units can also be replaced by a single unit as we did in the synchronous pair case. Again, we will use the network illustrated in Fig. 3.3 for explanation, but assume that units 3 and 4 were alternately firing pairs. Hyperplanes H_3 and H_4 specified by the link weights from the input to the hidden layer, are again, quite similar. But this time, the direction of the vectors $\boldsymbol{w}_3$ and $\boldsymbol{w}_4$ are almost the opposite to each other as shown in Fig. 3.5. Hence, the fusing maneuver for the synchronous pair can be used with some additional procedures.

The method employed in the synchronous case is used to determine the link weights from the input layer to unit *a*, except that vector $\boldsymbol{w}_a$ will have the direction bisecting the angle between either of the $\boldsymbol{w}$ vectors and a vector with the reversed direction of the other (e.g. $\boldsymbol{w}_3$ and $-\boldsymbol{w}_4$ in Fig. 3.5).

Two alternately firing units 3 and 4 were fused into one and they became a single unit *a*. If the units have (0, 1) output ranges, we can roughly write in symmetry to Eq. (3.8) as,

$$h_{3m} \cong 1 - h_{4m} \cong h_{am}\,, \qquad (m = 1,2,\ \ldots,\ M)\,. \tag{3.11}$$

In Eq. (3.11), h_{ap} is equivalent to the opposite of h_{4m} because $\boldsymbol{w}_4$ was reversed and the fusion took place to make unit a. The weighted sum of inputs u_6 of the output neuron 6 is,

$$\begin{aligned} u_6 &= w_{63}h_3 + w_{64}h_4 + w_{65}h_5 - w_{60} \qquad \text{(before fusion)} \\ &\cong w_{63}h_a + w_{64}(1 - h_a) + w_{65}h_5 - w_{60} \quad \text{(after fusion)} \\ &= (w_{63} - w_{64})h_a + w_{65}h_5 - (w_{60} - w_{64}) \\ &= w_{6a}h_a + w_{65}h_5 - w'_{60}\,. \end{aligned} \tag{3.12}$$

Therefore, the new link weight w_{6a} and the new firing threshold w'_{60} can be written as

$$w_{6a} = w_{63} - w_{64} \tag{3.13}$$

and

$$w'_{60} = w_{60} - w_{64}\,. \tag{3.14}$$

When the units employ the bipolar activation function with (–1, 1) output ranges such as Eq. (1.5), the threshold modification in Eq. (3.14) is unnecessary because

$$s(-u) = -s(u)\,, \tag{3.15}$$

and

$$h_{3m} \cong -h_{4m} \cong h_{am}\,. \quad (m = 1,2,\ \ldots,\ M) \tag{3.16}$$

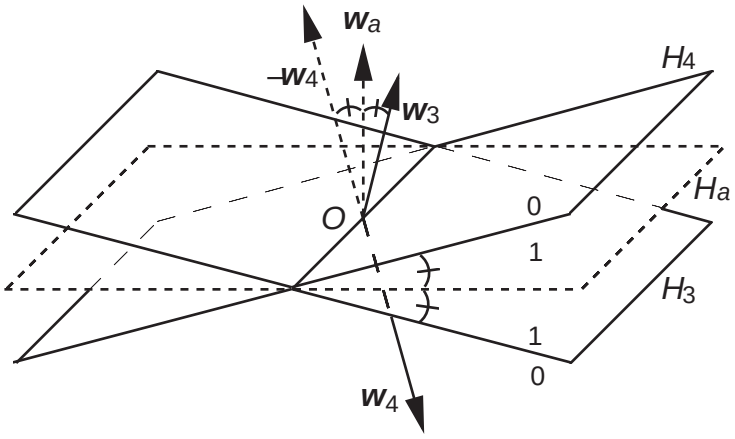


Fig. 3.5. Feature space separating hyperplanes and weight vectors for the alternately firing unit pair and their replacement in dashed lines.

3.3.4 Formal analysis of unit fusion by similarity evaluation

Here, a unified description of the compensations of the weights between the hidden and the output layers will be made. The idea of hidden layer unit fusion will be extended to the reduction of arbitrary feature unit pair, according to a detection criterion and a replacement with weight compensations. In order for the method to be applicable to a wider class of feature channel reduction, the symmetric nature of the hidden layer unit fusion, which was enabled by the compensations of the weights between the input and the hidden layers, will be omitted. Thus, the fusion operation of the channel pair will be defined as an operation to *merge* a channel to another, which is not a symmetric

operation.

Let us assume that channel i and j are to be fused to make a single unit i'. The weighted sum of the inputs from units i , j and the unity bias b to the subsequent layer unit k, can be written as

$$u_k = w_{ki}h_i + w_{kj}h_j + w_{kb} = w_{ki}(e_i + v_i) + w_{kj}(e_j + v_j) + w_{kb} , \tag{3.17}$$

where w, h, e and v are the connection weight, unit response, average unit response and the varying portion of the response, respectively. When the similarity criterion is used, the units i and j are highly correlated. Therefore, we can put

$$v_j \cong \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}v_i , \tag{3.18}$$

with σ being the standard deviation of the unit responses, and sgn(x) being the sign function defined as,

$$\mathrm{sgn}(x) = \begin{cases} 1 & (x \geq 0) \\ -1 & (x < 0) \end{cases} . \tag{3.19}$$

From Eqs. (3.17) and (3.18), we have

$$u_k \cong \{w_{ki} + \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}w_{kj}\}h_i + w_{kb} + w_{kj}\{e_j - \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}e_i\} , \tag{3.20}$$

implying that the connection weights should be changed as,

$$w'_{ki'} = w_{ki} + \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}w_{kj} \tag{3.21}$$

and

$$w'_{kb} = w_{kb} + w_{kj}\{e_j - \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}e_i\} , \tag{3.22}$$

where w' denote the connection weights after the fusion.

In view of Eqs. (3.20)~(3.22), the weight compensations used in the former subsection for hidden layer unit fusion implicitly assumed,

$$\sigma_i = \sigma_j \tag{3.23}$$

and

$$e_i = e_j . \tag{3.24}$$

which do not hold generally, especially when handling unnormalized feature signals instead of neuron responses.

Among the pruning methods in the literature, there exist few attempts that additionally compensates the remaining weights for the sake of map preservation. The strategy introduced above is a generalization of the pruning algorithm which is accompanied by a weight compensation for map preservation, introduced by Sietsma and Dow [33], Oshino, Ojima and Yamamoto [40] and the author [34] independently. The advantage of the method also resides in its simplicity. There are other approaches that are computationally more expensive. Castellano et al. try to recalculate the remaining affected weights by a gradient optimization apart from the BP training process [37]. Hassibi et al. introduced Optimal Brain Surgeon [36] which improves the Optimal Brain Damage pruning [35] by

Le Cun et al. In [36], rigorous calculation of the weight values for eliminating the error increase by removal of a link is introduced. However, the method is only applicable to a trained network whose error potential function in the neighborhood of the converged minimum can be approximated by a quadratic function.

3.3.5 Unit fusion by the variance evaluation

Here, another method of unit fusion to complement the fusion method by similarity detection, will be introduced. The detection method is equivalent to the second rule in Sietsma and Dow's detection rules [32], which was briefly addressed in Sec. 3.2. A channel with constant response, is a simple bias. Such a unit can be unified with the bias input commonly used in layered neural networks. This phenomena can be detected by monitoring the variance of the unit output. If a response variance of a unit falls below a threshold, it can be merged to the bias input, thereby enabling to reduce the unit numbers.

When unit i is detected to have a low variance response to the training sets, the fused unit i is a bias input b and the other unit j is constantly responding. In this case, modifications such as

$$w'_{kb} = w_{kb} + w_{kj} e_j \,, \tag{3.25}$$

is sufficient since $v_j \cong 0$.

Joint use of the similarity and variance criteria has been tried by Sietsma and Dow in [33], and also by the group of Yamamoto et al. in [40][41], where they experimentally showed that, the reduction according to the variance criterion is to be used at the early epochs of training, and the similarity criterion is to be used later on for an efficient reduction of the redundant units in a three-layered network. However, no justification of the selection of this ordering had been done.

In this work, a unified criterion based on the map distance will be used. The criterion can be used for selecting the best model for map preservation among those proposed by multiple model switching methods.

3.3.6 Unit splitting

When the network is in need of more degree of freedom, the model will be altered by adding a unit. The weights of the links connected to the newly added unit will be determined as follows. Here, the network is assumed to be an *L-N-O* network.

1. Find a hidden layer unit having minimum firing similarity (correlation) with any of the other hidden units. The amount

$$R_i = \sum_{n \neq i,\ n=1}^{N} |r_{in}| \tag{3.26}$$

is calculated for all the hidden layer units (n = 1,2, ..., N), and the unit with minimum R_i is selected. Here, the chosen unit and the added unit will be referred to as units i and j, respectively.

2. Set the weights from the input layer to unit j as

$$w_{jl}^{new} = 0.1 w_{il}^{old} + d_{il}, \qquad (l = 1,2, \ ..., \ L) \tag{3.27}$$

and the weights from unit j to the output layer as

$$w_{kj}^{new} = 0.1 w_{ki}^{old}, \qquad (k = 1,2, \ ..., \ O). \tag{3.28}$$

The term d_{il} in Eq. (3.27) denote a small zero mean random number.

3. Modify the weights from unit i to the output layer as

$$w_{ki}^{new} = 0.9 w_{ki}^{old}, \qquad (k = 1,2, \ ..., \ O). \tag{3.29}$$

This is a reversed procedure of fusing a synchronously firing pair, except for the norm of new unit's feature vector $\boldsymbol{w}_j$. As defined in (3.27), H_j is identical with H_i, but the norm of the feature vector $\boldsymbol{w}_j$ is 1/10 of $\boldsymbol{w}_i$ with some added noise. Eqs. (3.28) and (3.29) define an asymmetrical division of weights to the output layer. These measures allow unit i to preserve the obtained feature, and unit j to easily change and grasp another feature.

3.3.7 Unit installation

On expanding the model, another way of adding a hidden layer unit is conceivable. This method will be referred to as *unit installation*.

1. Set the weights from the input layer to the newly installed unit j as

$$w_{jl}^{new} = d'_{il}, \qquad (l = 1,2, \ ..., \ L) \tag{3.30}$$

where d'_{il} denotes a random number.

2. Set the weights from unit j to the output layer as

$$w_{kj}^{new} = 0, \qquad (k = 1,2, \ ..., \ O). \tag{3.31}$$

Since the weights set in Eq. (3.28) are all zero, the unit will initially have no effect to the map. As the BP training proceeds, they will be altered. This method is also easily applicable to expanding the model by adding neurons of other type.

When two model expansion methods are compared, unit installation is a method which throws in a unit tuned to a random feature, whereas unit splitting tries to start the search for new feature from the neighbor of a feature that is already recognized by the unit that will be split.

3.3.8 The virtue of model switching by splitting in the BP training

As an example of the merit of using MS during BP training, let us put that a training process of a three-layered neural network is caught in a local minimum of the error function. The situation can be described by the notations defined in Sec. 1.4.2 as,

$$E = \sum_{m=1}^{M} E_m >> 0, \tag{3.32}$$

$$G_0^T = D_1^T X_0 = 0, \tag{3.33}$$

and

$$G_1^T = D_2^T X_1 = 0. \tag{3.34}$$

When the number of units in the hidden layer is incremented by unit installation, the new connection weight matrices and the output trace matrix both signified by the hat will be,

$$\widehat{W}_0 = [W_0^T \mid \boldsymbol{w}_{0,N+1}^T]^T \in R^{N+1} \times R^{L+1}, \tag{3.35}$$

$$\widehat{W}_1 = [W_1 \mid \boldsymbol{w}_{1,N+1}] \in R^O \times R^{N+2} \tag{3.36}$$

and

$$\widehat{X}_1 = [X_1 \mid \boldsymbol{x}_{1,N+1}] \in R^M \times R^{N+2}, \tag{3.37}$$

where $\boldsymbol{w}_{0,N+1}$,$\boldsymbol{w}_{1,N+1}$ and $\boldsymbol{x}_{1,N+1}$ denote the weight vectors of the connections impinging to and fanning out from the new unit, and the response trace of the new unit, respectively. Since the map is not altered by adding the unit, there is no change in the training error; namely,

$$\widehat{E} = E. \tag{3.38}$$

The necessary condition for the network to be still trapped in the minimum after adding the unit will be,

$$\widehat{G}_0^T = \widehat{D}_1^T X_0 = [D_1^T \mid \boldsymbol{d}_{1,N+1}^T]^T X_0 = 0, \tag{3.39}$$

and

$$\widehat{G}_1^T = D_2^T \widehat{X}_1 = D_2^T [X_1 \mid \boldsymbol{x}_{1,N+1}] = 0, \tag{3.40}$$

for the gradient matrices of the connections. Therefore, the conditions to the added unit will be,

$$\boldsymbol{d}_{1,N+1} X_0 = 0 \tag{3.41}$$

and

$$D_2^T \boldsymbol{x}_{1,N+1} = 0. \tag{3.42}$$

In case of unit installation where the weight vectors $\boldsymbol{w}_{0,N+1}$ and $\boldsymbol{w}_{1,N+1}$ will be set to a random vector and a zero vector, the probability of the added unit to fulfill the conditions in Eq. (3.42) will be small. Thus, the gradient matrix in Eq. (3.40) will be nonzero, and the error will further decrease by training as illustrated in Fig. 3.6. When the training is accomplished ($E = 0$), the decrease will not take place since $D_2 = 0$.

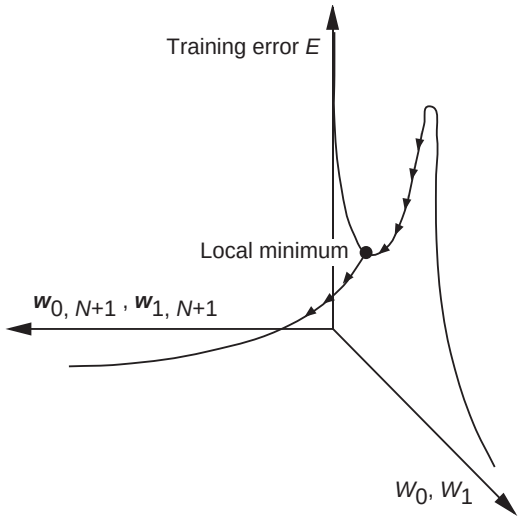


Fig. 3.6. After the training is caught in the local minimum of the error function $E(W_0, W_1)$, an additional unit is added to the hidden layer maintaining the error value. Since the resulting gradient matrix will be nonzero, the training error will further decrease.

When unit *n* in the hidden layer employing a linear transfer function was split instead of installing a unit of zero influence, the response trace of the new unit will be linearly dependent to the existing unit *n*. This situation satisfies Eqs. (3.41) and (3.42), which indicates that the training is still caught in the same minimum. However, when the units employ a nonlinear transfer function, the new gradient matrix will generally be nonzero, which enables to further decrease the error and escape from the local minimum. A detailed discussion on the determination of the connection weights of the split units will be found in the next subsection.

The BP training process jointly using MA can be considered to be a process of searching the model and the map at the same time. The model ought to be the one which can accommodate the training conditions such as in Eq. (1.1), and further, it should be a minimal model that performs the necessary mapping for the ease of use. MA can be effective in aid of the inefficient and time consuming process of feature extraction using the BP algorithm, such as the herd effect introduced in Sec. 1.4.2. Since the detection and switching naturally maintains the independence of the responses of the unit vectors, group of units consisting a herd will be replaced by less number of units. When the limited model scheduling is used, the reduced units can be immediately re-installed to capture a new feature. MS which allows MA at arbitrary occasions during training, opens up whole new branching paths to the stepwise learning progress of BP, which was impossible when the training was restricted to be done in a single functional model.

3.4 Map distance by unit fusion, splitting and installation

In this section, methods for evalutanting the map distances caused unit fusion, splitting and installation will be discussed. Theorems estimating the upper bounds of the map distance caused by unit fusion will be given. These upper bounds enable to approximate the map distances without actually constructing and evaluating the networks after MS.

3.4.1 Upper bounds of the map distance by unit fusion

Selection of the best map to switch to among the numerous candidate is a costly task, because the map must be constructed in a neural network form, and the Fitness Indices must be evaluated for each network. Here, theorems giving the upper bounds of the map distances as a function of similarity r_{ij} and variance σ_j will be given. Instead of generating all the networks and evaluating one by one, the upper bounds given by the theorems can be used as alternative approximations of the map distances for each candidate.

Theorem 1

Let the vector map of a three layered network be denoted by $\boldsymbol{g}_1$, and the map of the same network with one hidden unit j pruned by fusing with unit i, be denoted by $\boldsymbol{g}_2$. By denoting the distance of the two maps by $D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2)$,

$$D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2) \le \frac{2}{O}\left(\sum_{k=1}^{O} w_{kj}^2\right)(S'_{max})^2\sigma_j^2(1-|r_{ij}|) \tag{3.43}$$

holds using the following notations.

O : Number of output layer units.

S'_{max} : Maximum derivative of the unit transfer function.

w_{kj} : Connection weight between an output unit k and a hidden unit j.

When unit j is to be fused with the bias,

$$D_{bj}(\boldsymbol{g}_1, \boldsymbol{g}_2) \le \frac{1}{O}\left(\sum_{k=1}^{O} w_{kj}^2\right)(S'_{max})^2\sigma_j^2, \tag{3.44}$$

holds instead. B

Proof

The map distance of two networks before fusion ($\boldsymbol{g}_1$) and after fusion ($\boldsymbol{g}_2$) is,

$$D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2) = \frac{1}{MO}\sum_{m=1}^{M}\sum_{k=1}^{O}\{g_{1km}-g_{2km}\}^2 = \frac{1}{MO}\sum_{m=1}^{M}\sum_{k=1}^{O}\{s(u_{1km})-s(u_{2km})\}^2, \tag{3.45}$$

where $s(u)$ and u are the (sigmoid) transfer function and the weighted sum of inputs of the output units, respectively. Since a hidden layer unit pair was fused, the difference in the output layer unit potential is small. Therefore, an approximation of

$$s(u_{1km}) - s(u_{2km}) \cong s'(u_{1km})\,(u_{1km} - u_{2km}) \quad , \tag{3.46}$$

can be safely used. Using Eq. (3.46), the map distance will be written as,

$$D_{ij} \cong \frac{1}{MO}\sum_m \sum_k \{s'(u_{1km})\,(u_{1km} - u_{2km})\}^2 = \frac{1}{MO}\sum_m \sum_k s'(u_{1km})^2 \left\{ w_{kj}(v_j^p - \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i} v_i^p) \right\}^2 , \tag{3.47}$$

provided that u_{1kn} and u_{2kn} are close, with *s'* being the derivative of the transfer function *s*. Using the notation in Eqs. (3.17) and (3.18), the last expression is found.

By using $S'_{max} \equiv \max\{s'(u)\}$, we can write

$$D_{ij} \le \frac{1}{MO}(S'_{max})^2 (\sum_k w_{kj}^2) \sum_m \{v_{jm} - \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i} v_{im}\}^2 \tag{3.48}$$

as the upper bound. The last summation can be further simplified by using relations $\{\mathrm{sgn}(r_{ij})\}^2 = 1$ and

$$r_{ij} = \sum_{m=1}^{M} v_{im} v_{jm} \,/\, M\sigma_i\sigma_j \,, \tag{3.49}$$

as

$$\begin{aligned} \sum_m \{v_{jm} - \mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i} v_{im}\}^2 &= \sum_m (v_{jm})^2 + \frac{\sigma_j^2}{\sigma_i^2}\sum_m (v_{im})^2 - 2\mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}\sum_m v_{jm} v_{im} \\ &= M\sigma_j^2 + \frac{\sigma_j^2}{\sigma_i^2} \cdot M\sigma_i^2 - 2\mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i} \cdot M\sigma_i\sigma_j r_{ij} \\ &= 2M\sigma_j^2\{1 - \mathrm{sgn}(r_{ij}) r_{ij}\}. \end{aligned} \tag{3.50}$$

Substituting Eq. (3.50) to Eq. (3.47), we have

$$D_{ij} \le \frac{2}{O}(S'_{max})^2 (\sum_k w_{kj}^2)\sigma_j^2 (1 - |r_{ij}|) \,. \tag{3.51}$$

When unit *j* is to be fused with the bias, it is clear that

$$v_i = v_b = 0. \tag{3.52}$$

By substituting Eq. (3.52) into Eq. (3.48), we have,

$$D_{bj} \le \frac{1}{MO}(S'_{max})^2 (\sum_k w_{kj}^2)\sum_m (v_{jm})^2 = \frac{1}{O}(S'_{max})^2 (\sum_k w_{kj}^2)\sigma_j^2 . \tag{3.53}$$

(End of Proof)

Theorem 1 can also be used to evaluate the distance of the hidden layer responses when the input channels are fused in a three-layered network. However, for evaluating the map distance measured at the output layer for the input channel fusion, the following theorem can be used.

Theorem 2

Let the vector map of a three layered network be denoted by $\boldsymbol{g}_1$, and the map of the network with one input channel *j* pruned by fusing with channel *i*, be denoted by $\boldsymbol{g}_2$. By denoting the distance of the two maps by $D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2)$,

$$D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2) \le \frac{2}{O}\{\sum_{k=1}^{O}(\sum_{n=1}^{N} w_{kn}w_{nj})^2\}(S'_{max})^4\sigma_j^2(1-|r_{ij}|) \tag{3.54}$$

holds with the following notations.

O	:	Number of output layer units.
N	:	Number of hidden layer units.
w_{kn}	:	Connection weight between output unit k and hidden unit n.
w_{nj}	:	Connection weight between hidden unit n and input channel j.

When channel j is to be fused with the bias,

$$D_{bj}(\boldsymbol{g}_1, \boldsymbol{g}_2) \le \frac{1}{O}\{\sum_{k=1}^{O}(\sum_{n=1}^{N} w_{kn}w_{nj})^2\}(S'_{max})^4\sigma_j^2 \tag{3.55}$$

holds. B

Proof

The map distance of the two models $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ is,

$$D_{ij}(\boldsymbol{g}_1, \boldsymbol{g}_2) = \frac{1}{MO}\sum_{m=1}^{M}\sum_{k=1}^{O}\{g_{1km}-g_{2km}\}^2 = \frac{1}{MO}\sum_{m=1}^{M}\sum_{k=1}^{O}\{s(u_{1km})-s(u_{2km})\}^2. \tag{3.56}$$

Using Eq. (3.46), the distance D_{ij} can be approximated as, $D_{ij} \cong \frac{1}{MO}\sum_m\sum_k\{s'(u_{1km})(u_{1km}-u_{2km})\}^2$

$$\le \frac{1}{MO}(S'_{max})^2\sum_k\sum_m\{\sum_{n=1}^{N} w_{kn}h_{1nm}-\sum_{n=1}^{N} w_{kn}h_{2nm}\}^2, \tag{3.57}$$

as it was done in the proof of Theorem 1, with h_{1nm} and h_{2nm} denoting the hidden layer outputs of the two models. It can be further written as,

$$\begin{aligned}
&= \frac{1}{MO}(S'_{max})^2\sum_k\sum_m\{\sum_{n=1}^{N} w_{kn}(s(u_{1km})-s(u_{2km}))\}^2 \\
&\le \frac{1}{MO}(S'_{max})^4\sum_k\sum_m\{\sum_n w_{kn}(u_{1km}-u_{2km})\}^2 \\
&= \frac{1}{MO}(S'_{max})^4\sum_k\sum_m\{\sum_n w_{kn}w_{nj}(v_{jm}-\mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}v_{im})\}^2 \\
&= \frac{1}{MO}(S'_{max})^4\sum_k\{\sum_n w_{kn}w_{nj}\}^2\sum_m\{v_{jm}-\mathrm{sgn}(r_{ij})\frac{\sigma_j}{\sigma_i}v_{im}\}^2 \qquad (3.58)\\
&= \frac{2}{O}(S'_{max})^4\sum_k\{\sum_n w_{kn}w_{nj}\}^2\cdot\sigma_j^2(1-|r_{ij}|), \qquad (3.59)
\end{aligned}$$

using $\{\mathrm{sgn}(r_{ij})\}^2 = 1$ and Eq. (3.49), again.

When channel j is to be fused with the bias, $v_i = v_b = 0$ holds and Eq. (3.58) becomes,

$$D_{bj} \le \frac{1}{O}(S'_{max})^4\sum_k\{\sum_n w_{kn}w_{nj}\}^2\cdot\sigma_j^2 . \tag{3.60}$$

(End of Proof)

Theorems 1 and 2 give the upper bound of the map distance when a channel is fused with another channel or with the bias. We will call the upper bounds in the right hand sides of Eqs. (3.43), (3.44), (3.54) and (3.55), as D_{ij}^{max}.

The definitions of D_{ij}^{max} indicate several interesting points. On fusion of two channels, the practice to select the channel pair with the maximum absolute similarity is beneficial, for it will suppress the distance of the two models. However, it is seen in Eqs. (3.43) and (3.54) that the variance of the channel to be eliminated should be taken into account as well. For the case of merging a channel with the bias, Eqs. (3.44) and (3.55) clearly shows that selecting a channel with minimum response variance will help to suppress the map distance.

In the neural network models used in this work, the maximum derivative of the transfer (sigmoid) function S'_{max} is always 1. Under this condition, the two Theorems indicate that D_{ij}^{max} are the functions of the connection weights as well. Besides the nature of the channel responses, the map distances can be suppressed by keeping the power of the weight vectors small. When repetitively reducing the network model, pruning a unit by fusion does not give rise to the training error at the early stage of training. However at the later stages of the training, unit fusion could cause a large increase in the training error, and a longer additional training was necessary to recover. Since the lengths of the weight vectors tend to grow monotonically as the BP training proceeds, the observed phenomenon can be explained by Theorems 1 and 2. In view of this fact, one way to keep the map distance caused by channel fusion may be the use of additional penalty terms in the error, which prevents the lengths of the weight vectors to grow too large. This approach has been reported in the context of structural learning with forgetting [18], which tries to decay the unnecessary weight connections, for obtaining a simpler network.

Only analyses of the proposed two model switching methods will be provided here. However, any additional model switching strategy can be rated in the equal measure of map distance. Therefore, use of map distance will suggest a multistrategy model switching scheme.

A straightforward way of selecting the switching candidate would be to use the upper bounds D_{ij}^{max} in the Theorems 1 and 2 among the candidates directly. In this case, the model candidate with a minimum D_{ij}^{max} will have the highest fitness. This method will be called the *minimum upper bound* (MUB) method.

As addressed above, the length of the weight vectors tend to grow monotonically when BP is used without any penalty terms in the training criterion. Therefore, it can be foreseen that the switching will occur only in the early stages of training when the MUB selection strategy is used. This nature can be beneficial since the inefficiency such as the herd effect is known to take place in the early stage of BP training. However, since the channels installed in place of the reduced channel will initially have a near-zero weight vectors, repetitive selection of a newly installed channel may occur, regardless of the efficiency of the channel's response pattern. In such a case, an exceptionally

longer passive epoch for the newly installed channel should be necessary.

As an alternative to the MUB selection, only the factors in D_{ij}^{max} that have their origin in the natures of the channel responses may be extracted to be the detection criterion. This may enable to set a switching criterion which is valid throughout the whole training process. It will also be equivalent to using the MUB in a network whose weight vectors are normalized. A measure named *effective map distance* (EMD) will be used as an alternative distance criterion. EMD Δ_{ij} consists of the factors originating in the channel response features among the upper bound of map distance D_{ij}^{max}, and will be defined as

$$\Delta_{ij} = \begin{cases} \sigma_j^2 & \text{if } i = b \text{ (bias)}, \\ 2\sigma_j^2(1 - |r_{ij}|) & \text{otherwise.} \end{cases} \tag{3.61}$$

Use of EMD for the fitness evaluation will be called the EMD method.

3.4.2 Map distance by unit splitting and installation

In expansion of the model, the new model includes the immediate model. Therefore, evaluating from viewpoint of map preservation, infinite variations of new models and maps which perfectly preserve the immediate map can be made. Therefore the map distances by unit splitting and installation were calculated as being zero.

3.5 Neural network pruning by unit fusion

In this section, the superiority of map preservation by unit fusion method will be shown in comparison with the other pruning methods

3.5.1 Pruning examples

The fusion method used for model size reduction (pruning) was tested on several problems in comparison with the other methods. Results of two typical examples will be presented here. Starting from a network with redundant set of hidden layer units, one unit was pruned by each method every time the network finished its training. After the pruning, the training was continued with the remaining network.

The increase of the error caused by the pruning and the number of presentations necessary for re-learning was recorded at every pruning event. Neurons with sigmoid activation function having (0, 1) output range in Eq. (1.4) were used. The networks were trained with the normal back-propagation learning rule, and no momentum term was used in updating the link weights. Here, the following four pruning methods were compared.

1) Random pruning (Referred to as the “Random” method.) : A hidden layer unit to be pruned was selected at random.

2) Pruning by similarity analysis (The rule 1 of Sietsma and Dow's method evaluated by unit

similarity. Referred to as “Similarity” method.) : By the similarity analysis of Eq. (3.5), either unit of the pair having maximum absolute correlation was pruned. No modification to the link weights were made. Rules 2 and 3 were not used.

3) Pruning by calculating the sensitivity of the error to unit removal (Mozer and Smolensky's method. Referred to as “Sensitivity” method.) : The sensitivity with respect to connection weight w_{ij} is defined as

$$S_{ij} = E(w_{ij} = 0) - E(w_{ij} = w_{ij}^f), \tag{3.62}$$

which is the increase of the error E caused by pruning link w_{ij}. Here, w_{ij}^f is the link weight in the redundant network finally reached by the initial training. The sensitivity defined in (3.62) was summed up for the connections from a hidden layer unit to all the output units, to calculate the error sensitivity to the removal of the unit, then the one yielded minimum sensitivity was pruned.

4) The fusion method presented in this thesis : The pair of neurons were selected by the similarity analysis of Eq. (3.5), and they were fused into one using the “synchronous” method when $r_{ij} > 0$, and the “alternate” method when $r_{ij} < 0$.

A. The Encoding Problem

The first problem is the 8 bit encoding problem which is to learn an identity mapping of orthogonal patterns [5]. The training input-output set consists of 8 pairs of identical unit vectors. Therefore, a reasonably compact network to solve the 8 bit encoding problem will have 8 input, 3 hidden and 8 output units (8-3-8 network). The initial network configuration was set to 8-10-8. After the learning was achieved, pruning and re-training took place alternately until it reached the minimal 8-3-8 set. For both initial and re-training processes, the learning constant η was set to 0.5. Each training session was continued until the mean squared error criterion of 0.0025 was reached. For each of the pruning methods, 100 trials were made starting from random initial connection weights and thresholds chosen from (–0.5, 0.5) range. In Table 3.1, it is seen that the fusion method reduced the number of training pattern presentation for re-learning to 40 ~ 80% compared to “sensitivity” method, which proved the second best. In the last column, the computational cost through the whole trial relative to that of random pruning is shown. The cost is computed with an assumption that the computational cost grows linearly with the number of the links [31]. The fusion method could cut 32% of the cost compared to random pruning.

Table 3.1. The number of training presentations for initial learning and re-learning after pruning in the encoding problem. The leftmost column for 10 hidden units is the initial learning and the other columns show the iterations to recover damage given by the pruning. In "Comp. cost" column, the relative cost of computation to reach the minimal network is listed.

H. layer units >	10	9	8	7	6	5	4	3	**Comp. cost**
1. Random	331.5	100.8	116.5	148.9	203.9	305.8	490.8	1243.2	**1**
2. Similarity	331.5	83.9	101.5	146.4	203.7	293.9	457.5	1258.2	**0.97**
3. Sensitivity	331.5	64.7	89.9	129.1	187.0	282.0	459.9	1251.7	**0.93**
4. Fusion	331.5	22.8	35.2	56.1	93.2	168.8	287.3	1052.6	**0.68**

B. Simple character recognition

```
...@...   @@@@@@@   ..@@@..   ..@@@..   @@@.@@@
...@...   .@....@   ...@...   .@...@.   .@...@.
..@.@..   .@.....   ...@...   @.....@   .@...@.
..@.@..   .@@@@..   ...@...   @.....@   .@...@.
.@@@@@.   .@.....   ...@...   @.....@   .@...@.
.@...@.   .@....@   ...@...   .@...@.   .@...@.
@@@.@@@   @@@@@@@   ..@@@..   ..@@@..   ..@@@..
```

Fig. 3.7. The 7×7 character patterns used for the pattern recognition problem. The input is 0 for '.' and 1 for '@'.

Next, the four pruning methods were tried on a simple character recognition problem. The task is to learn the 5 character patterns shown in Fig. 3.7. Each character consists of 49 pixels in the 7×7 domain. The network was trained to produce a different unit vector at the output for each of the 5 character patterns. The initial network was chosen to be a 49-10-5 network, and it was pruned to a 49-2-5 configuration. In contrast to the encoding problem, the appropriate size of the hidden layer is unknown. Here, the minimum network configuration was chosen in accordance with the maximum absolute similarity of the hidden layer units. This stopping criterion will be discussed later. The learning constant η, the error criterion and the range of initial random weights/thresholds were set at 0.5, 0.01 and (–0.5, 0.5), respectively. For each method, 100 trials were made. The results are shown in Table 3.2. It proved that the total computational cost was reduced to 2/3 using the fusion method compared with the random pruning. When the network configuration is quite redundant, the fusing method is greatly effective to reduce the re-training time. “Similarity” and “Sensitivity” methods also enable some reduction of re-learning epochs compared with random pruning. However, when the network is well pruned, “Similarity” and “Sensitivity” methods require same or even more re-learning epochs compared with the “Random” method. This fact indicates that it is important to

preserve the removed unit's feature especially when the network becomes small. In Fig. 3.8, the average increase of error due to the pruning is plotted. It is seen that pruning by fusion gave only 1/2 ~ 1/10 damage to the mean error compared with the other methods, and hence the recovery by post-pruning learning was easier throughout the iterated pruning.

Table 3.2. The number of training presentations for initial learning and re-learning after pruning for the character recognition network.

H. layer units >	10	9	8	7	6	5	4	3	2	**Comp. cost**
1. Random	79.9	20.3	23.8	29.3	42.1	56.4	96.6	173.0	1213.6	**1**
2. Similarity	79.9	17.5	19.1	25.7	35.0	47.1	82.2	167.1	1297.3	**0.98**
3. Sensitivity	79.9	9.0	13.9	20.0	26.8	39.3	86.0	174.6	1252.3	**0.93**
4. Fusion	79.9	0.1	0.6	1.9	5.3	10.2	23.0	128.0	1084.6	**0.68**

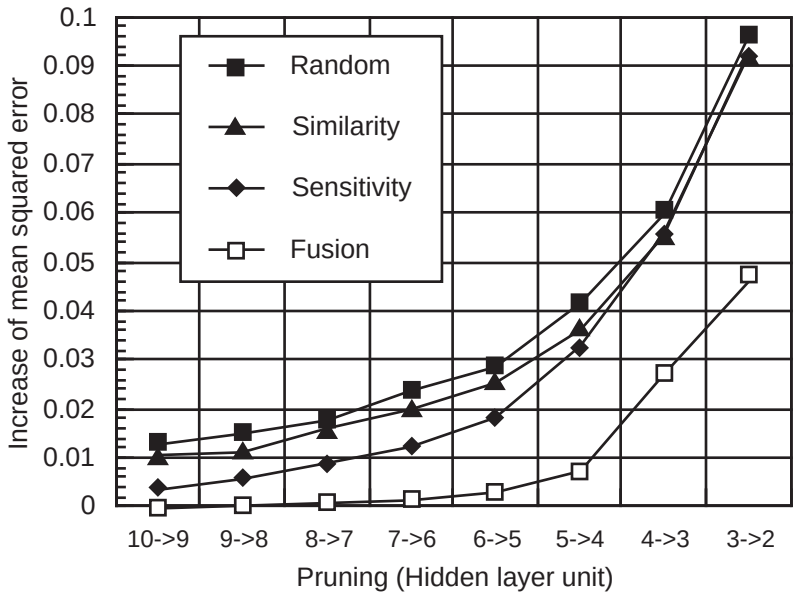


Fig. 3.8. The average increase of error when a unit is pruned, for each of the four pruning methods.

3.5.2 Discussion

Maintenance of linear independence among the extracted features

In the pruning experiments, the decision whether or not to prune another unit was made according to the maximum absolute similarity of the hidden layer units R, defined as,

$$R = max(|r_{ij}|) . \qquad (i, j \in 1,2, \ ..., \ N, \ i \neq j)$$

N: Number of hidden layer units. (3.63)

Through the application of the fusion method to many problems, it was observed that, when R reduces to values as small as 0.1 after several fusions, no more pruning can be done (Fig. 3.9), and further pruning will produce an overpruned network. However, there were a few exceptions. On problems such as the parity problem [5] (Output is 1 if the input has even bits of 1's, and 0 otherwise) and the rule-plus-exception problem[30][31] (Implementing $x\,y \vee \bar{x}\,\bar{y}\,\bar{z}\,\bar{u}$), maximum absolute similarity R were not small enough even when the networks were pruned to theoretically expected minimum sizes. So, it can be said that maximum absolute similarity R does not always reduce to a small amount when the minimum configuration is reached, however for most of the problems, it is a worthwhile rule of thumb that the pruning ought to be stopped when R does decrease to the vicinity of 0.1.

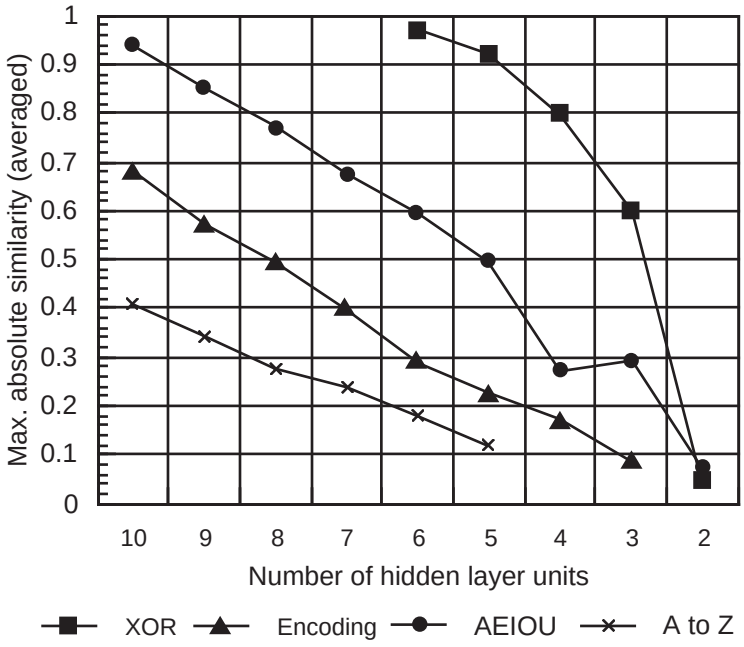


Fig. 3.9. The change of maximum absolute similarity as the pruning proceeds.
XOR: Learning the EX-OR function. Minimal network is 2-2-1.
Encoding: The 8-bit encoding problem. Minimal network is 8-3-8.
AEIOU: The recognition problem of the five character patterns 'A','E','I','O' and 'U' shown in Fig. 3.7. A 49-2-5 net was minimal.
A to Z: The recognition of full 26 alphabet character set from which the training set in Fig. 3.7 originate. A 49-5-5 net was minimal.

When the unit responses in Eq. (3.5) can be assumed to be binary values, i.e. $h_i,\ h_j \in \{0,\ 1\}$, the similarity measure of two units r_{ij} can be written as,

$$r_{ij} = \frac{Mm_{ij} - a_i a_j}{\left\{a_i a_i (M - a_i)(M - a_j)\right\}^{1/2}}, \tag{3.64}$$

with P, a_i, a_j and m_{ij} being the cardinalities of the training pair set, training subset which satisfies

$h_i = 1$, $h_j = 1$, and $(h_i = 1) \cap (h_j = 1)$, respectively.

The linear relation of m and r_{ij} of a case when $M = 100$ and $h_i = h_j = 50$ in Fig 3.10 shows that the section of m_{ij} which gives the *non-fusing range* of $-0.1 \le r_{ij} \le 0.1$ will be, $22.5 \le m_{ij} \le 27.5$. It is clear that when the maximum absolute similarity R falls in this range, the responses of the hidden layer units against the training inputs are almost perfectly independent, and further pruning will cause a overpruned state that can be hard to recover with the remaining set of units.

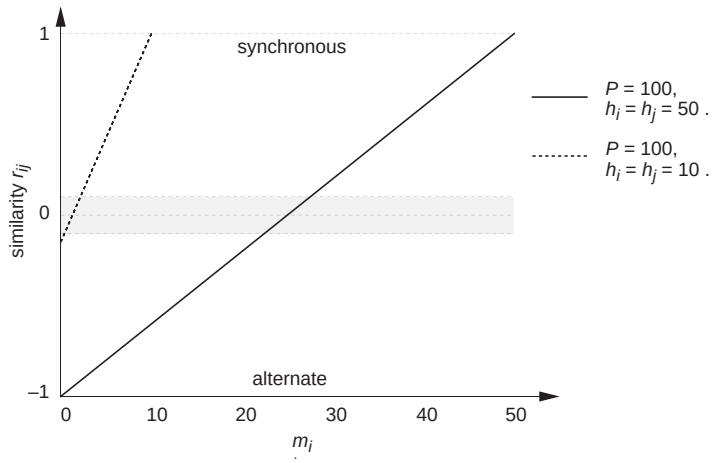


Fig. 3.10. The linear relations between the number of the training sets that the two units jointly fire (m) and the unit firing similarity (r_{ij}). The shaded area shows the *non-fusing range* of $-0.1 \le r_{ij} \le 0.1$.

The example above assumes the case when the training points in the feature space are evenly divided to the discrimination borders defined by the hidden layer units, which also corresponds to the case which the hidden layer firing pattern variances are at their maximum. In actual training of the network, this is rarely the case. Another linear relation of m_{ij} and r_{ij} when P =100 and $h_i = h_j$ = 10 is also plotted in Fig. 3.10. This is a case which both units fire only to the small portions (10%) of the whole training set. This type of firing patterns will be necessary in a classification problem of many classes with a *winner-takes-all* encoding in the hidden layer, such as clustering using localized kernel functions as in the case of Radial Basis Function (RBF) networks. It is seen in Fig. 3.10, that even when m=0, corresponding to a case when the two units are firing against a separate subset, the units will be fused in the alternate mode, which is not desirable.

It is not possible to clearly determine a non-fusing range for all kinds of classification problems. Examples of relatively inefficient and efficient cases of feature space segmentation are shown in Fig. 3.11. In the real-world problems, the solutions will be the combinations of the two cases. Namely, the distributions of the training sets of a classification problem may allow for some classes to be encoded efficiently as in Fig. 3.11(b), but for some classes a winner-takes all encoding

of Fig. 3.11 may be unavoidable.

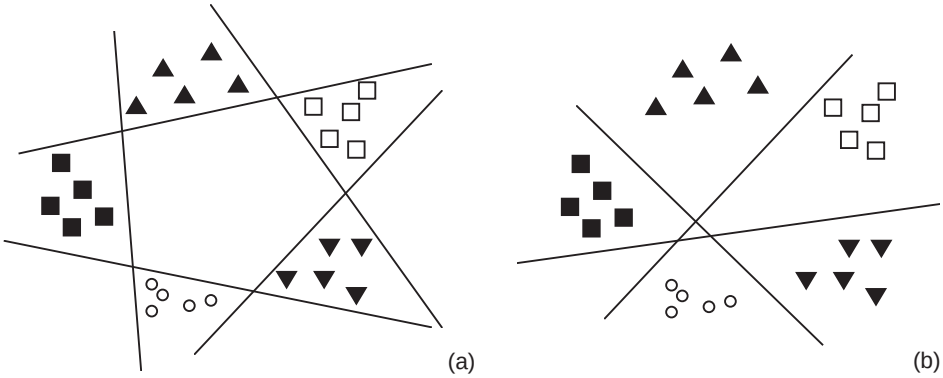


Fig. 3.11. Distribution of the training inputs in the feature space for a virtual five class classification problem. P=25. (a) The winner-takes-all encoding where a hidden layer unit fires selectively to a particular class input. For any hidden unit pair (i, j), a_i=a_j=5 and m=0. Thus, r_{ij}= -0.25. (b) Segmentation of the feature space in a more efficient way with less hidden units. For any hidden unit pair (i, j), a_i=a_j=10 and m=5. Thus, r_{ij}= 0.17.

When the class distributions have intricate borders, even the winner-takes-all encoding may not be sufficient. However, it may be a worthwhile criterion to set the non-fusing border R to allow the winner-takes-all encoding for all the classes.

The merit of model switching by weight compensation

The fusion method can be used as a new technique for pruning redundant hidden units in layered neural networks which, in contrast to the conventional unit removing methods, fuses two units into one so that the properties of the two are preserved. Through computer simulations, it was shown that the re-learning required to regain the capability after pruning can be reduced drastically where the network is well redundant, and to 70 ~ 80% even when the net is sub-minimal compared with the other pruning methods. This amounted to 30 ~ 40% reduction of the total computational cost to reach the minimum configuration on our simulations. Although additional tasks such as firing pattern similarity analyses to choose the fusing pair candidates and modifications on link weights are necessary, they are required only once before pruning. These tasks correspond to negligibly small amount of computational overhead in the course of iterated training. It was seen that when the fusing method was used, the damage given to the network (or the increase of error) was significantly smaller (1/2 ~ 1/10) than the other methods. This feature enables the quick recovery after the pruning even when the network is less-redundant and conventional pruning methods are ineffective. The strategy to fuse the pair of units having maximum absolute similarity is most effective when the training patterns are evenly distributed in the input pattern space. When the training patterns are quite unevenly distributed, the method may lose advantage by attempting to fuse a non-redundant pair of similarly

firing units.

A pruning technique similar to the fusion method have been independently proposed by Sietsma and Dow [33], where only the weights between the hidden and the output layer are changed in the similar way as proposed here. A comparison in the style of the experiments in the former section has been done in [42]. There the relative computational cost to reach the minimal network rated (Proposed fusion method : Sietsma & Dow's method = 0.68 : 0.71), which showed the advantage of the fusion method which employs the modifications of the weights between the hidden and the input layers. This fact was also confirmed by the comparison of the average increase of MSE when a unit was pruned, where the fusion method was superior.

3.6 BP training with MS controlled by the MS Index

In view of the fusion method's success in network pruning, a network that automatically adjusts itself to a minimal size, according to the framework introduced in Chapter 2, was developed. The proposed training algorithm selecting the occasion of MS by the MS Index was compared with a conventional method. From the experiments, it will be shown that the proposed method enables to obtain a small model with high generalization ability, in a very efficient manner.

In Table 3.3, the features of the training method and the network is summarized.

Table 3.3. Features of the network and the training method.

Objective of MS	1. Efficient training 2. Small model realization
General • Network type	3 layered network with 2 layers of first order sigmoid units.
• MA scheme	MS consulting MS Index. Add/remove hidden layer units.
MS • Scheduling	Unlimited model
• Fitness Evaluation	Map preservation (inheritance)
• Fitness Index	$I_F(f_N, f'_N) = 1 - D(f_N, f'_N)$
• Stress Index	$I_S(E, \frac{\partial E}{\partial t}) = (E - E_0)(\frac{\partial E}{\partial t} + I_0)$
• MS Index	$I_{MS} = I_F I_S$
• MS candidate $(I_{MS} \geq 0)$	Unique map by unit splitting
$(I_{MS} < 0)$	All possible by unit fusion

3.6.1 Experiment – Training speed and generalization ability –

The proposed training method was tried in a classification of 7×7 bitmap patterns of the 26 alphabet characters. Some of the patterns to be learned are shown in Fig. 3.7. The training process of the proposed network whose features are shown in Table 3.3 was compared to another network employing conventional MA timings.

Common conditions

Training input :	26 alphabet characters in 7×7 bitmap
Output encoding :	Unit vector output
Initial network model :	(input-hidden-output) = (49 - 1 - 26)
Target error E_0 :	0.01
Training gain η :	0.04
Initial map after MA :	Unit fusion and splitting
Selection scheme :	Map distance (map preservation)

Termination condition : Trained network of (49 - 5 - 26) model reached.

Network 1 (BP with MS : proposed)

MA occasions : Determined by the MS Index.

Misc. settings : $I_0 = 2.0\times10^{-3}$, $I_{MS}^{+} = 1.4\times10^{-4}$, $I_{MS}^{-} = -5.0\times10^{-8}$

Network 2 (BP with conventional MA timings)

MA occasions : Expansion : Split when not trained after $N_{passive}$ epochs.
Reduction : Fuse when trained.

Misc. settings : $N_{passive} = 50$

A. MS and training speed

In Fig. 3.12, the change in the number of hidden layer units and the training error for the two networks are shown for a typical trial.

In the conventional approach, the initial model expansion is very slow. This may seem to be due to the long interval (50 epochs) for evaluation of the training progress, however, when the interval was made smaller, there were cases such that the training would not end due to repetitive expansion and reduction of the model size after the training have mostly converged.

In the proposed method using MS, besides the fast expansion of the model, it should be noted that the reduction of the model has been started already before the error have converged. Quick initial expansion and prediction of convergence contributes to the quick learning in the method with MS.

The ratio of the computational time for the two methods averaged for 10 trials were (conventional : MS) = (100% : 69.8%) . It is clear that the method enables to obtain a small model very efficiently.

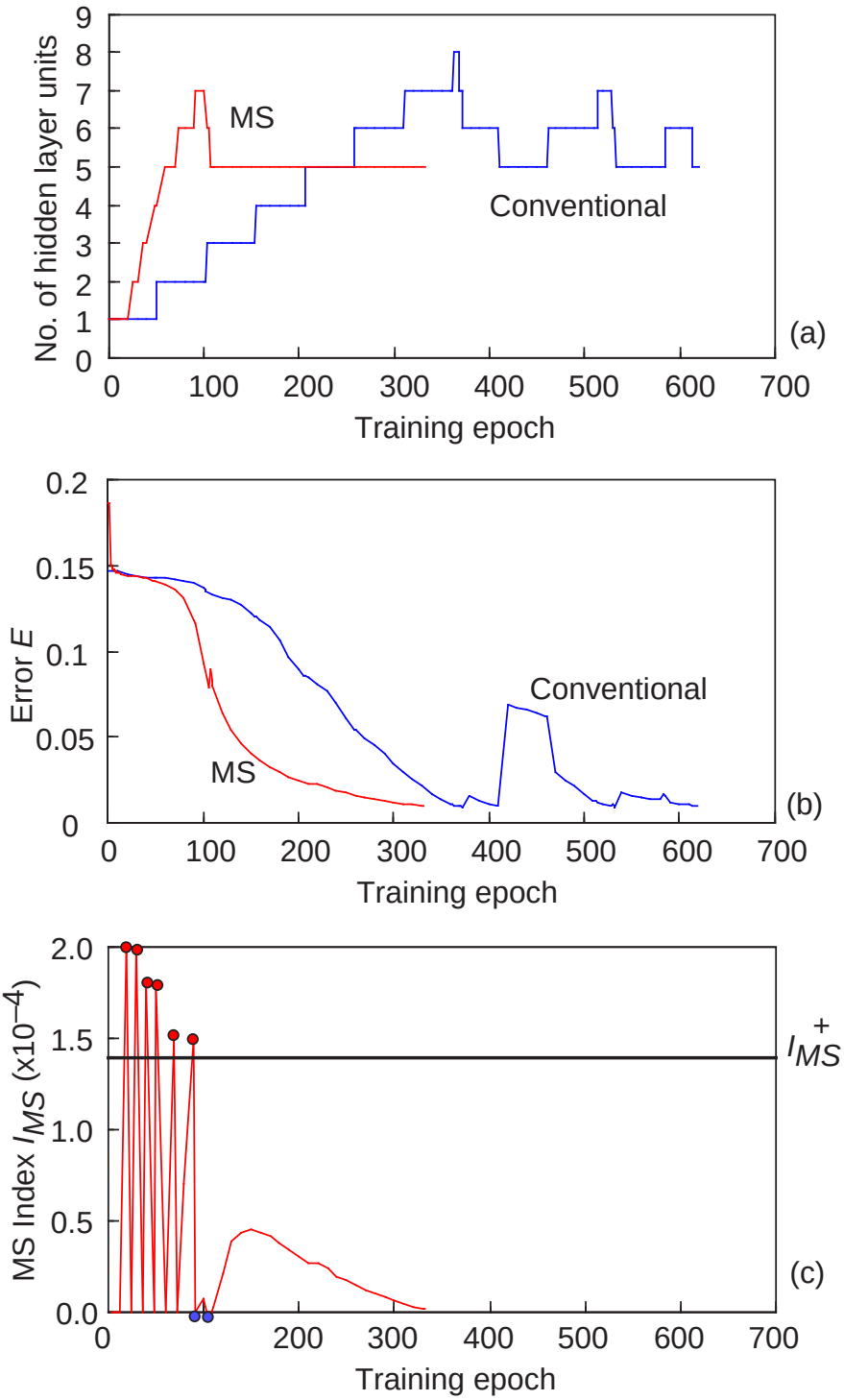


Fig. 3.12. The change in the number of hidden layer units, training error and the MS Index for the proposed training method with MS and the method with conventional MA occasions. MS occasions are shown with circular markers in (c).

B. Escaping from local minima

When the minimal 49-5-26 network with no MA was used, almost half of the training attempts got

caught in a local minima of the error potential function. However, when the proposed training method with MS was used, quick escape from the local minima was also possible as shown in Fig. 3.13, allowing all the attempts to reach the global minimum.

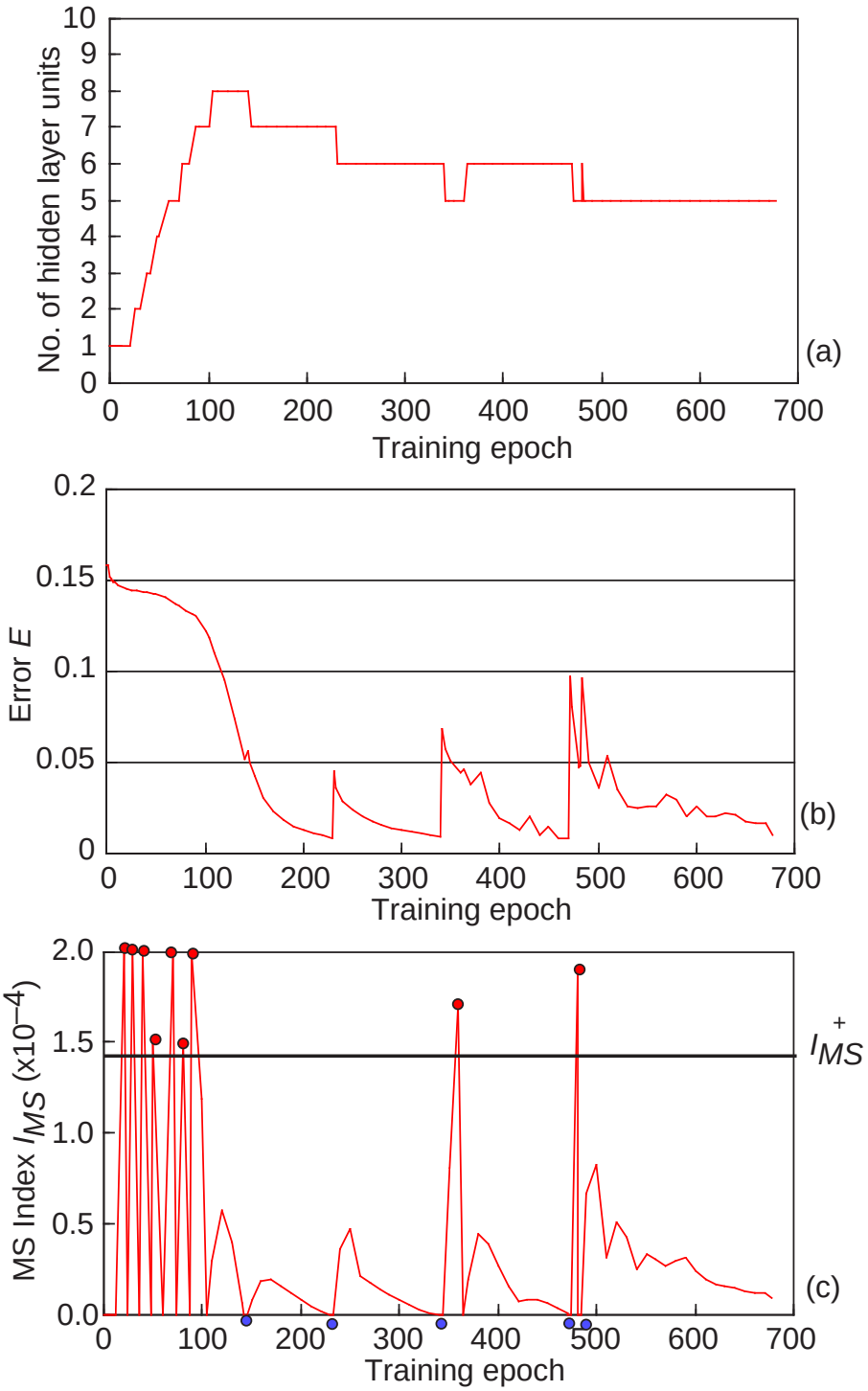


Fig. 3.13. The progress of training for the case when the proposed method was captured at a local minima (ca. 350 epoch). After several attempts to escape by re-expanding the model, the training was successfully accomplished.

C. Initial map selection by unit fusion/splitting and generalization ability

When the networks trained by the MS method was evaluated by a noisy input, it was found that the they possessed higher generalization abilities when compared with networks trained without any model alterations. The reason for this result will be discussed in the next section.

On testing, the networks were evaluated by test sets with random 2 positions among the 49 inputs being reversed as the added noise. The average rates of correct recognition for 10 trials are shown in Table 3.4.

Table 3.4. Comparison of the generalization ability by the recognition rates of the noisy inputs.

Model	Recognition rate (%)	
	Plain BP	BP with MS
49-5-26	93.1	91.5
49-6-26	95.0	92.3

3.7 Initial map selection by unit fusion and generalization ability

3.7.1 Projection learning

In [54]-[58], Ogawa and his group give a formal framework for implementing various classes of generalization abilities to layered networks. Here, their results will be summarized, mainly weighting to the essence of projection learning.

Conditions and notations

Target function

The theory of neural network training in [54]-[58] is formalized as a method for obtaining the best approximation function of the true scalar function $f(\boldsymbol{x})$, with $\boldsymbol{x}$ being an input vector. The function f is assumed to be an element of a reproducing kernel Hilbert space (RKHS) denoted by H. The reproducing kernel in H can be expressed as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{s=1}^{\dim H} \varphi_s(\boldsymbol{x})\varphi_s(\boldsymbol{x}') \tag{3.65}$$

using an orthonormal basis $\{\varphi_s(\boldsymbol{x})\}_{s=1}^{\dim H}$ in H.

Neural network

The network which tries to implement an approximation function f_0, has a layered structure of L

inputs, N hidden units and one output unit. The hidden layer units work as an array of basis functions $\{h_n\}_{n=1}^N$, which spans a function subspace in H denoted by S_N. The weights from the hidden layer to the output unit form a set $\{w_n\}_{n=1}^N$, also expressed as, $\boldsymbol{w} = (w_1\ w_2\ ...\ w_N)^T$ in a vector form. The approximation function implemented by the network will be,

$$f_0(\boldsymbol{x}) = \sum_{n=1}^{N} w_n h_n(\boldsymbol{x})\ . \tag{3.66}$$

Training set

The training set consisting of true input-output pairs is denoted by $\{(\boldsymbol{x}_m, y_m)\}_{m=1}^M$. Here, $\{y_m = f(x_m)\}_{m=1}^M$ holds. The selection of the training inputs will introduce the sampling operator

$$A = \sum_{m=1}^{M} (\boldsymbol{e}_m^{(M)} \otimes \overline{K(\boldsymbol{x}, \boldsymbol{x}_m)}), \tag{3.67}$$

where $\{\boldsymbol{e}_m^{(M)}\}_{m=1}^M$ is a set of M dimensional unit vectors with their m-th element being unity. The operator " $\cdot \otimes \cdot$ " denotes the Schatten product defined by $(f_1 \otimes \overline{f_2}) f_3 = (f_3, f_2) f_1$. The $\langle f_1, f_2 \rangle$ operator denotes the inner product of f_1 and f_2. Using the sampling operator, the training outputs $\boldsymbol{y} = (y_1\ y_2\ ...\ y_M)^T$ can be expressed as $\boldsymbol{y} = Af$. Additionally the reproducing kernel matrix $\boldsymbol{K}$ is defined as $\boldsymbol{K} = AA^*$.

Projection learning

The projection learning is an estimation of the orthogonal projection of the original function f to a subspace in H. Since the maximal subspace which this is possible is $R(A^*)$, an orthogonal projection of f to $R(A^*)$ denoted as f_0 will be estimated. The approximation function f_0 obtained in this way is optimal in the nature that it is the closest function to f within $R(A^*)$. The learning operator of projection learning can be written as,

$$A^{(P)} = A^\dagger + Y(I_M - AA^\dagger), \tag{3.68}$$

where Y is an arbitrary operator mapping from R^M to H, I_M is the $M \times M$ identity matrix, and $A^\dagger$ is the Moore-Penrose generalized inverse of the sampling operator A. The process of training set selection and estimation of the approximation function in projection learning is illustrated in Fig. 3.14.

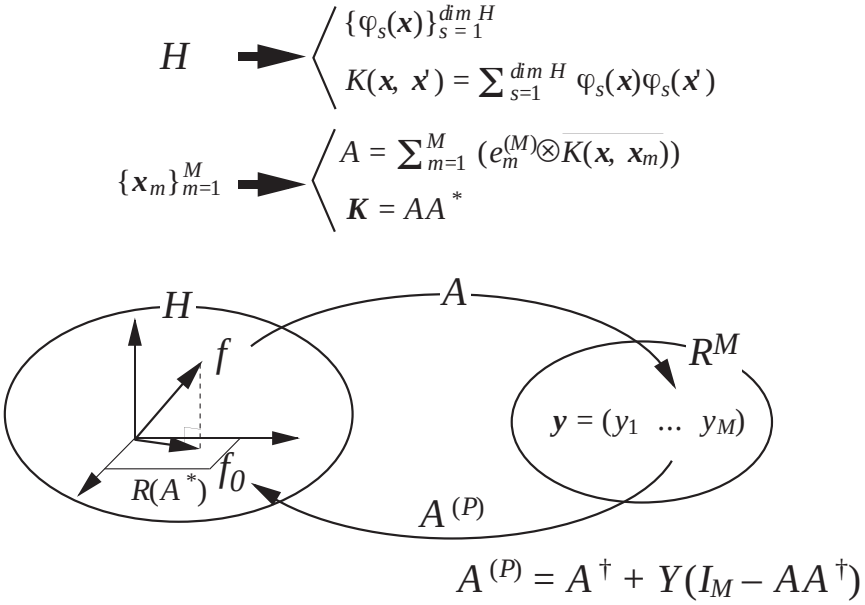


Fig. 3.14. The process of training set selection and projection learning. Selection of the RKHS H will determine the set of basis functions and the reproducing kernel. Further, selection of the training inputs will determine the sampling operator and the reproducing kernel matrix.

Projection Generalizing Neural Network (PGNN)

The projection learning can be implemented in a neural network by the following procedures.

1. Select the number of hidden layer units as $N \geq rank\ \boldsymbol{K}$.
2. Select the hidden layer basis functions $\{h_n\}_{n=1}^{N}$ so that $S_N \supset R(A^*)$ holds.
3. Set the weight vector as

$$\boldsymbol{w} = \{(U^*)^\dagger A^{(P)} + (I_N - UU^\dagger)T\}\,\boldsymbol{y}\ , \tag{3.68}$$

where T is an arbitrary $N \times M$ matrix, and operator U is defined as

$$U = \sum_{n=1}^{N} (e_n^{(N)} \otimes \overline{h_n(\boldsymbol{x})})\ . \tag{3.69}$$

The network function determined in this way will be equivalent to the projection learning, for,

$$f_0 = U^*\boldsymbol{w} = A^{(P)}\boldsymbol{y}\ . \tag{3.70}$$

Admissibility of memorization learning with respect to projection learning

Since the BP training of layered networks using the criterion in Eq. (1.1) is memorization learning, the conditions when memorization learning will implement projection learning at the same time, is most important. In the following, the memorization learning criterion and the projection learning criterion will be denoted as J_M and J_P, respectively. The set of learning operators obtained by criterion J will be denoted as $A\{J\}$.

Partial admissibility $A\{J_P\} \subset A\{J_M\}$ always holds. Further, under conditions of $N(A) = H$ or

$N(A) = \{0\}$, complete admissibility $A\{J_P\} = A\{J_M\}$ holds. The former condition of complete admissibility is of no practical importance for it is equivalent to the case when all training outputs are 0. The necessary and sufficient condition of the latter condition has been derived to be,

$$rank\ \boldsymbol{K} = dim\ H\ . \tag{3.71}$$

If the training set which defines the reproducing kernel matrix $\boldsymbol{K}$ is selected so that Eq. (3.71) holds, projection learning will always be achieved by memorization learning.

3.7.2 Hidden layer basis functions limited by unit fusion

In case of training a three-layered network such that the basis function set $\{h_n\}_{n=1}^N$ is also acquired by the training process. Additionally, since there is no freedom of choosing the training sets at will, the condition in Eq. (3.71) cannot be satisfied.

However, even under such constraints, there can be several measures to take for improving the generalization ability of the network, namely to make the approximation function closer to f_0.

Since the search space for the approximation function will be spanned by the hidden layer units basis $\{h_n\}_{n=1}^N$, it is better to employ a basis function set that are linearly independent to each other. However, as the $N(A)$ component in the subspace spanned by the basis will be harmful to the generalization ability, *N(A)* component should be reduced.

The basis function can be written in a form of sums of the components of $R(A^*)$ and $N(A)$ as,

$$h_n = n_n + r_n\ , \qquad (n = 1,2,\ ...,\ N) \tag{3.72}$$

$$n_n \in N(A)\ , r_n \in R(A^*)\ .$$

The responses of the hidden layer units against the training inputs $\{\boldsymbol{x}_m\}_{m=1}^M$ are,

$$h_n(\boldsymbol{x}_m) = r_n(\boldsymbol{x}_m). \tag{3.73}$$

Therefore, in detecting the reducible factors by unit fusion, similarity and variance reflect the nature of components $\{r_n\}_{n=1}^N$. In unit fusion, the reduction candidates are units such that, (a) the response variance is low (variance detection), or (b) response is similar to another unit (similarity detection). When (a) is applied, it is clear that it will contribute to the units with less $r_n \in R(A^*)$ components. This may contribute to "pruning" the $n_n \in N(A)$ factors. When (b) is applied, the redundancy among the $r_n \in R(A^*)$ will be reduced. In case of the above experiments, it can be considered that reduction of $n_n \in N(A)$ factors are contributing to improvement of the generalization ability.

When further limitations such as $\|h_n\|^2 = 1$ applies in a case of limited model scheduling introduced in Sec. 2.3.2 in which the the reduced basis functions will be re-defined and reused, repetitive hidden layer unit fusion and installation can gradually reduce the $n_n \in N(A)$ factors, and make the subspace spanned by the basis function set $\{h_n\}_{n=1}^N$, close to $R(A^*)$, which will further contribute to the improvement of the generalization ability.

3.8 Smooth model switching controlled by the MS Index

In this section, a variant of the training method governed by the MS Index, which allows a smooth change in the model by controlling the linear independence of the responses of the hidden layer units will be introduced. In the following, unit fusion is done only by the similarity detection. Therefore the Fitness Index for map preservation defined in Eq. (3.2) is evaluated in EMD method and further simplified to,

$$I_F(f_N, f'_N) = 1 - |r_{ij}| \tag{3.74}$$

where r_{ij} denotes the similarity of the channel pairs to be fused when switching from map f_N to f'_N.

3.8.1 Linear transformation and the modified training rule

As seen in the previous sections, model switching, especially pruning causes an increase of error. This increase of error is caused by the mapping discrepancy in the two maps, which can sometimes cause the training to lose the accumulated map totally.

Here, reduction of the increase in the error caused by unit fusion and splitting have been tried by changing the weights from the hidden layer to the output layer gradually, spending several additional training epochs, instead of a sudden change. In the following, the units use bipolar activation functions defined in (1.5) with (–1, 1) output range. Now, let us think of a matrix defined as

$$X = \begin{bmatrix} 1-|\xi| & \xi \\ \xi & 1-|\xi| \end{bmatrix}. \tag{3.75}$$

The parameter ξ in X is varied within the (–0.5, 0.5) domain. If $\xi = 0$, $X = I$. If $\xi = 0.5$, then $(o_i' \; o_j')^{\mathbf{T}} = X(o_i \; o_j)^{\mathbf{T}}$ would be a projection to $o_i' = o_j'$. And if $\xi = -0.5$, $X(o_i \; o_j)^{\mathbf{T}}$ would be a projection to $o_i' = -o_j'$. This transformation will be applied to the output signal of the unit pair which is to be fused, or the pair just split. Since the parameter ξ controls the effective independence of the two channels, it will called the *crosstalk* parameter.

When a pair is to be fused, ξ will either be increased or decreased starting from ξ_0=0, thereby effectively limiting the independence of the two channels with the BP training continuing. By employing this gradual transformation, the mapping gap between the models on switching can be reduced to zero. Similarly, when a pair was made by splitting, ξ will be initially set to ξ_0=0.5, forcing the pair to fire synchronously. Then, ξ will be gradually decreased to 0.

The parameter ξ will be changed externally using the amount of the MS Index I_{MS}. Parameter ξ will be calculated as the function of I_{MS} as,

$$\xi(I_{MS}) = \xi_0 - \mathrm{sgn}(r_{ij})\, F\left(\sum_t I_{MS}(t)\right), \tag{3.76}$$

where F is a monotonically increasing function, and the sum of I_{MS} is taken since the last time ξ was reset to ξ_0. The scheme of splitting and fusion in relation with the firing modes and the crosstalk parameter ξ is shown in Fig. 3.15.

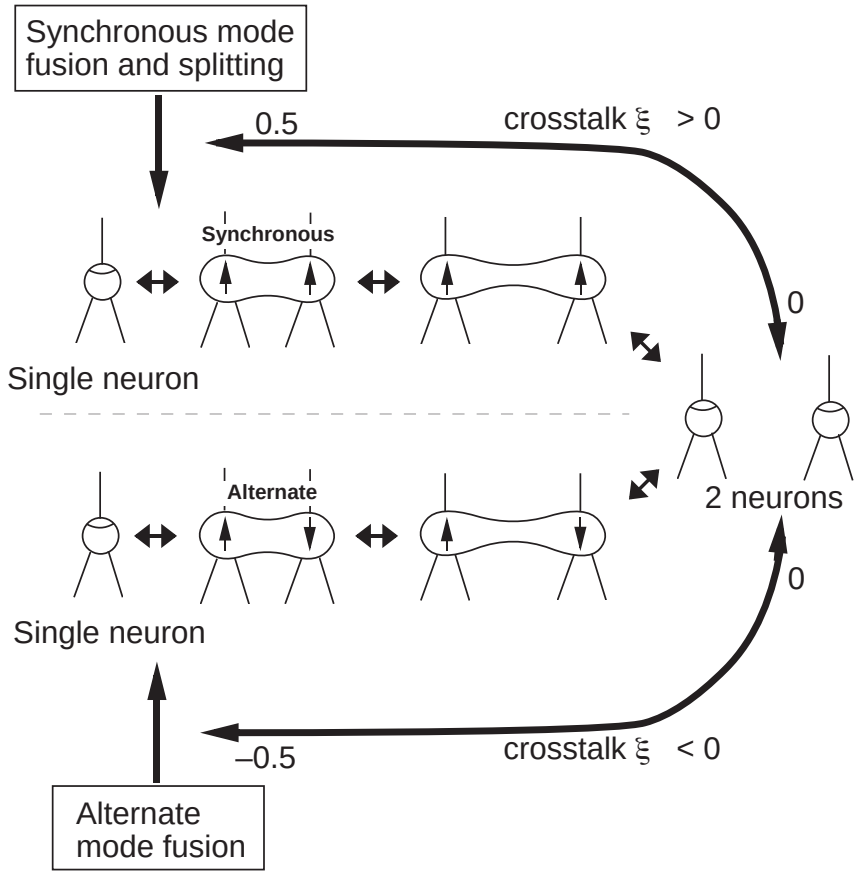


Fig. 3.15. The scheme of the continuous fusion and splitting of a unit pair controlled by the crosstalk ξ.

The linear transformation by matrix X affects not only the forward signals but also the backward error signals. In order to decrease the error in gradient descent manner, the backpropagated error δ in Eq. (1.9) must be redefined for the units with the transformed outputs. For the units i and j, δ_{im} and δ_{jm} are,

$$\delta_{im} = \sum_{k=1}^{O} \left[\delta_{kn} \left\{ w_{ki} (1 - |\xi|)\, s'(u_{im}) + w_{kj}\, \xi\, s'(u_{jm}) \right\} \right], \tag{3.77}$$

and

$$\delta_{jm} = \sum_{k=1}^{O} \left[\delta_{kn} \left\{ w_{ki}\, \xi\, s'(u_{im}) + w_{kj} (1 - |\xi|)\, s'(u_{jm}) \right\} \right], \tag{3.78}$$

respectively.

3.8.2 Phase transition during the training

The training with hidden layer unit splitting and fusing has four activation phases. “Active”,

"Passive", "Intermediate-Synchronous" and "Intermediate-Alternate". The transition flow of the phases are illustrated in Fig. 3.16.

Passive Phase

This mode takes place for a certain length of time, when the network starts learning or immediately after the unit fusion. This phase is necessary because the error value tends to be disturbed for a while after such events. During this phase, the network is trained with plain BP rule. After a given length of time (epochs), the network turns to be in the Active phase.

Active Phase

This is a short (only one epoch) phase after the Passive mode. The flow is branched to other phases according to the MS Index I_{MS} . On reducing the model, a new pair of hidden layer units are chosen through similarity analysis, and the network will enter an Intermediate phase to fuse them. When expanding the model, a unit is selected and will be split. Transformation parameter ξ of the split pair would be set to 0.5.

Intermediate Phase (Synchronous / Alternate)

This is when a pair of units is on their way of fusing, or has been split but still not throughly separated ($\xi \neq 0$). Because the direction of change depends on I_{MS}, a pair once started to fuse ($\xi \rightarrow 0.5$ or $\xi \rightarrow -0.5$) may stop and return to an independent pair of units ($\xi \rightarrow 0$). There are two subphases in this phase; *Synchronous* and *Alternate*, denoting the firing modes of the selected unit pair. In the Synchronous phase, ξ is limited to be within [0, 0.5]. At every training epoch, the value of ξ is updated according to Eq. (3.76). When ξ reaches 0, the units are independent from each other. At the other extremity when $\xi = 0.5$, the units are fused. In the Alternate phase, ξ is in [–0.5, 0] domain. Again, when $\xi = 0$, the channels will be independent and when $\xi = -0.5$, two units will be fused. The Passive phase will take place after the Intermediate phase.

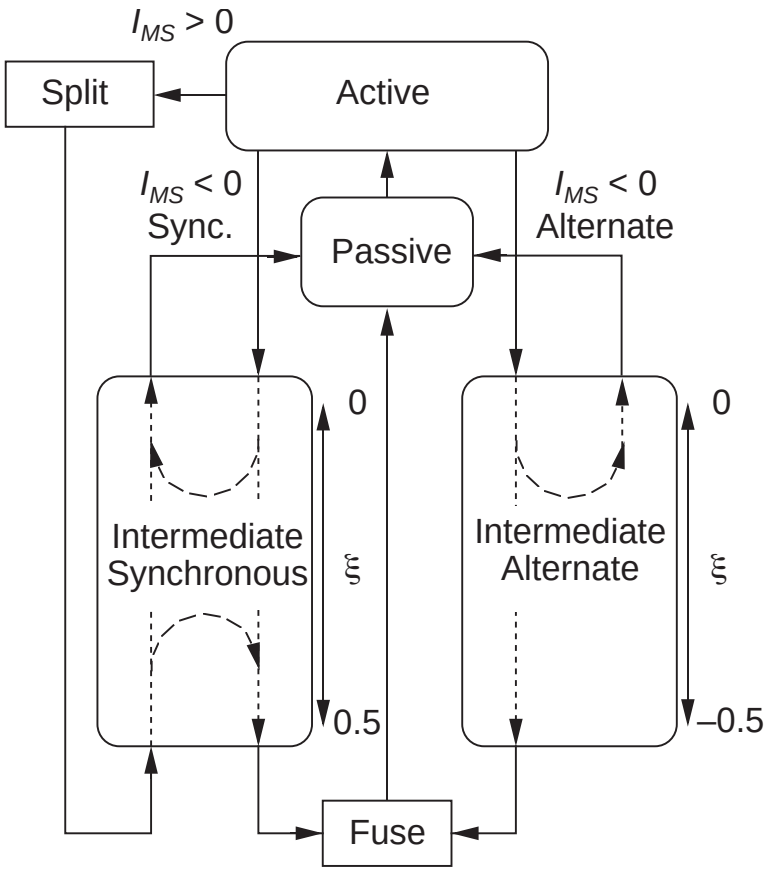


Fig. 3.16. The phase transition flow during the training of the network with hidden layer unit splitting and fusing.

3.8.3 Experimental results

The proposed learning algorithm with model switching was applied to networks being trained for two problems, to observe the ability of self-adjusting the hidden layer size. The problems tried were the XOR problem and an alphabetical character recognition problem. As for the function F for calculating ξ from the accumulated stress, a bipolar double sigmoid function with hysteresis as defined and plotted in Eq. (3.79) and Fig. 3.17, respectively, was used. The saturating growth of the sigmoid function is employed for stabilizing the effect of the accumulated stress, and the hysteresis curve is for avoiding the *jittering* repetition of splitting and fusing.

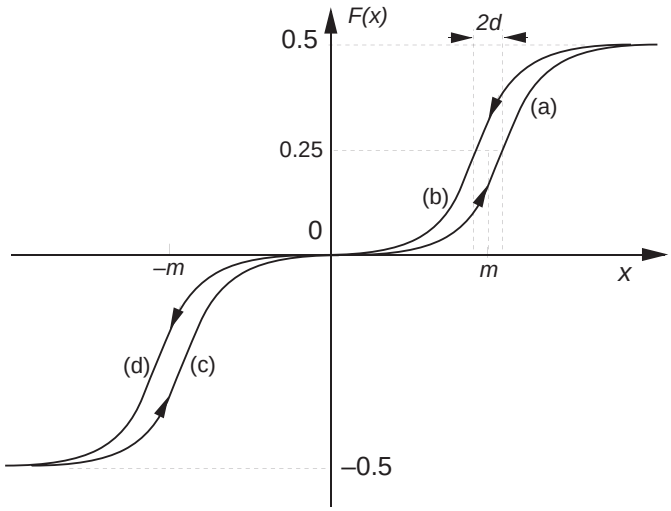


Fig. 3.17. The attenuation function with hysteresis applied to the accumulated stress for calculation of the crosstalk ξ. The four curves (a) – (d) are defined as

$$F(x) = \begin{cases} a(x - m - d) & \text{(a) Synchronous, Fusion} \\ a(x - m + d) & \text{(b) Synchronous, Splitting} \\ a(x - m - d) - 0.5 & \text{(c) Alternate, Splitting} \\ a(x - m + d) - 0.5 & \text{(d) Alternate, Fusion} \end{cases}, \qquad (3.79)$$

where

$$a(x) = 0.5\{1 + exp(-x)\}^{-1}. \qquad (3.80)$$

For m and d, 0.25 and 1.0×10^{-4} were used.

XOR problem

This is a famous problem of training a network to mimic an XOR logic gate. Since the class distribution of the input patterns are not linearly separable, it is not learnable by a single perceptron [2][5]. The minimal network configuration in the three layer style is known to be a 2-2-1 network. The initial network was set to be a 2-1-1 network, which is has an equivalent learning ability to a perceptron. The parameters were set to $\eta = 0.1$, $E_0 = 0.04$ and $I_0 = 0.1$, respectively. The number of the Passive epochs was 20. The number of hidden layer units, the error E and the crosstalk ξ are plotted throughout a training process of the network in Fig. 3.18.

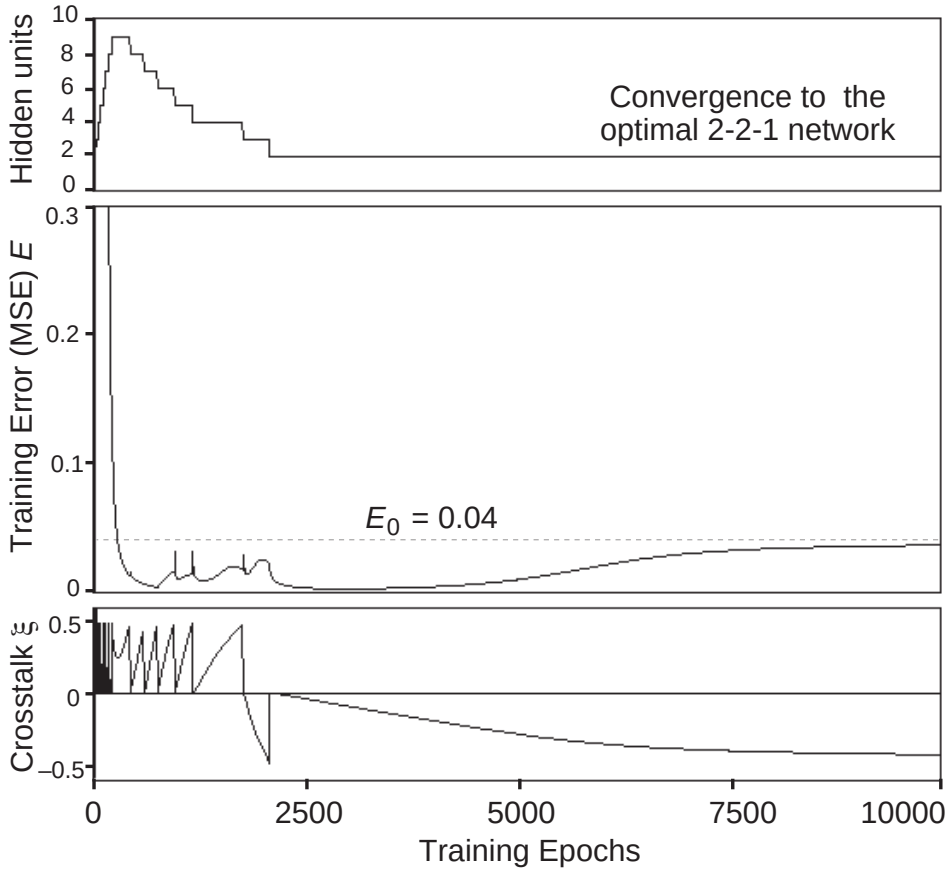


Fig. 3.18. The change of the number of hidden layer units, the error E and the crosstalk ξ during training the network for the XOR problem.

In Fig. 3.18, it is seen that the number of hidden layer units quickly increased initially. After the accomplishment of the target error criterion E_0, the accumulated stress turned to negative, and the hidden layer units were fused one by one. Even after reaching the minimal 2-2-1 network, the accumulated stress was still negative, and the (negative) crosstalk between the final unit pair grew. This resulted in the increase of the error, which converged at the target E_0.

Alphabetical character recognition

To see how the system performs when the size of the network is larger, it was tried in a classification of 7×7 bitmap patterns of the 26 alphabet characters. The network was trained to produce a 26-bit unit vector assigned to each character pattern. According to the preliminary experiments using MLP's without hidden layer adjustment, the minimal size that could obtain the map was a configuration of 49-5-26.

On trying with the proposed network, the initial configuration of the network was set to 49-1-26, and the parameters of $\eta = 0.1$, $E_0 = 0.01$ and $I_0 = 0.1$ were used. The number of the Passive

phase epochs was set to 10.

The result of the network's training and size-adjustment is shown in Fig. 3.19. The splitting and fusing occurred in the same manner as in the XOR problem. After pruning to 49-5-26 network, the error E rose and trembled at $E \sim E_0$ (= 0.04). There, the curve of parameter ξ became flat. Therefore, the network was regarded to have reached the optimal size when {The change in ξ during the past 500 epochs} < 0.01. Out of 20 trials, one attempt failed to converge, 4 converged to a 49-6-26 network, and 15 converged to the minimum 49-5-26 network. Thus, it can be concluded that the network adjusts itself to a sufficiently small size at a high rate.

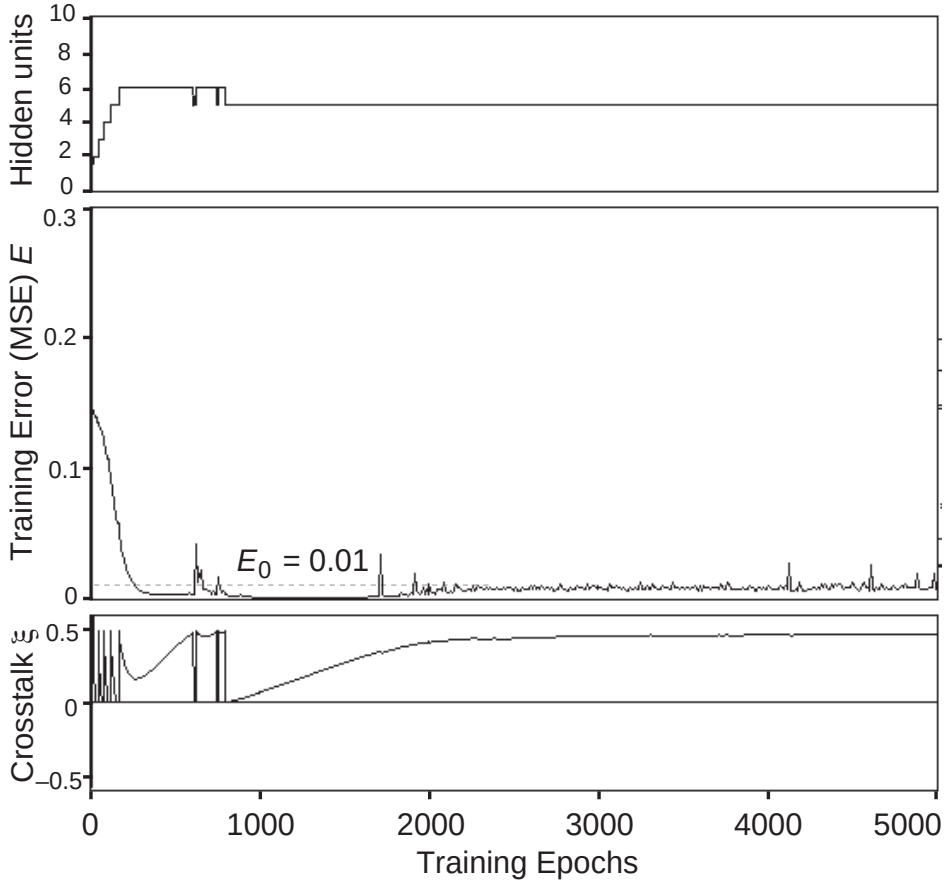


Fig. 3.19. The change of the number of hidden layer units, the error E and the crosstalk ξ during training the network for the alphabetical character recognition problem.

3.8.4 Discussion

In the above experiments, the stress parameter was defined so that the number of hidden layer units will converge to the minimal size in which the training could be accomplished. The hidden layer size to which the algorithm converges can be controlled in an indirect manner by setting a target maximum absolute similarity R_0, and using a Fitness Index definition of

$$I_F = 1 - |r_{ij}| - R_0 \ . \tag{3.81}$$

The target R_0 will control the degree of orthogonality of the hidden layer basis functions. When R_0 approaches 0, the required orthogonality will be stricter, resulting in a smaller hidden layer. When R_0 approaches 1, the restrictions will become milder, and will result in a larger hidden layer.

Networks of various hidden layer size were generated in the alphabetical character recognition using the layer size adjustment algorithm with the MS Index defined in Eq. (3.81). The networks were evaluated by test sets with random 2 positions among the 49 inputs being reversed as the added noise. The average rates of correct recognition for 10 trials each are plotted for the networks with model switching based on the stress of Eq. (3.81), controlling the maximum absolute similarity R and those without any model switching. In the results, it is notable that the recognition rates for the networks made by the proposed algorithm give higher classification rates, especially when the model size is small. In view of the discussion of model selection and generalization ability in Sec. 3.7.2, the results can be explained. When linear independence among the basis functions of the hidden layer units are forced by the proposed model switching scheme, the dimension of the function space spanned by the hidden layer S'_{NBP} will always be larger than those obtained by BP training without any restrictions. Thus the search space for the approximation function will be larger, resulting in a better approximation of the true function f.

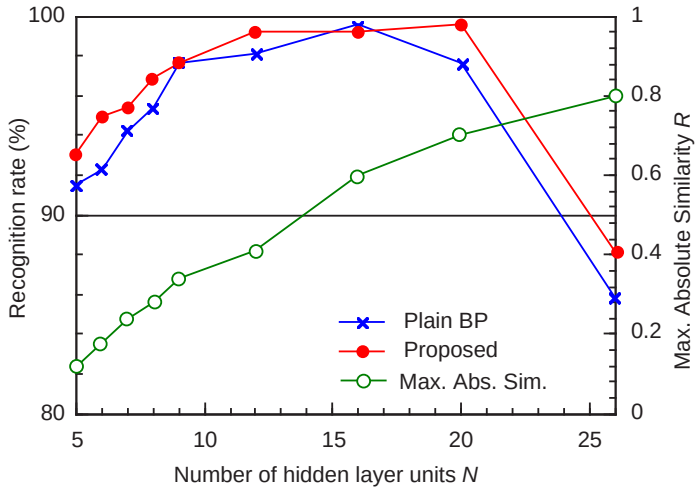


Fig. 3.20. Comparison of the classification rates of the noisy inputs for the network trained with plain BP and those trained with the proposed algorithm for model switching during BP training. The maximum absolute similarity of the hidden layer of the networks trained by the proposed algorithm is shown as well. All results are averages of 10 trials.

3.9 Summary

In this chapter, MS was implemented in the BP training of MLP networks. As the Fitness Index for selecting the initial maps, degree of map inheritance was evaluated. First, a method of measuring the degree of map inheritance, named *map distance* was defined. Further, the Fitness Index using the *map distance* measure was defined for use in the network training with MS. Then, newly proposed methods for MA, named *unit fusion*, *splitting* and *installation* were introduced. In order to simplify the map distance evaluation process, an approximate method for estimating the map distance caused by the unit fusion was introduced using the theorems giving the upper bounds of the map distances. The proposed methods for MA was applied to network pruning. The unit fusion method was compared with several well known pruning methods. In the experiments, the superiority of the unit fusion method in map preservation was shown. Then, the proposed MS according to MS Index preferring map preserving candidates was applied to the BP training. There, it was shown that the whole training process of obtaining a small trained network, will be much efficient when MS by MS Index was used, in comparison with the conventional MA strategies. Further, the relation between the generalization ability of the network and the selection of the initial map in the new model using the proposed MA methods and map distance, was discussed. There, it was shown that altering the model by unit fusion has a positive effect to improving the generalization ability of the network. As a variant of the proposed training method by MS, a method which allows a smooth change in the model by controlling the linear independence of the responses of the hidden layer units was introduced. There, it was shown that the training process will smoothly converge to a small network attaining the target training error.

Chapter 4
Model switching in Radial Basis Function networks and hybrid networks

4.1 Introduction

4.1.1 Pattern recognition using locally tuned kernel functions

When the model of the *a posteriori* probability distribution function (PDF) of each class is unknown, but can be assumed to be a smooth and continuous function, firm class information resides only at the training set element points and their close neighborhood. When a new input is distant from either of the training inputs, it is desirable if it points out that the certainty to the class estimation for the input is low, or simply that the answer is unknown. Use of certainty factors or memberships based on the distances from prototypes are common in order to implement this kind of feature.

Among the neural network models employing nonlinear discriminant functions, those with locally tuned convex kernels, such as the RBF net [20][25] are popularly used for clustering [21] and approximation of class probability density functions (PDF) for further classification [22].

When the membership function of a class of data is to be estimated using a set of training sets, especially for the sake of classification, a popular method will be to define the membership function as an estimate of the PDF for use in the Bayes decision making [3]. On estimating the PDF, methods such as the Parzen's method [1][3][4] are available, which defines the estimation to be the sum of a set of locally tuned kernel functions, each center positioned at a training input belonging to the class. The Probabilistic Neural Network (PNN) model introduced by Specht [22] is a direct implementation of this process in a neural network style. PNN relies on the Parzen estimation of PDFs [1][3] as the sums of identical kernel functions positioned at each training input. Although this is a very generic approach, the computational cost can be a concern, since the cost of calculation of the PDF will grow in proportion to the number of the class training input. For the reduction of the computational cost, it would be convenient if the whole PDF can be approximated by another (larger) nonlinear discriminant function.

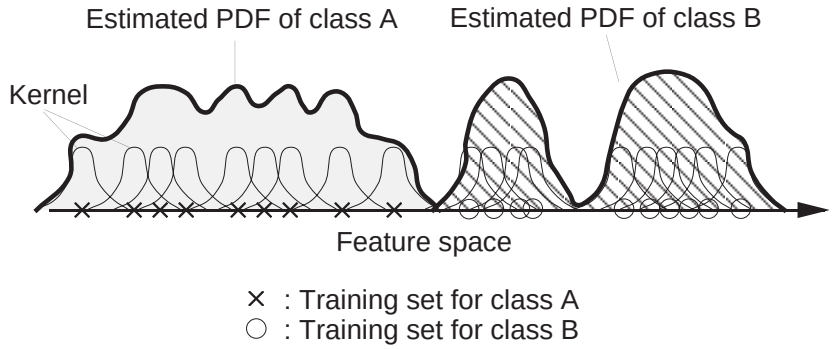


Fig. 4.1. The Parzen's estimation of the probability density functions, defined as the sums of the kernel functions localized at the training inputs.

The same task of building the classifier can be viewed as finding an approximation function $f(\boldsymbol{x})$ from a finite set of sampled data, namely to solve the inverse problem to reconstruct the original function $f_0(\boldsymbol{x})$, as discussed by Ogawa[54][57] and reviewed in Sec. 3.7.1. A trivial way of satisfying the requirements for constructing a projection generalizing neural network would be to use a set of M hidden units each of which basis function is a point spread (kernel) function positioned at each training input.

A similar result has been reported by Poggio and Girosi in [20], on obtaining an approximation function which minimizes the cost function defined as the sum of the training error and the regularization penalty term. The derived solution is a linear weighted sum of M Green's function [6] of the self-adjoint differential operator of the penalty term.

The strict derivations reviewed above all require that the number of hidden layer units are equal or greater than the number of training sets. However, this is an unrealistically severe condition when implementing the network in a computer, especially for a large-scale problem.

4.1.2 MS in obtaining a reduced model with suitable class borders

One compromise, especially possible in classification problems where the approximation functions are indicator functions, would be to use less number of kernel functions (hidden units), each representing a subset of training data. An extreme case would be when the class PDF is known to be a well defined function such as the Gaussian distribution, where each class can be represented by a single kernel function. Use of reduced kernels have been tried in the context of PDF estimation named the *Reduced Parzen Classifier* by Fukunaga et. al [3], and function approximation by the *Generalized Radial Basis Function Networks* by Poggio and Girosi [20], both using Gaussian kernel functions.

In obtaining a trained network of the reduced model, MS may be applied for gradually reducing the model, when a scalable kernel such as Gaussian function is used. In this chapter, BP training by MS will be used for reducing the model size and training the network simultaneously, in

an efficient manner. For the efficiency, map inheritance is important as was the case in Chapter 3. Therefore, the degree of map inheritance will be used as the Fitness Index among the switchable candidates.

In addition to the map inheritace attribute, selection of the switchable candidates by the nature of the class borders will be tried. On applying MS to the RBF-type networks, further measures for selecting the switchable model, in favor of higher ability for recognizing the unfamiliar inputs, will be taken. In the latter half of this chapter, Fitness Index for selecting the order of the class borders will be introduced.

In this chapter, MS in RBF networks and hybrid networks will be discussed. In Sec. 4.2, A network consisting of RBF type hidden units, named Hyperellipsoid Clustering Network (HCN) will be introduced. The applicability of MS to HCNs for efficient training, realization in a small model, and improvement of the rejection ability of unfamiliar inputs, will be sought. By using the training method, it will be shown that above goals will be attained. In Sec. 4.3, applicability of MS to networks with mixed configurations of first and second order units in its hidden layer will be tried. There, an additional operation for switching from a second order unit to a set of first order unit will be introduced. By combining the method with unit fusion and installation methods introduced in the former chapter, a method for BP training with MS for the “hybrid” network, in the style of Fig. 2.3 will be introduced. The algorithm will be applied to a classification problem requiring class borders of different natures, and it will be shown that the generalization ability can be improved as an efficient characterization of the classification borders will be possible.

4.2 MS in Hyperellipsoid clustering networks

4.2.1 Hyperellipsoid clustering network (HCN)

The network introduced here belongs to the higher-order network models that allow the potential u to be n-th order ($n \geq 2$) polynomial discriminants in the input feature space. Since the decision border is nonlinear, a more flexible segmentation of the feature space is possible, with the cost of increased number of parameters. The idea of the higher-order threshold logic unit (HOTLU) already appears in the early days of research on neural networks [2], but they were not used for real world applications due to the *curse of dimensionality* (explosion of the number of terms) [4]. Supervised training rules of the HOTLU appeared in [16]. There, it is addressed that by analyzing the problem domain and selecting the HOTLU unit with appropriate terms of a modest number, the network can learn and generalize very efficiently. The general supervised training rule of multilayer HOTLUs in the backpropagation style appeared in [5] and [17]. The training rule employed in this work conforms to the general backpropagation rule, with some additional restrictions.

In the network model used here, the hidden layer unit order is limited to two. Furthermore, the potential of the n-th unit in the hidden layer is constrained to be formally of

$$u_n = r_n^2 - \|\boldsymbol{H}_n (\boldsymbol{x} - \boldsymbol{\mu}_n)\|_E^2 = r_n^2 - (\boldsymbol{x} - \boldsymbol{\mu}_n)^T \boldsymbol{H}_n^T \boldsymbol{H}_n (\boldsymbol{x} - \boldsymbol{\mu}_n) , \tag{4.1}$$

using the following notations :

$r_n \in R$: The radius parameter.

$\boldsymbol{\mu}_n = (\mu_{n1} \ldots \mu_{nI})^T \in R^N$: The center vector.

$\boldsymbol{H}_n = [H_{nst}] \in R^N \times R^N$: The weight matrix.

The transfer function of the hidden layer unit is the same sigmoid function as in Eq. (1.4). Thus, the output of unit n is,

$$h_n = s(u_n) \qquad (n = 1,2, \ \ldots, \ N) . \tag{4.2}$$

This unit will be referred to as the *second order sigmoid unit*, or simply as the O_2 unit. The hidden units in MLPs whose unit potentials are defined as linear functions will be called the *first order sigmoid unit*, or simply as the O_1 unit.

A unit in the output layer of the HCN takes the fan out of the hidden layer units and calculates the weighted sum with no bias as,

$$o_k = \boldsymbol{w}_k^T \boldsymbol{h} \qquad (k = 1,2, \ \ldots, \ O), \tag{4.3}$$

where $\boldsymbol{w}_k = (w_{k1} \ldots w_{kJ})^T \in R^O$ is the weight vector of the k-th output unit, and $\boldsymbol{h} = (h_1 \ldots h_N)^T \in R^N$.

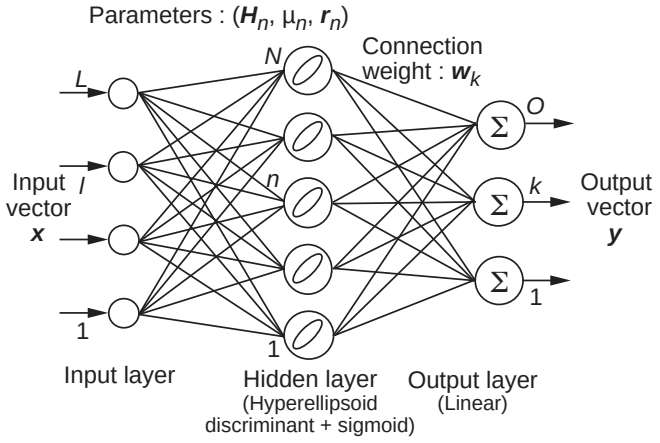


Fig. 4.2. The hyperellipsoid clustering network.

By restricting the quadratic term of the discriminant to be positive semidefinite as in Eq. (4.1), the discrimination border will always be a hyperellipsoid. Same restrictions and neighborhood definitions apply to the kernels of the Gaussian Radial Basis Function networks [20][25], and therefore the model can be considered to be a member of RBF networks. With use of the radius parameter and the sigmoid function, however, there is an additional freedom in the selection of the membership function. In other terms, the relative gain [23] of the sigmoid function can be controlled. Examples of the membership functions of the Gaussian function and the joint use of hyperellipsoid discriminant and the sigmoid function are compared in Fig. 4.3. There, the additional freedom in the

profile of the membership function is obvious. This network model using the hyperellipsoid discriminant and the sigmoid function in the hidden layer, will be referred to as the *Hyperellipsoid Clustering Network* (HCN).

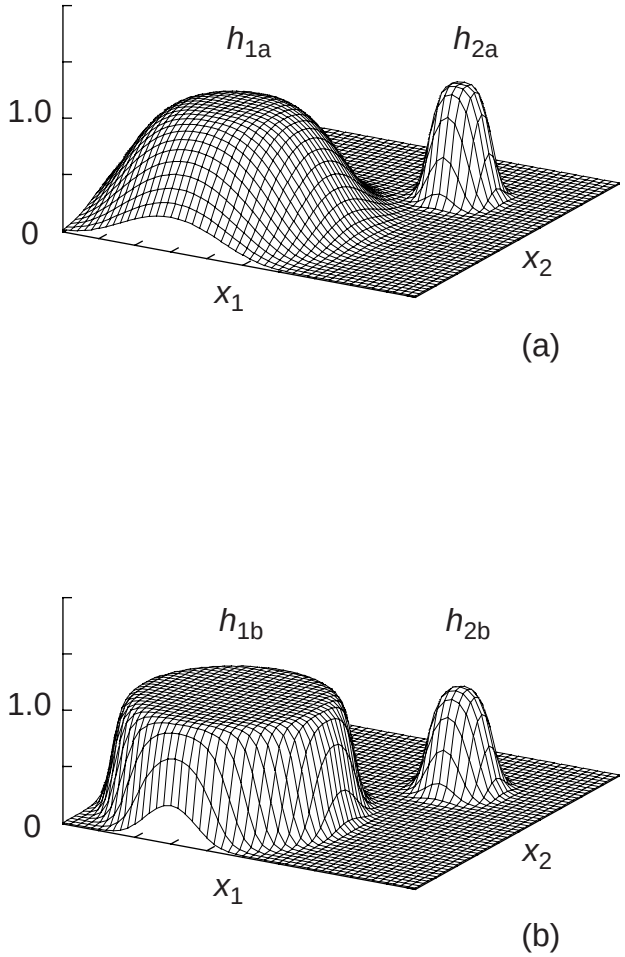


Fig. 4.3. A comparison of the membership functions made by, (a) bivariate Gaussian functions and (b) joint use of (hyper) ellipsoid discriminants and sigmoid functions. The membership function pairs (h_{1a} h_{1b}) and (h_{2a} h_{2b}) both share the center vectors. It is seen that membership functions of various profile are enabled in (b) by the added freedom of the radius parameter.

The training method used in the HCN is based on the batched backpropagation law with momentum terms. In the following formulas, however, the momentum terms will be omitted for simplicity. At each training cycle, an arbitrary parameter *s* will be modified as,

$$s(\tau + 1) = s(\tau) - \Delta s(\tau) = s(\tau) - \frac{1}{M}\sum_{m=1}^{M} \Delta s(\tau)_m = s(\tau) - \frac{1}{M}\sum_{m=1}^{M} \eta_s \frac{\partial E_m}{\partial s(\tau)}, \tag{4.4}$$

with τ and η_s denoting the time and the training gain, respectively. Modification measures for each parameters in the style of Δs_m (time omitted) in Eq. (14) can be written as,

$$\Delta \boldsymbol{w}_{km} = \eta_w (o_{km} - t_{km}) \boldsymbol{h}_m , \tag{4.5}$$

$$\Delta r_{nm} = \eta_r \delta_{nm} (2r_n) , \tag{4.6}$$

$$\Delta \boldsymbol{H}_{nm} = \eta_H \delta_{nm} \{-2\boldsymbol{H}_n (\boldsymbol{x} - \boldsymbol{\mu}_n)(\boldsymbol{x} - \boldsymbol{\mu}_n)^T\} \tag{4.7}$$

and

$$\Delta \boldsymbol{\mu}_{nm} = \eta_\mu \delta_{nm} \{2\boldsymbol{H}_n^T \boldsymbol{H}_n (\boldsymbol{x} - \boldsymbol{\mu}_n)\}. \tag{4.8}$$

In Eqs. (16)-(18), δ_{nm} denotes the *backpropagated error* [5] of

$$\delta_{nm} = h_{nm}(1 - h_{nm}) \sum_{k=1}^{O} w_{kn}(o_{km} - y_{km}) . \tag{4.9}$$

An additional cost term was used for tuning the weight matrix $\boldsymbol{H}_n$ and the radius parameter r_n. For enabling a “tight bounding by hyperellipsoids” to appropriately implement the recognition of the unfamiliar inputs, the “volume” of the hyperellipsoids should be kept small as long as it does not harm the achievement of training. This can be done by setting some penalty term to restrict the radius. The distance from the center to the edge of the hyperellipsoid in the direction of the *i*-th principal component can be written as $|r_n| / \sqrt{\lambda_{ni}}$, where λ_{ni} is the *i*-th eigenvalue of the matrix $\boldsymbol{H}_n^T \boldsymbol{H}_n$, which is always positive. Thus, a penalty to suppress the absolute value of the radius parameter r_n can be considered to be effective. Since the response of the network can be quite unstable when this penalty is excessive, another term to prevent the eigenvalues from becoming too small, was necessary. This second restriction could be implemented indirectly by preventing the Euclidean norm of the matrix $\boldsymbol{H}_n$ from becoming too small. Consequently, the modifications to the weight matrix $\boldsymbol{H}_n$ and the radius parameter r_n were formulated as,

$$\boldsymbol{H}_n(\tau + 1) = \boldsymbol{H}_n(\tau) - \frac{1}{M}\sum_{m=1}^{M} \Delta \boldsymbol{H}_{nm}(\tau) + \phi_H \frac{\partial |\boldsymbol{H}_n(\tau)|_E^2}{\partial \boldsymbol{H}_n(\tau)} \tag{4.10}$$

and

$$r_n(\tau + 1) = r_n(\tau) - \frac{1}{M}\sum_{m=1}^{M} \Delta r_{nm}(\tau) - \phi_r \frac{\partial (r_n(\tau))^2}{\partial r_n(\tau)} , \tag{4.11}$$

with ϕ_H and ϕ_r being the penalty term gains.

Before starting the training, the center vectors $\boldsymbol{\mu}_n$ (n = 1,2, ..., N) of the hidden layer will be initialized to enable an efficient division of the feature space where the training information exist. The initial center vectors will be determined by applying the *k*-means clustering [4] to the training set inputs as in the Generalized RBF networks of [19]. After the convergence of the clustering algorithm, the prototype vectors will be used as the initial center vectors.

The training output encoding and the network output interpretation will be the same as in the

case of MLPs. Since the output is the weighted sum of the cluster membership function of the hidden layer units, it can be justified to reject an output vector that does not have a significant winner. These outputs will be interpreted as a response against an input of an unknown class. This feature to point out the unfamiliar input will be referred to as *rejection of unfamiliar inputs*, or simply *rejection*. When inputs that are unfamiliar to the network is to be rejected, a membership threshold θ_m will be used for class interpretation of the output vector as,

$$\text{class} = \begin{cases} \arg\max_k (o_k), \ (k = 1,2, \ ..., \ O) & \text{if } o_k > \theta_m \\ \text{unknown}, & \text{otherwise.} \end{cases} \tag{4.12}$$

4.2.2 Fitness index for map preservation and rejection of unfamiliar inputs

Here, the scheme of MS will be applied to HCN. The main objective to use MS in HCN training is to obtain a small network with high recognition rate and adequate ability of rejecting unfamiliar inputs in an efficient way.

On altering the model, unit fusion will be used. On fusing units, map preservation is again, very important. Therefore, the Fitness Index will be based on the map distance between the immediate network function and the candidates.

Furthermore, simple fusion of kernel functions can lead to a network with low rejection ability without additional caution besides map preservation. In Fig. 4.4 a sample of such a case is shown. When there are several candidates with similar map distances, unit pairs that are tuned to the closer regions will result in a higher rejection ability.

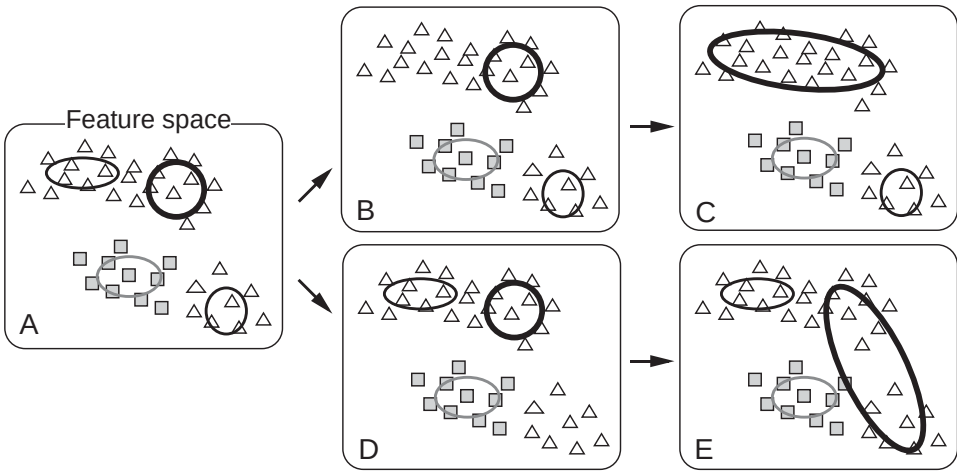


Fig. 4.4. Two switching candidates B and D may have same map distances. However, from the point of rejection ability of unknown inputs, it is preferred that B is selected, and further BP training will make C.

In order to give priority to fusion of kernels that are placed close together, the Fitness Index used for the MS in HCN were defined as follows. Note that this priority does not apply in case of model expansion by unit installtion.

Positive mode :

Candidates by unit installation :

$$I_F(f_N, f'_N) = \begin{cases} \{D_{max} - D(f_N, f'_N)\} / D_{max}, & \text{if } \{D(f_N, f'_N) < D_{max}\} \\ 1 & \text{otherwise} \end{cases} \tag{4.13}$$

Negative mode:

Candidates by unit fusion ($r_{ij} \geq 0$) :

$$I_F(f_N, f'_{Nij}) = \begin{cases} \dfrac{1}{L^2 D_{max}} \{L^2 - |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|_E^2)\}\{D_{max} - D(f_N, f'_{Nij})\}, & \text{if } \{D(f_N, f'_{Nij}) < D_{max}\} \\ \dfrac{1}{L^2} \{L^2 - |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|_E^2)\} & \text{otherwise} \end{cases} \tag{4.14}$$

Here, f'_{Nij} denote the candidate function made by fusing the i-th and j-th hidden unit, and L is the input dimension to the network. When compared with the original definition in Eq. (3.2), the modifier regarding the distances between the centers of the kernel functions are added.

The features of the HCN network trained by BP with MS is summarized in Table 4.1.

Table 4.1. Features of the HCN network trained with MS.

Objective of MS	1. Efficient training 2. Small model realization 3. Unknown class rejection
General • Network type	HCN
• MA scheme	MS consulting MS Index. Add/remove hidden layer units.
MS • Scheduling	Unlimited model
• Fitness Evaluation	Map preservation + rejection class border heuristics
• Fitness Index	$I_F = \frac{1}{L^2 D_{max}} \{L^2 - \|\|\mu_i - \mu_j\|\|_E^2)\}(D_{max} - D)$
• Stress Index	$I_S(E, \frac{\partial E}{\partial t}) = \text{sgn}(E - E_0)\,\text{sgn}(\frac{\partial E}{\partial t} + I_0)$
• MS Index	$I_{MS} = I_F I_S$
• MS candidate $(I_{MS} \geq 0)$	Unique map by unit installation
$(I_{MS} < 0)$	All possible by unit fusion

4.2.3 Experiment

The effect of using model switching during the training of the HCN will be shown using an artificial four-class data defined in a two-dimensional domain shown in Fig. 4.5. This problem projects the nature of the data in commonly found in the many classification problems, such as : (1) Basically the data clusters are localized, however, there are some overlaps among the clusters. (2) The data resides only at a portion of the feature space; there are areas where no data.

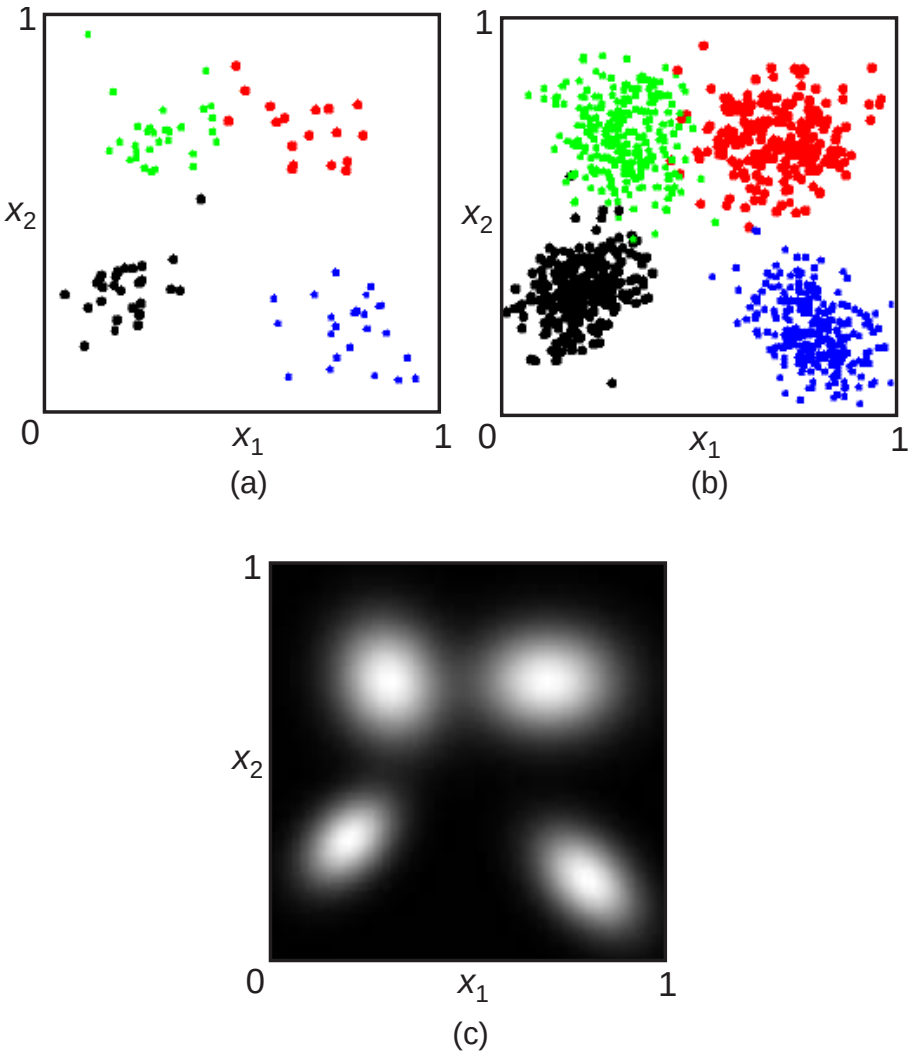


Fig. 4.5. An artificially generated cluster data of four classes. (a) Training set (M = 100). (b) Test set (M= 1000). (c) Generating Gaussian PDF.

Three types of networks and training strategies were tested.

Common conditions

Training input :	100 2-D input of 4 class.
Output encoding :	Unit vector output
Target error E_0 :	0.01

Network 1 (HCN)

Units (input-hidden-output)	(2-100-4) .
Training gains	η_μ= 0.01, η_H = 0.5, η_r = 0.01, η_w = 0.1.
Momentum term gains	α_μ=α_H = α_r = 0.9 .
Penalty term gains	$\phi_H = 5.0\times10^{-6}$, $\phi_r = 1.0\times10^{-6}$.

Initial parameters are randomly selected from the following sections.

w : (–0.1, 0.1) .
h : (diagonal element) (6.0, 7.0) * .
h : (non-diagonal element) (–0.1, 0.1) * .
r : (0.5, 1.0) .
μ : tuned to each training input .

* Initial weight matrix $\boldsymbol{H}$ is set to start the training from a near-hypersphere status.

MA Not applicable

Network 2 (Proposed : HCN with MS by MSIndex)

Initial model and training parameters are identical with those of Network 2.

See. Table 4.1 for conditions regarding MS.

Misc. settings : $I_0 = 0.0,\ I_{MS}^{+} = 0.5,\ I_{MS}^{-} = -0.95$

Network 3 (MLP)

Units (input-hidden-output) (2-4-4) .
Training gain $\eta = 5.0$.
Momentum gain $\alpha = 0.9$.
Initial weights Selected randomly from (–0.1, 0.1) .
MA Not applicable

A. Computional cost for training

In Table 4.2, the relative amount of computation required for training the networks measured by pure computational time, are shown. By using MS, the cost is reduced due to fast reduction of hidden layer units. In case of the proposed network 3, the majority of the trials switched the network to a (2-4-4) model, which requires only 1/25 th of computation for using. The use of heuristics for

Table 4.2. Comparison of the computation required for training. Numbers are averaged for 10 trials.

Network	Relative computational time
1. HCN	1.00
2. HCN with MS	0.38
3. MLP	0.30

B. Membership threshold and the rejected area

Next, the change in the recognition rate for the test set, and the ratio of the area within the input domain was evaluated by changing the membership threshold θ_m in Eq. (4.12). Ideally, the recognition rate will be maintained high, even when a large portion of the input domain is judged as "unknown". The result is shown in Fig. 4.6. It is seen that proposed Network 3 best identifies the

unknown input without losing the classification ability for the test set. In Fig. 4.7, the rejected region are compared for the MLP and the HCN with MS. It is clear that the latter reproduces the nature of the original PDF in Fig. 4.5(c) better.

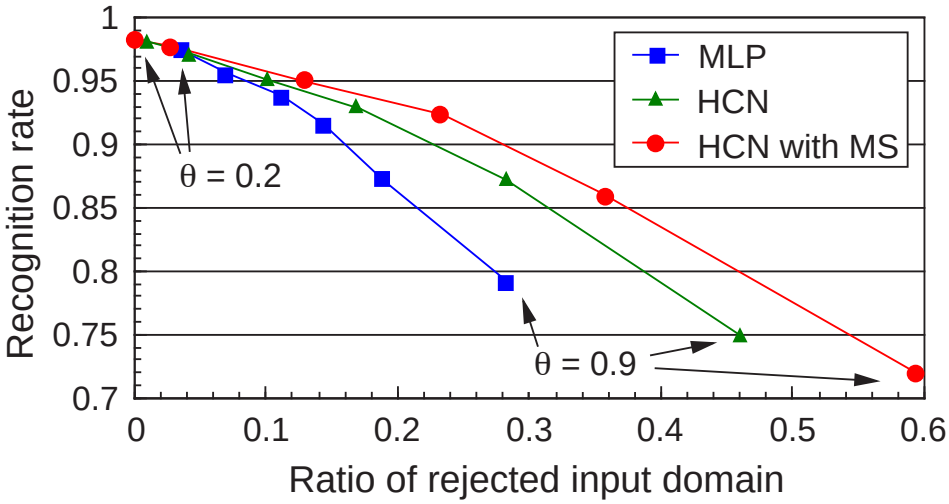


Fig. 4.6. The change in the recognition rate and the ratio of the rejected input domain, when the membership threshold θ_m is changed.

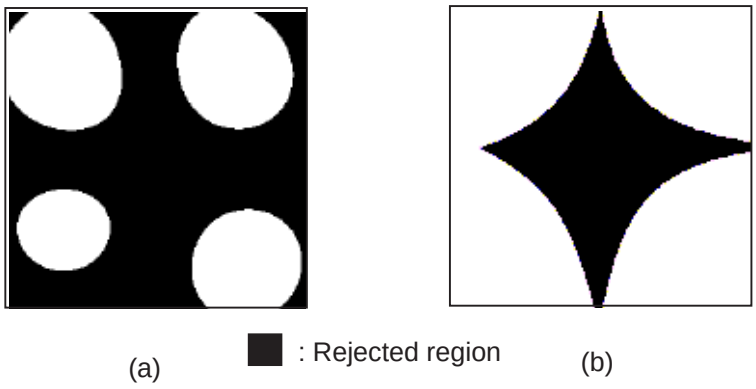


Fig. 4.7. The rejected region for the 2D classification problem, generated by the PDF in Fig. 4.5(c). (a) HCN with MS. (b) MLP. Membership threshold : $\theta_m = 0.9$.

In view of the results of the two experiments, it can be concluded that the HCN with MS allows to construct a classifier of a small size in a very efficient manner, having superior ability of recognizing the unfamiliar inputs. In Chapter 6, this network will be used as the classifer for the defect classification system of semiconductors.

4.3 Model switching in hybrid layered networks

The first order sigmoid units (O_1 units) used in MLPs and the second order sigmoid units (O_2 units) used in HCNs both have their merits and demerits. Also, their features are often complementary. When the distribution of the classes can be characterized to be in an open region of the feature space, linear discriminants perform better. The open region may be simulated by a O_2 unit, however, the model size can be unnecessarily larger. When the class input is distributed in a small local region as seen in many real-world data, localized discriminants are more suitable for representing the class. For dealing with lots of such clusters, use of linear discriminants can become very costly with low generalization ability. The merit of employing both types of units have been discussed in [25], where a tree search algorithm with pruning for a suitable model has been presented. In this section, a layered network architecture having both types of hidden layer units called the *hybrid network* will be introduced. Then a novel model switching path from one O_2 unit to multiple O_1 units will be introduced. The Fitness Index will be encoded so that the map preservation measure of selecting a model with minimum map distance will be overridden by switching from O_2 unit to O_1 units, where applicable. In the experiment, the hybrid network with MS will be applied to the classification of an artificial data requiring both linear and nonlinear class borders. For comparison, MLPs and HCNs trained under a static model will be tried as well.

4.3.1 The hybrid network

The architecture of the hybrid network is shown in Fig. 4.8. The network consists of L input, N hidden and O output units. The input layer with a bias channel holds the input $\boldsymbol{x} = (x_1 \ldots x_L)^T \in R^L$ to be fed to the hidden layer.

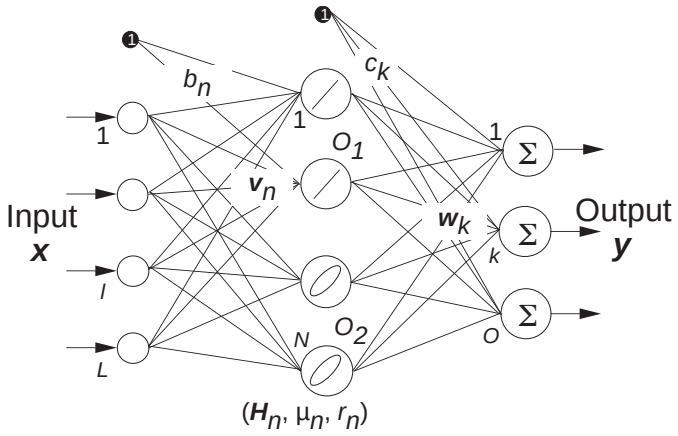


Fig. 4.8. The hybrid network.

The hidden layer has both O_1 units and O_2 units. The response of the n-th hidden unit will be,

$$h_n = s(u_n) = \{1 + \exp(-u_n)\}^{-1} \qquad (n = 1,2, \ ..., \ N). \tag{4.15}$$

For the O_1 units, the unit potential will be defined as,

$$u_n = \boldsymbol{v}_n^T \boldsymbol{x} + b_n \qquad (n = 1,2, \ ..., \ N_1), \tag{4.16}$$

with $\boldsymbol{v}_n = (v_{n1} \ ... \ v_{nL})^T \in R^L$, $b_n \in R$ and N_1 being the weight vector, the bias term and the number of O_1 units, respectively. In the training process, parameter sets $\{(\boldsymbol{v}_n, b_n)\}_{n=1}^{N_1}$ will be tuned by BP. For the O_2 units, the unit potential will be defined as,

$$u_n = r_n^2 - \|\boldsymbol{H}_n (\boldsymbol{x} - \boldsymbol{\mu}_n)\|_E^2 \qquad (n = N_1+1, \ ..., \ N_1 + N_2), \tag{4.17}$$

using the notations defined in the former section. On training, parameter sets $\{(\boldsymbol{H}_n, r_n, \boldsymbol{\mu}_n)\}_{n=N_1+1}^{N_1+N_2}$ will be altered by BP, with N_2 being the number of O_2 units. Naturally,

$$N_1 + N_2 = N$$

holds.

The output layer unit takes the fan out of the hidden layer units and calculates the weighted sum with a bias as their outputs as,

$$o_k = \boldsymbol{w}_k^T \boldsymbol{h} + c_k \qquad (k = 1,2, \ ... \ , \ O), \tag{4.18}$$

where $\boldsymbol{w}_k = (w_{k1} \ ... \ w_{kN})^T \in R^N$ and $c_k \in R$ denote the weight vector and the bias of the k-th output unit, and $\boldsymbol{h} = (h_1 \ ... \ h_N)^T \in R^N$. The weight vector $\boldsymbol{w}_k$ and the bias c_k will be modified by BP training.

4.3.2 Model switching from O_2 unit to O_1 unit

The motive for using both types of hidden units in the hybrid network, is to assign the units to classification borders which each unit type is suited for. Therefore, an alternative switching methods between the two unit types will be necessary as well. A method to construct a multivariate Gaussian function by two layers of first order sigmoid units are introduced in [27]. However, in order to maintain the number of layers in the network, a similar type of replacement of O_2 unit by combinations of O_1 units within the same hybrid layer will be employed.

When an O_2unit is selected to be switched to O_1 units, 2^L O_1 units will be prepared and tuned to make a polyhedron region to simulate the original region which the O_2unit was tuned to. This operation will be done as in the following.

First, using a weight matrix of an O_2 matrix, let $\boldsymbol{H}_n^T \boldsymbol{H}_n$ be diagonalized as,

$$\boldsymbol{H}_n^T \boldsymbol{H}_n = \boldsymbol{Q}_n \boldsymbol{\Lambda}_n \boldsymbol{Q}_n^{-1}, \tag{4.19}$$

with $\boldsymbol{\Lambda}_n$ consisting of eigenvalues $\{\lambda_{nl}\}_{l=1}^{L}$ in its diagonal elements, and $\boldsymbol{Q}_n$ consisting of eigenvectors $\{\boldsymbol{e}_{nl}\}_{l=1}^{L}$ in its columns. The vertices $\{\boldsymbol{V}_{ni}\}_{i=1}^{2L}$ of the polyhedron will placed where the principal axes meet the hyperellipsoid surface. Thus,

$$\{\boldsymbol{V}_{ni}\}_{i=1}^{2L} = \{\{\boldsymbol{\mu}_n + t|r_n|\lambda_{nl}^{-1/2} \boldsymbol{e}_{nl}\}_{l=1}^{L}\}_{t=-1, 1} . \tag{4.20}$$

By defining each hyperplane of the polyhedron to include L neighboring vertices, 2^L hyperplanes will be made. See Fig. 4.9 for the case of $L = 2$.

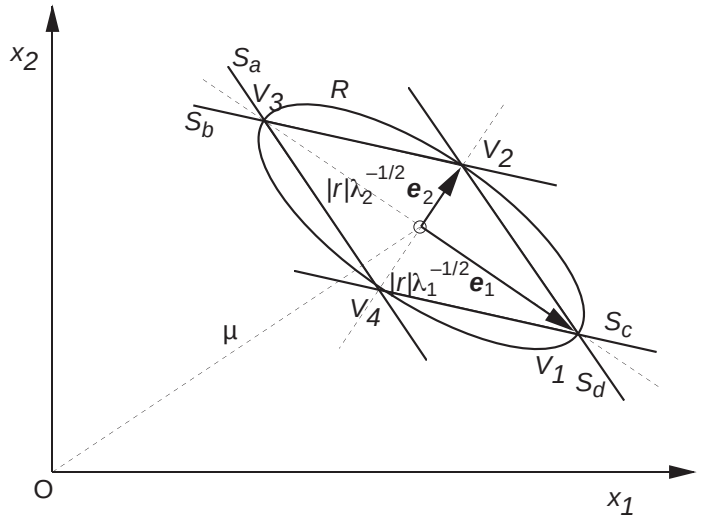


Fig. 4.9. Model switching by replacement of an O_2 unit R by 2^L O_1 units (S_a, S_b, S_c, S_d), for $L = 2$, preserving the characterized region. Vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are unit vectors in the first and second principal components of the ellipse. The index for the hidden neuron n is omitted.

Connection weights connected to the O_1 unit n_1 which was switched from the O_2 unit n can be determined as follows.

Input-Hidden

Since the O_1 unit's discrimination hyperplane includes L vertices $\{\boldsymbol{V}_i\}_{i=1}^{L}$, the weight vector $\hat{\boldsymbol{v}}_{n_1}$ for unit bias $b_{n_1} = 1$ can be obtained by solving a linear equation ,

$$\boldsymbol{V}_{n_1}\hat{\boldsymbol{v}}_{n_1} + \boldsymbol{I} = 0 , \tag{4.21}$$

for $\hat{\boldsymbol{v}}_{n_1} \in R^L$, where $\boldsymbol{V}_{n_1} = [\boldsymbol{V}_1 \cdots \boldsymbol{V}_L] \in R^L \times R^L$ and $\boldsymbol{I} = [1 \cdots 1]^T \in R^L$. The discrimination hyperplane will be,

$$\alpha_{n_1}(\hat{\boldsymbol{v}}_{n_1}^T \boldsymbol{x} + 1) = 0 . \tag{4.22}$$

Parameter $\alpha_{n_1} \in R$ will be determined so that the unit potential u_n at the center ($\boldsymbol{\mu}_n$) of the O_2 unit will be preserved. Therefore,

$$u_n(\boldsymbol{\mu}_n) = r_n^2 = \alpha_{n_1}(\hat{\boldsymbol{v}}_{n_1}^T \boldsymbol{x} + 1) = u_{n_1}(\boldsymbol{\mu}_n) \tag{4.23}$$

holds. From Eq. (4.23), the weights and the bias for the new O_1 unit will be determined as,

$$\boldsymbol{v}_{n_1} = \alpha\hat{\boldsymbol{v}}_{n_1} = r_n^2\hat{\boldsymbol{v}}_{n_1}/(\hat{\boldsymbol{v}}_{n_1}^T\boldsymbol{\mu}_n + 1) \tag{4.24}$$

and

$$b_{n_1} = \alpha = r_n^2/(\hat{\boldsymbol{v}}_{n_1}^T\boldsymbol{\mu}_n + 1) . \tag{4.25}$$

Hidden-Output

The weights and the biases will be determined to preserve the unit potential of an arbitrary output unit k.

When there is a input at $\boldsymbol{x} = \boldsymbol{\mu}_n$, the O_2 unit and the bias will make a potential of

$$u_{k1} = w_{kn} s(r_n^2) + c_k \,. \tag{4.26}$$

After switching, this will be,

$$u_{k2} = 2^L w'_{kn_1} s(\boldsymbol{v}_{n_1}^T \boldsymbol{\mu}_n + b_{n_1}) + c'_k = 2^L w'_{kn_1} s(r_n^2) + c'_k \,, \tag{4.27}$$

where the primes denote the weights and biases for the new O_1 unit to be determined.

For an input at a very distant point from $\boldsymbol{x} = \boldsymbol{\mu}_n$, the potential for the O_2 unit would be,

$$u_{k3} = c_k \,, \tag{4.28}$$

for the O_2 unit response will be near-zero. After switching, at least 2^{L-1} among the 2^L O_1 units will be responding in near-unity. Thus, we will write,

$$u_{k4} = 2^{L-1} w'_{kn_1} + c'_k \,. \tag{4.29}$$

By maintaining $u_{k1} = u_{k2}$ and $u_{k3} = u_{k4}$, it can be solved for w'_{kn_1} and c'_k as,

$$w'_{kn_1} = 2^{-(L-1)} w_{kn} s(r_n^2) \tag{4.30}$$

and

$$c'_k = c_k - w_{kn} s(r_n^2) \,, \tag{4.31}$$

respectively.

4.3.3 Fitness index for map preservation and selection of suitable class borders

Here, the MS will be applied to the proposed hybrid network. The main objective to use MS in training is to obtain a simple model that suits the problem for higher generalization ability.

Three ways for altering the model will be provided. First is the switch from a O_2 unit to several O_1 units as defined above. Second is the installation of a O_2 unit, and the third is the fusion of hidden layer units. Since efficient modelling of the class borders is the main issue here, the Fitness Index will be encoded so that the first switching path will have the highest priority, where applicable. Since the class borders of the O_2 units that tries to characterize linear borders tend to grow highly eccentric, the n-th O_2 unit with a highest eccentricity q_n exceeding a predetermined threshold q_0 will be selected. The eccentricity q_n will be defined as,

$$q_n = \frac{\max_l |r_n| \lambda_{nl}^{-1/2}}{\sum_{l=1}^{L} |r_n| \lambda_{nl}^{-1/2}} \,. \tag{4.32}$$

The definition of the Fitness Index used in the MS of hybrid networks is as follows.

Candidates by O_2 unit to O_1 unit switching :

$$I_F(f_N, f'_N) = aU(q - q_0) \tag{4.33}$$

Candidates by unit installation and fusion:

$$I_F(f_N, f'_{Nij}) = \begin{cases} \{1 - D(f_N, f'_{Nij})\} / D_{max}, & if\ \{D(f_N, f'_{Nij}) < D_{max}\} \\ 1 & otherwise \end{cases} \tag{4.34}$$

In Eq. (4.33), $U(x)$ is an indicator function

$$U(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases} \tag{4.35}$$

and a is a factor satisfying $a > 1$ to ensure the priority of O_2 unit to O_1 unit switching .

The features of the hybrid network trained by BP with MS is summarized in Table 4.3.

Table 4.3. Features of the hybrid network trained with MS.

Objective of MS	1. Small model realization 2. Improvement of generalization ability
General • Network type	Hybrid net with O_1 and O_2 sigmoid hidden layer units
• MA scheme	MS consulting MS Index. Add/remove hidden layer units.
MS • Scheduling	Unlimited model
• Fitness Evaluation	Map preservation + heuristic class border selection
• Fitness Index	$O_2 \rightarrow O_1$ switching : $I_F(f_N, f'_N) = aU(q - q_0)$ Installation and fusion : $I_F(f_N, f'_{Nij}) =$ $\{1 - D(f_N, f'_{Nij})\} / D_{max}$
• Stress Index	$I_S(E, \frac{\partial E}{\partial t}) = \text{sgn}(E - E_0)\ \text{sgn}(\frac{\partial E}{\partial t} + I_0 E)$
• MS Index	$I_{MS} = I_F I_S$
• MS candidate $(I_{MS} \geq 0)$	$O_2 \rightarrow O_1$ switching Unique map by unit installation
$(I_{MS} < 0)$	$O_2 \rightarrow O_1$ switching All possible by unit fusion

4.3.4 Experiment

An artificial three-class training set in a two dimensional feature space requiring both linear and second order classification borders was used to evaluate the training algorithm of the hybrid network. The distribution of the training set is shown in Fig. 4.10. Class 1 and class 2 data were generated according to a 2D Gaussian distribution and a distribution made by a combination of two 2D step functions, respectively. The third class was generated uniformly where the probability distribution of the Classes 1 and 2 were both below 0.005. The size of the training set and the test set were 300 and 1000, respectively, both generated using the same distribution functions.

Conditions

Training input :	300 2-D input of 3 class.
Output encoding :	Unit vector output
Target error E_0 :	0.02

Network (Hybrid network trained by BP with MS)

Initial model (input-[hidden O_2, O_1]-output)
(2-[1, 0]-4) .

O_2 unit :

Training gains	η_μ= 0.01, η_H = 0.5, η_r = 0.01, η_w = 0.1.
Momentum term gains	α_μ=α_H = α_r = 0.9 .
Penalty term gains	$\phi_H = 5.0\times10^{-6}$, $\phi_r = 1.0\times10^{-6}$.

Initial parameters are randomly selected from the following sections.

w : (–0.1, 0.1) .
h : (diagonal element) (6.0, 7.0) .
h : (non-diagonal element) (–0.1, 0.1) .
r : (0.5, 1.0) .
μ : tuned to each training input .

O_1 unit :

Training gain	η = 5.0 .
Momentum gain	α = 0.9 .
Initial weights	Selected randomly from (–0.1, 0.1) .

MS
See. Table 4.1 for conditions regarding MS.

Misc. settings :	I_0 = 0.002, I_{MS}^{+} = 0.5, I_{MS}^{-} = –0.95

A. Training progress

An example progress of training by the training algorithm is shown in Fig. 4.11. After installation of two O_2 units, clusters of classes 1 and 2 were both characterized by one O_2 unit each, and the error was quickly reduced. After a more precise extraction of the class borders, a O_2 unit which characterized Class 2-Class 3 border was switched to four O_1 units. Further, the switched O_1 units were reduced by fusion to two. Due to the flow of the training algorithm in Fig. 2.3, the training will not be terminated until there exists no more switchable models. It should be also noted that no significant increase in the error is observed after each switch in the model. This fact shows the effectiveness of the map preservation in MS.

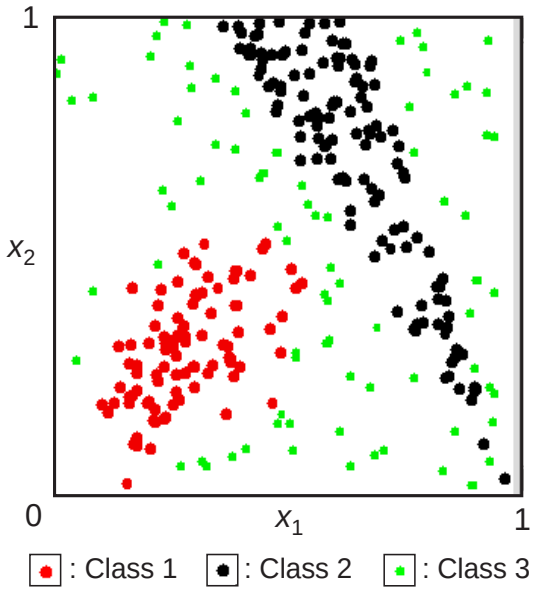
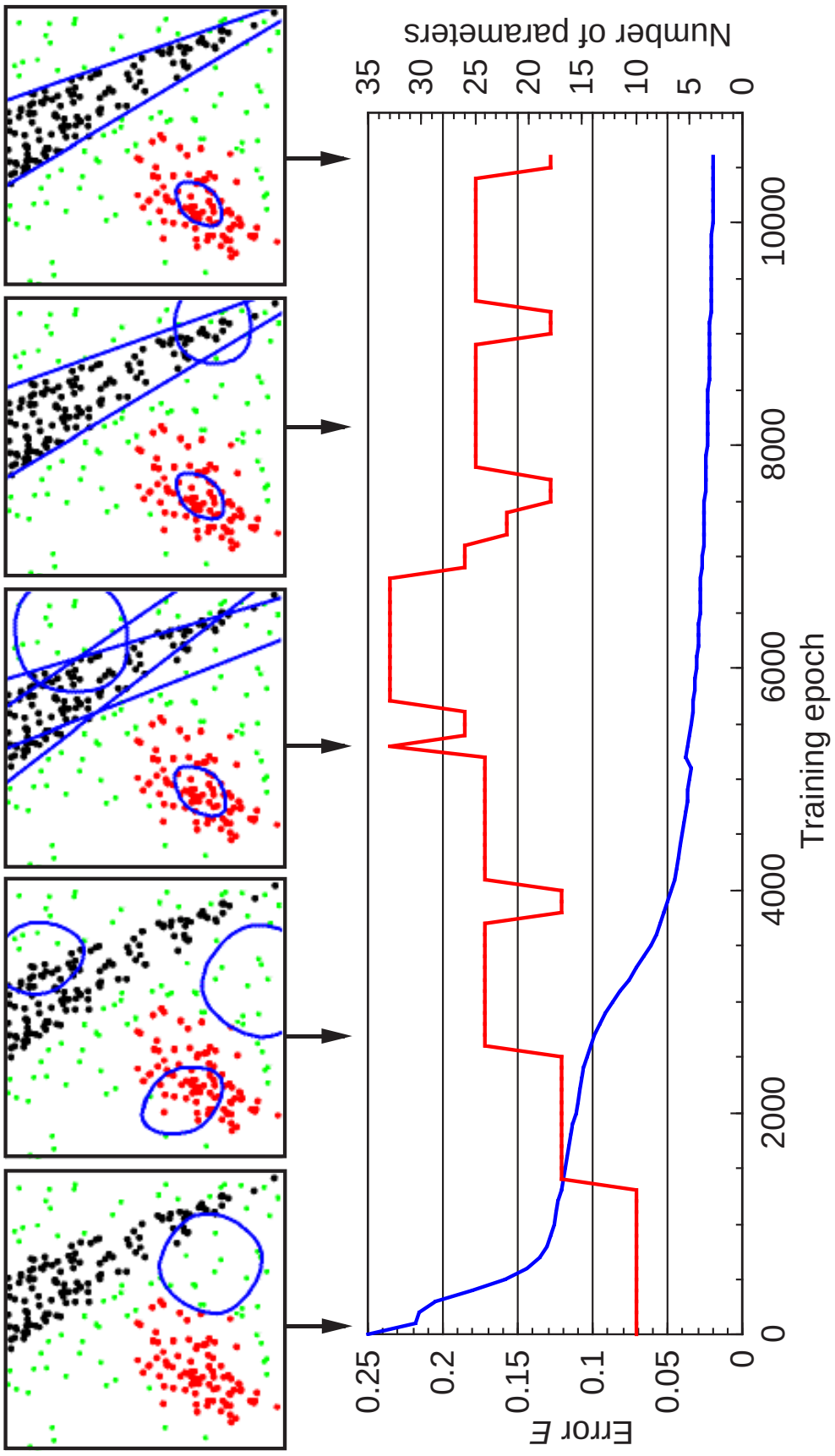


Fig. 4.10. The distribution of the three-class training set. M = 300.

Fig. 4.11 (Next page). The progress of the error reduction and the model alteration by the proposed training algorithm. In the upper half, the traces of the points where the hidden layer units respond with 0.5 are shown for the hybrid network. Note that the traces do not strictly match with the actual classification borders due to the bias. In the lower half, the training error and the number of parameters in the network are plotted.


Number of parameters
35
30
25
20
15
10
5
0
0.25
0.2
0.15
0.1
0.05
0
Error E
0
2000
4000
6000
8000
10000
Training epoch

B. Comparison

In the experiment, three types of networks with different hidden layer configurations and training methods were compared.

(Network 1) $N_1 = 5$, $N_2 = 0$, trained by BP without MS.

(Network 2) $N_1 = 0$, $N_2 = 4$, trained by BP without MS.

(Network 3) Initially $N_1 = 0$, $N_2 = 1$, trained by BP with MS.

In Fig. 4.12, the traces of the points where the hidden layer units respond with 0.5, that approximately match with the classification borders of the networks after training, are shown.

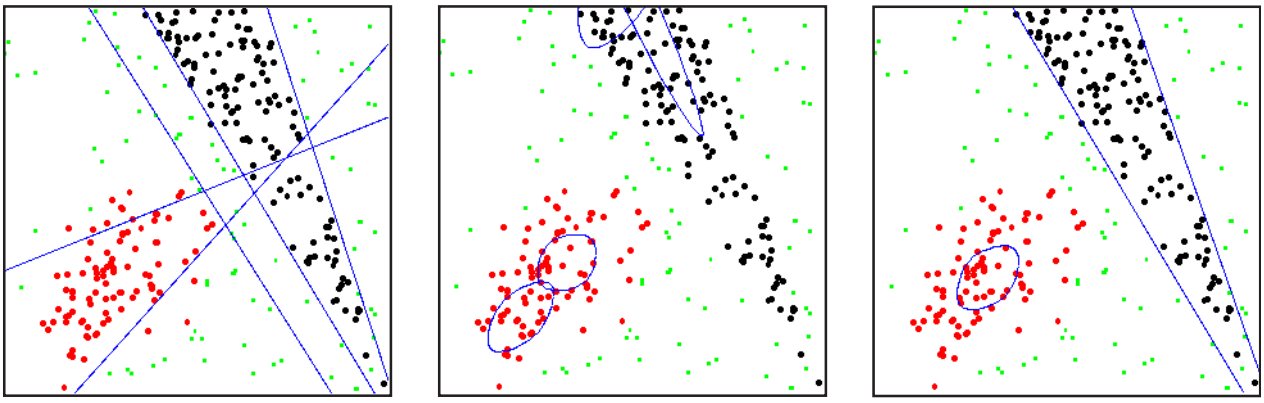


Fig. 4.12. The traces of the points where the hidden layer units respond with 0.5, approximately showing the classification borders of the networks after training. (a) Network 1. (b) Network 2. (c) Network 3.

Table 4.4. Classification rates for the test set. The numbers are the averages of 10 trials for each network.

Network (Hidden layer)	Error rate
1. $N_1 = 5$, $N_2 = 0$. No MS	7.5 %
2. $N_1 = 0$, $N_2 = 4$. No MS	5.8 %
3. $N_1 = 0$, $N_2 = 1$. Use MS.	5.4 %

In Table 4.4, the error rates to the test set averaged for 10 trials for each network, are shown. It should be noted that in the proposed network 3 with MS, the model was also determined in the training process. In all the trials for network 3, the localized cluster and the linear border were appropriately characterized using the O_2 and O_1 units. The hidden layer configuration at error conversion were $(O_1 : O_2) = (2 : 1)$ mapped as in Fig. 4.12 for eight cases. Two other cases

converged to (2 : 2) and (3 : 1). For all the cases, the second order border separating classes 1 and 3, and the linear borders separating classes 2 and 3 were successfully characterized by the O_2 and O_1 units, respectively.

It is notable that the hybrid network trained with MS achieves better generalization when compared with the others employing predetermined models and plain BP training. This is clearly due to the MS law which maintains the independence among the features extracted by the hidden layer units, and also the heuristic MS operation to switch the unit types in accordance to the nature of the class borders.

4.4 Summary

In this chapter, in addition to the map distance, the fitness index which also evaluates the nature of the class discrimination borders determined by the initial map in the new model was used.

In the first half of this chapter, A network consisting of second order sigmoid hidden units that are equivalent to RBF units, named Hyperellipsoid Clustering Network (HCN) was introduced. It was shown that a recognition system can be constructed, which points out the inputs that are distant from known training sets to be originating from an unknown class, in a small model having far less hidden layer units in comparison to the size of the training set.

In the latter half, applicability of MS to “hybrid” networks with mixed configurations of first and second order units in its hidden layer was sought. There, a novel MS operation for switching from a second order unit to a set of first order unit was introduced. By combining the method with unit fusion and installation methods introduced in Chapter 3, an algorithm for BP training with model switching for the hybrid network was introduced. The algorithm and the network was applied to an artificial classification problem requiring first and second order class borders, and it proved that better generalization could be achieved owing to an efficient characterization of the classification borders when compared with networks with uniform hidden layer unit types.

Chapter 5
Image Texture Segmentation Using Kernel Modifying Neural Networks

5.1. Introduction

Segmenting a digital image into regions having a different character is a first step to numerous methods for image analysis. Image segmentation methods can be further applied to applications such as visual inspection of manufactured products, diagnostic support in medical images and natural scene understanding. A flexible and a robust method for extraction of regional textural feature and its classification may provide a unified means for most of such applications.

For image segmentation using textures, extraction and characterization of the local texture information are essential. For extraction, 2D Gabor filters are being widely used. A 2D Gabor filter works as a bandpass filter for the local spatial frequency distribution, achieving an optimal resolution in both spatial and spatial-frequency domains [64]. The merit of employing Gabor filters for texture classification is that it provides a maximum spatial resolution in texture characterization, while keeping the filter as narrowband as possible for discrimination of the spectrally neighboring feature of different textures. It is also possible to use a bank of Gabor filters so as to represent a certain texture class with a feature vector whose elements are the amplitudes of the filter responses. This approach is known as multichannel texture classification.

Methodologies of multichannel texture classification using Gabor filters are discussed in [65]. There, each texture category within an image was assigned a Gabor filter, and the center frequency of each filter was set to a frequency where the power spectrum of the assigned texture gives the maximum peak, with some additional manual tuning. In the recognition phase, first, the amplitudes of the filters were compared after spatially smoothing by a Gaussian function, and then, each image region was classified to be of the texture class which corresponded to the filter with the maximum response.

Selection of the filter bands for efficient characterization of the various textures is one of the major issues in multichannel filtering. In the cases when the textures are unstable, e.g. peak frequencies have fluctuations. In such cases, simple comparison of the peak outputs may produce false results. This phenomenon is significant in analyzing images of natural origin, e.g. landscape images and ultrasonic echo scan of biological tissues. In such images, dominant frequency and orientation of the texture varies within a single texture class region which is to be discriminated from the other. Such a desired ability to be robust against such fluctuations may be named as unification of slightly different textures. When there is such need, a flexible classification will be made possible,

only when the amplitudes of the filter responses are interpreted to be the feature vector of the texture in focus. The subsequent classification can be left to a classifier which takes the feature vectors as inputs, and tells which texture class the focused image region belongs to.

The shortcomings of using the peak frequencies of each texture classes are also discussed in [70], where the spectral features that are “out of ordinary” are extracted using a measure named spectral feature contrast matrix, instead of spectral peaks. This policy is much robust when the textures are unstable, or when the difference of the texture to be segregated is slight. In such cases, an adaptive strategy is required both in feature extraction and classification.

If the filtering is computed as a two dimensional convolution of the image and a kernel with a finite support, this process (and the post filtering classification) can be incorporated into a classical layered neural network scheme with a local receptive field [2]. In this case, the receptive field is scanned over the whole image, producing the classification map. An integration of the convolution filter into a trainable neural network has been done in [69]. There, the connection weights between the input layer and the first hidden layer, which acts as the filter kernel, are allowed to change freely by plain backpropagation training. However, the nature of the filters adopted by learning is not necessarily bandpass, and further delineation (e.g. parametric) of the extracted feature will be difficult. Moreover, a large set of training images will be necessary as there will be far more parameters to tune by training, in comparison with using a certain parametric class of feature extractors.

In this chapter, a novel neural network architecture which incorporates the convolution filter kernel and the classifier in one, will be introduced. The network enables an automated texture feature extraction in the multichannel texture classification through simultaneous modification of the kernel and the connection weights by the backpropagation principle based training rule. In contrast to the approach in [69], the convolution kernel layer is constrained to be an array of Gabor filters having a spatially truncated finite support. Therefore, an efficient texture feature localization in the frequency domain is guaranteed.

In Section 5.2, the scheme of texture classification by multichannel filtering will be briefly reviewed. The architecture and the training rule of the proposed network model will be introduced in Section 5.3. The capability of the network and its training rule is verified using a basic problem on a synthetic texture image, and the possibilities of applying the network model to common texture classification and biological tissue classification in an ultrasonic echo image will be shown in Section 5.4. The necessity of using the higher-order statistical features for discriminating some classes of textures will be addressed in Sec. 5.5. In Secs. 5.6 and 5.7, an extended model of the network which enables bispectral feature extraction will be introduced, and the classification examples will be given. Sec. 5.8 will be devoted for the application of Model Switching (MS) using map distance detection criterion in the Gabor filter layer.

5.2. Texture segmentation by multichannel filtering

In this section, the method of texture classification by multichannel filtering will be reviewed. The typical flow of texture classification by multichannel filtering is shown in Fig. 5.1. Multiple (G) spatial frequency bandpass filters are prepared and are applied to the input image, producing G feature images, typically being the magnitudes of the filter responses. The feature values are treated as elements of a G-dimensional feature vector which indicates the class of the underlying local texture. The feature vectors are further classified with a classifier to explicitly point out which texture class the input region belongs to. By mapping the classifier response, the classification map with identical dimensions as the input image is formed.

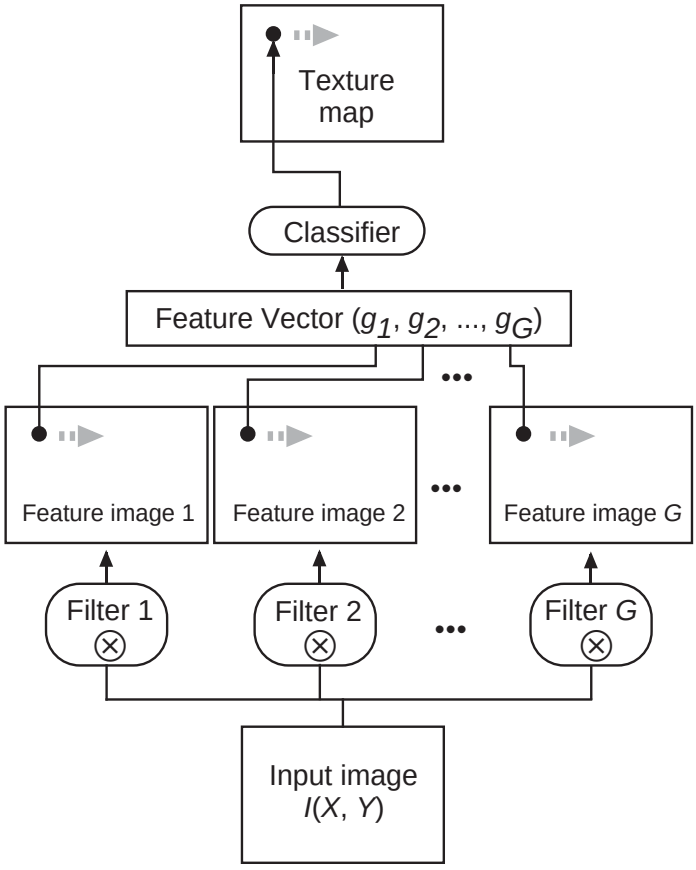


Fig. 5.1. The procedure of texture classification by multichannel filtering. The operator "⊗" denotes the two-dimensional convolution of the filter kernel and the image. The maps obtained by various convolution filters will be used as the feature vectors for use at the classifier to obtain the texture map.

An example of texture classification using multichannel filtering and a layered neural network classifier is shown in Fig. 5.2. The image used for this experiment (Fig. 5.2(a)) is an ultrasonic echo

scan of a healthy human subject in vivo [74][75]. The imaging region includes the skin, subcutaneous tissues (Region A) and the liver (Region B). The aim of this experiment is to extract the liver region from the other organs using the textural difference.

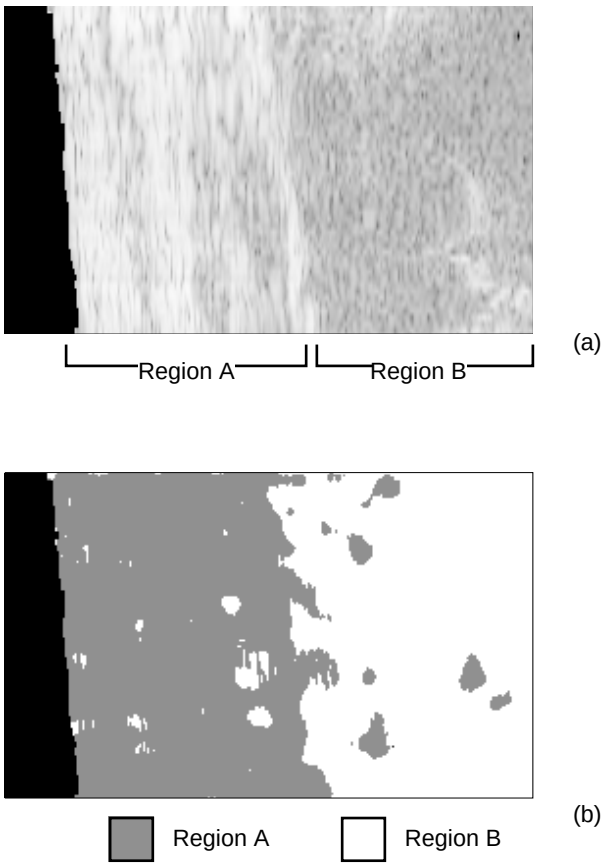


Fig. 5.2. (a) An ultrasonic echo scan image of a human subject *in vivo*. Region A includes subcutaneous tissues, fat and muscle. Region B is the underlying liver tissue. Observed external areas are masked in black. (b) Texture classification map obtained by multichannel filtering and a neural network classifier.

In this experiment, the amplitudes of nine Gabor filters were used for the characterization of the texture. The convolution kernel of a Gabor filter by the original definition is,

$$k_l(x, y; \omega_{lx}, \omega_{ly}, \sigma_{lx}, \sigma_{ly}) = \frac{1}{2\pi\sigma_{lx}\sigma_{ly}} \exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_{lx}^2} + \frac{y^2}{\sigma_{ly}^2}\right)\right\} \exp\{j\,(\omega_{lx}x + \omega_{ly}y)\}. \quad (l = 1,2,\ \ldots,\ G) \qquad (5.1)$$

It is a spatially localized wavelet as seen in Fig. 5.3(a). The filter band of the Gabor filter in Eq.(5.1) will be,

$$H_l(u,\ v) = \exp\left[-2\pi^2\{\sigma_{lx}^2(u - \omega_{lx})^2 + \sigma_{ly}^2(v - \omega_{ly})^2\}\right], \qquad (5.2)$$

which is a bivariate Gaussian-shaped band as shown Fig. 5.3(b). The parameter ω_{ls} : $(s \in \{x, y\})$ defines the center of the filter band, and the parameter σ_{ls} : $(s \in \{x, y\})$ defines the kernel spread in

each direction. This parameter σ_{ls} will be referred to as the bandwidth parameter in the following, because the reciprocal of σ_{ls} is proportional to the bandwidth.

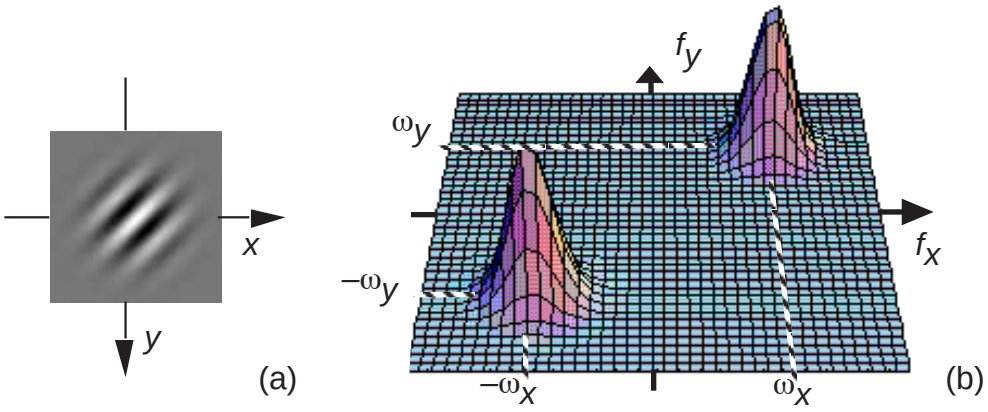


Fig. 5.3. (a) The real part of a Gabor kernel shown in grayscale. (b) Frequency response of the Gabor filter of (a).

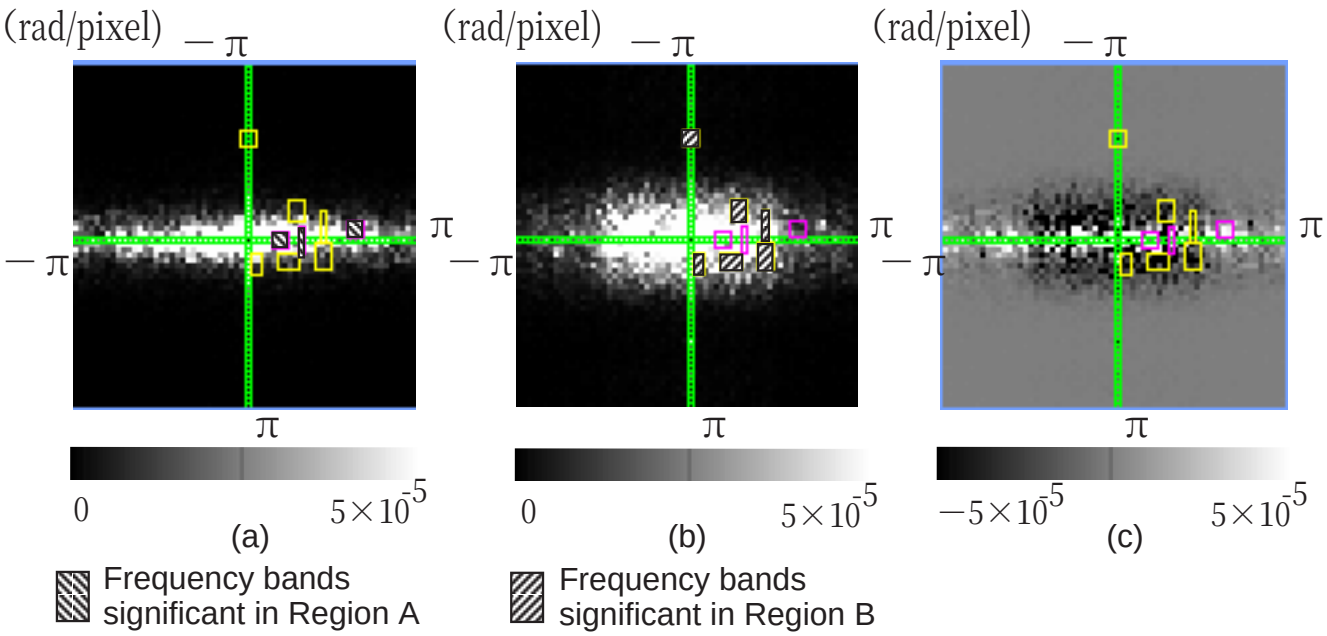


Fig. 5.4. (a) The averaged power spectrum of Region A ($\overline{P_A}$), and the filter bands selected to characterize the texture. (b) The averaged power spectrum of Region B ($\overline{P_B}$) and the filter bands selected to characterize the texture. (c) The difference of the average power spectra.

Average power spectrum of the two texture regions, denoted as $\overline{P_A}$ and $\overline{P_B}$ were calculated using five

64(pixel) × 64(pixel) subimages from each region for determining the filter bands. At the same time, the difference of the average power spectrums ($\overline{P_{diff}} = \overline{P_A} - \overline{P_B}$) was also calculated as shown in Fig. 5.4. The bands of the nine Gabor filters were chosen at frequencies where there were significant differences in the average power spectrum in view of $\overline{P_{diff}}$. Thus, the filters were selected by finding the "out of others" spectral features as in [70]. After the filter selection, the feature images for all of the nine Gabor filters were calculated to prepare the feature vectors.

Various classifier types are available after the feature images are obtained. Nearest neighbor scheme [70], layered neural networks [67][69], self-organizing maps [69], and the combination of the latter two [68] have been used in the classification stage of the multichannel filtering. Due to its flexibility in the learning stage, a two-layered neural network of (input, hidden, output) = (10, 2, 2) configuration trained by backpropagation method [5] was used here. The network inputs were chosen to be the nine filter responses and the pixel grayscale level, for the echo level contributed to tissue classification in this case. Each output unit was assigned a texture class, and was trained to fire when the input subimage was of the assigned texture. In the recognition phase, the network output was interpreted in a winner-takes-all basis.

The tissue classification map is shown in Fig. 5.2(b). It is seen that the liver region is extracted with an accuracy of 93.8%. It is reported in [65][69] that spatial smoothing of filter responses can improve the classification results, especially when the texture regions can be clearly divided.

In this section, an example of texture classification by the multichannel approach was shown. However, the method employed here suffers from a major drawback that the filter bands have to be manually selected. When the number of texture classes is small, this may be done in view of the differences in spectral features. However, such a method will be inefficient when there are many texture classes, or when the spectral differences are slight. Therefore, an automated procedure for filter band selection is in need.

5.3. The Kernel Modifying Neural Network

In this section, a neural network architecture which unifies the convolution kernel for multichannel filtering and the classifier, will be introduced. The network's training rule based on the gradient descent method enables a gradual adaptation of both the connection weights of the classifier and the Gabor filter parameters.

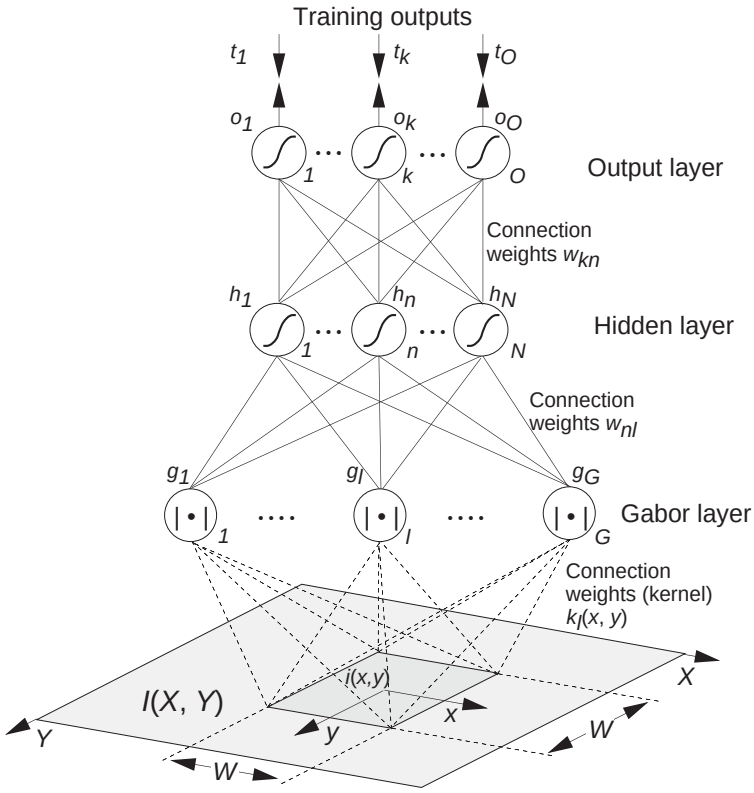


Fig. 5.5. The Kernel Modifying Neural Network (KM Net).

5.3.1 Architecture

The setup of the proposed Kernel Modifying Neural Network (KM Net) is illustrated in Fig. 5.5. The network has a layered architecture of a feedforward type, where all the units in the adjacent layers are fully connected, and no intra-layer connections exist. In this approach, the receptive field will be scanned over the input image, and the feature values for each input subimage will be given as magnitudes of the inner products of the filter kernels and the subimage. Therefore, the extracted feature and method of classification is essentially equivalent to the multichannel texture classification of Fig. 5.1.

The first unit layer of the network will be called the Gabor layer. All Gabor layer units share the same W (pixel) $\times$ W (pixel) receptive field in the input image $I(X, Y)$. The support of the Gabor filter is thus spatially truncated providing that the kernel spread be relatively small in comparison with the receptive field size. The image portion within the receptive field will be referred to as subimage $i(x, y; X, Y)$, of which the origin of the local discrete pixel coordinate xy is placed at point (X, Y) of the global coordinate.

Each of the G Gabor layer units receives the grayscale level information from all pixels in the subimage $i(x, y)$. The grayscale of the input subimage was rescaled to [0, 1] range and DC components were subtracted to provide invariance against changes in brightness and contrast. The

response of the l-th Gabor unit to the subimage $i(x, y)$ is defined as

$$g_l = A\left|\sum_{x=-W/2}^{W/2-1}\sum_{y=-W/2}^{W/2-1} k_l(x, y)\, i(x, y)\right|, \quad (l = 1,2, \ ..., \ G) \tag{5.3}$$

where k_l is the convolution kernel defined in Eq. (5.1), and A is the scaling constant which will be set to 20.0 throughout the experiments reported in this paper. Therefore, the output of a Gabor unit g_l is the magnitude of the filter response against the input subimage $i(x, y)$, whose spatial convolution kernel is $k_l(x, y)$. The parameter $\omega_{ls} : (s \in \{x, y\})$ defines the central frequency component of each direction, and the parameter $\sigma_{ls} : (s \in \{x, y\})$ defines the kernel spread in each direction. This parameter σ_{ls} will be referred to as the bandwidth parameter in the following, because the reciprocal of σ_{ls} is proportional to the frequency bandwidth of the Gabor filter.

The subsequent network layers are called the hidden layer and the output layer, having N and O units respectively. The units in these two layers respond with a sigmoid response function with (0, 1) range, of the weighted sum of the outputs of the units in the preceding layer.

5.3.2 Training

Sets of training subimages $\{i_m(x, y)\}_{m=1}^{M}$ and the desired response vectors $\{\boldsymbol{y_m} = (y_{m1}\, y_{m2} \ldots\ y_{mO})\}_{m=1}^{M}$ are used to train the network. Here M denotes the number of training sets. The training error is defined as

$$E = \sum_{m=1}^{M} E_m = \frac{1}{2}\sum_{m=1}^{M}\sum_{k=1}^{O} (t_{km} - o_{km})^2, \tag{5.4}$$

where E_m is the error for the m-th training set and o_{km} $(k = 1,2, \ ..., \ O)$ are the output of the units in the output layer.

The weights of the connection between the Gabor layer and hidden layer $(w_{nl} : (n = 1,2, \ ..., \ N, \ l = 1,2, \ ..., \ G))$, and those between the hidden layer and the output layer $(w_{kn} : (k = 1,2, \ ..., \ O, \ n = 1,2, \ ..., \ N))$ are updated according to the "batch updating" backpropagation law, as

$$w_{kn}(t + 1) = w_{kn}(t) - \frac{\eta}{M}\sum_{m=1}^{M} \delta_{km} h_{nm} \tag{5.5}$$

where

$$\delta_{km} = (y_{km} - o_{km}) o_{km}(1 - o_{km}) \tag{5.6}$$

and

$$w_{nl}(t + 1) = w_{nl}(t) - \frac{\eta}{M}\sum_{m=1}^{M} \delta_{nm} g_{lm} \tag{5.7}$$

where

$$\delta_{nm} = h_{nm}(1 - h_{nm})\sum_{k=1}^{O} \delta_{km} w_{kn}\,. \tag{5.8}$$

Here, $\{h_{nm}\}_{n=1}^{N}$, t and η denote the hidden layer unit outputs, time and the training gain constant, respectively.

In the proposed Kernel Modifying Neural Network, the backpropagated error is also utilized to modify the convolution kernel in order to minimize the classification error at the output. The filter kernels are modified preserving their Gabor nature; by modifying the central frequency $\omega_{ls} : (s \in \{x, y\})$ and the bandwidth $\sigma_{ls} : (s \in \{x, y\})$ of each filter. These parameters will be updated by the same gradient law as for the connection weights to minimize the output error E. To write the law explicitly,

$$\omega_{ls}(t+1) = \omega_{ls}(t) - \frac{\eta_\omega}{M} \sum_{m=1}^{M} \frac{\partial E_m}{\partial \omega_{ls}(t)} \qquad (s \in \{x, y\}) \tag{5.9}$$

and

$$\sigma_{ls}(t+1) = \sigma_{ls}(t) - \frac{\eta_\sigma}{M} \sum_{m=1}^{M} \frac{\partial E_m}{\partial \sigma_{ls}(t)} \qquad (s \in \{x, y\}). \tag{5.10}$$

Here, η_ω and η_σ are the training gain constants. Further description of the KM Net's training rule is shown in Appendix A.

5.3.3 Test Phase

On applying the trained KM Net for texture classification, the receptive field is moved, scanning through the area of testing. The network output corresponding to a subimage at a certain position will be interpreted as the classification result for the center of that subimage. Typically, each output unit is assigned to fire selectively in response to a known texture class. Thus, upon testing, the input region is judged to be of the texture class corresponding to the unit with a maximum output.

5.4. Experiments and results

5.4.1 Texture region extraction from a computer generated image

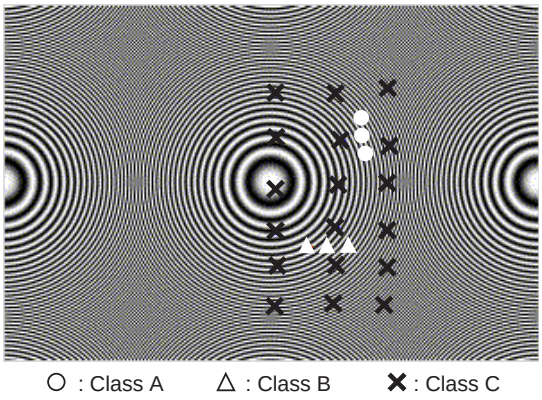


Fig. 5.6. The underlying image is the texture generated by a computer as in Eq. (5.11), with 256 gray levels and of 300 (pixel) × 200 (pixel) dimension. The image includes regions with all frequencies in all directions. The legends show the centers of the training set subimages.

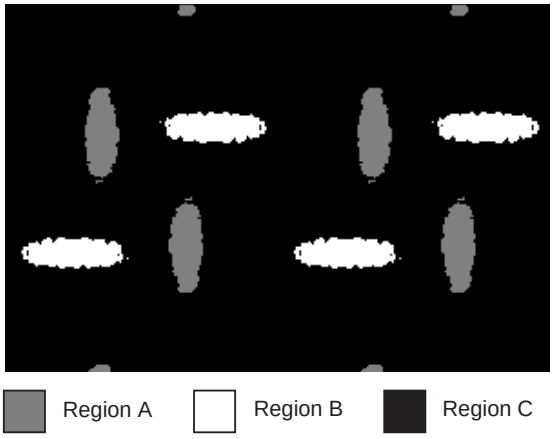


Fig. 5.7. The classification map of the image in Fig. 5.6, produced by the trained KM Net. The texture regions A and B are correctly extracted as instructed by the training sets.

A basic texture classification was carried out to test whether the Gabor layer can capture the appropriate spectral texture features through training. It is necessary that the Gabor units pass the frequency bands that show good contrast from the other class, thereby capturing the singularity in the spectral features for each texture class.

The texture image used for this experiment is shown in Fig. 5.6. It is a computer generated image which includes all monotonic frequency components in all directions. A T(pixel) × T(pixel)

image whose image gray level $U(x, y)$ is prescribed to be

$$U(x, y) = \cos[\frac{2\pi}{T}(x^2 + y^2)] \qquad (-T/2 < x \le T/2 \text{ and } -T/2 < y \le T/2), \tag{5.11}$$

is tiled to make the whole image. The tile size T was set to be 128 (pixel), and the size of the whole image is 300 (pixel) × 200 (pixel).

In this experiment, the KM Net will be trained to extract two specific texture regions from the image. The legends "E", "C" and "1" in Fig. 5.6 show the centers of the 32 (pixel) × 32 (pixel) training subimages. The two texture regions (classes A and B) to be extracted are the texture under the markers "E" and "C". The training sets marked by "1" are provided as the third class (class C), representing the remaining regions. In order to classify these three classes, at least two Gabor layer units are necessary, each filtering the spectral peak that the near-monotonic textures of classes A and B have. This experiment is for testing whether the training can modify the filter bands of the Gabor layer units to provide an efficiently discriminable set of feature vectors for classification at the subsequent layers.

The size of the network used for this experiment was $(G, N, O) = (2, 2, 3)$, which is a minimal configuration for this problem. The change of the filter bands superimposed on an average power spectrum of each texture class, and the average response of the Gabor units against the training sets of the class are shown at three stages of training in Fig. 5.8. At initial state, the filter bands are given randomly (Fig. 5.8(a)). The Gabor units do not show any selective responses at this stage. After 140 training epochs, the filter bands move towards the spectral peaks of classes A and B (Fig. 5.8(b)). Some selectivity is already seen in the firing patterns of the Gabor units, and this enables a quick decrease in the classification error at the output layer. At the 930-th training epoch, the filter bands settle at each of the peaks and some additional calibration in the bandwidth parameters are observed to conform to the oblong spectral distribution (Fig. 5.8(c)). The Gabor units' firing patterns are now highly selective. The output error is negligibly small, resulting in the convergence of the trainable network parameters due to less back-propagated error.

After training, the texture map was made by scanning the input region through the whole image and classifying them into the three classes. In the produced texture map in Fig. 5.7, the regions used as the training sets for classes A and B are correctly classified, as well as the other parts of the image having the same textures.

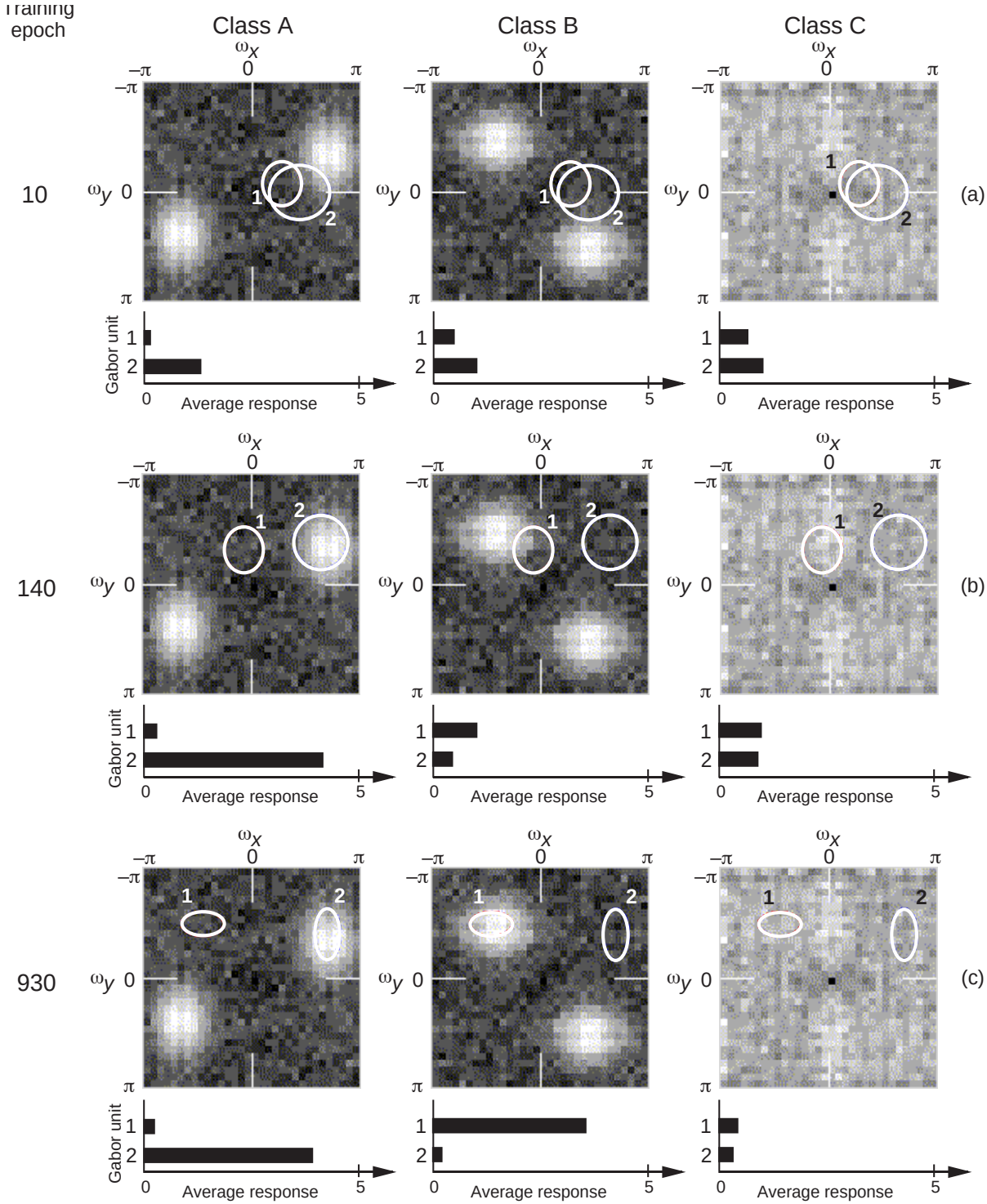


Fig. 5.8. The change of the bands of the two Gabor units during the training of texture region classification in Fig. 5.6. The underlying pattern is the average power spectrum of the texture of the class. The filter bands are drawn in white ellipses whose radii in each direction is σ_{is}^{-1} ($i \in \{1, 2\}$, $s \in \{x, y\}$). The average response of the Gabor units to the texture classes are also shown.

When tried with the minimum network size as in the above example, there were cases that the training could not be accomplished even with 10000 training epochs. Such cases were due to the network parameters being caught in local minima of the error potential function. Most among them were caused by herd effect in the early stages of training; two filters being tuned to the same peak and not being able to change either of the two to the other peak. The ratio of successful error convergence was investigated for the same problem with different number of Gabor units (G). Evaluated by 100 training attempts starting from various random network parameters, the rate of successful training was as low as 59% with G=2. However with G=3 and G=4, it was improved to 95% and 100%, respectively. Clearly, the use of several redundant filters can improve the probability of effective feature extraction. It can be more evident especially when applied to the classification of real world textures as they may have a more complex and sometimes unstable spectral features. The detection and removal of the redundant feature element (Gabor unit) is possible by several pruning methods reviewed in [38]. Also, it can be done by the proposed fusion method for the feature channel reduction. MS in attempt to assist the gradient based learning rule for efficient feature extraction in the limited model scheduling will be tried in Sec. 5.8.

5.4.2 Segmentation of common textures

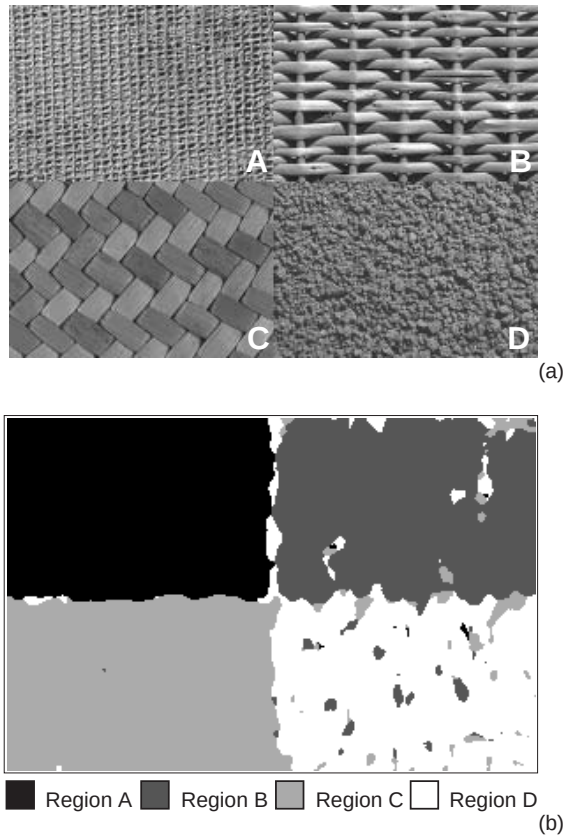


Fig. 5.9. (a) A mosaic texture image selected from the image collection in [73]. The original image names are, Fabric 17 (Class A), Fabric 14 (Class B), Fabric 0 (Class C) and Food 5 (Class D). (b) Classification map by a KM Net.

Next, the KM Net was used for the classification of common textures obtained from natural and artificial existence to examine the applicability to scene segmentation and object extraction. The image used for this experiment is shown in Fig. 5.9(a). This image is a combination of four images selected from a texture image collection [73].

The network was trained to discriminate the four image regions. For this problem, a KM Net sized (G, N, O) = (8, 6, 4) was used. Twelve subimages of 32(pixel) × 32(pixel) from each texture region were used to train the network. The training was accomplished in 1849 training epochs to attain MSE of 0.002 with training parameters $\eta = 1.0$, $\eta_\omega = 1.0$ and $\eta_\sigma = 1.0$. The classification result tried for the whole image is shown in Fig. 5.9(b). There, a maximum classification rate of 94.1% was achieved. For the average classification rates, see Sec. 5.8.2, where the average rates are compared for the normal KM Net and for the case when MS in the Gabor layer was used.

5.4.3 Tissue classification in an ultrasonic echo scan image

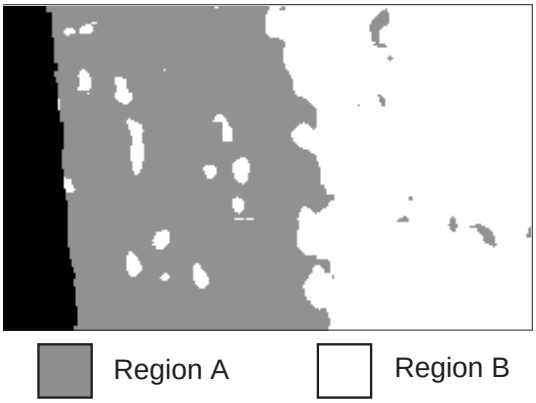


Fig. 5.10. A tissue classification map obtained with use of the KM Net for the ultrasonic echo scan in Fig. 5.2(a).

The KM Net was also applied to the tissue classification problem of Fig. 5.2. The configuration of the network was (G, N, O) = (9, 6, 2), and 21 subimages from each texture regions were used as the training sets. The rate of classification when applied to the same image was 94.2%; the obtained classification map is shown in Fig. 5.10. Although the rate is only slightly better when compared with the result shown in Fig. 5.2(b), it is clear that the filter band selection process could be replaced by the KM Net's training rule based on the reduction of the classification error. It is also noteworthy that the result in Fig. 5.10 has been obtained without any grayscale information (zero frequency component) of the echoscan image.

5.4.4. Discussion

The ranges of the filter parameters

In all experiments, the filter bands were allowed to change freely according to the modification measures calculated by the training. Naturally, the bands of some filters spread across the Nyquist frequency borders. In the initial approach, the central frequency and the bandwidth of the Gabor filters were allowed to change in range where the aliased energy could be kept small. Making such hard limits in the parameter range, however, typically caused the training processes to be caught in artificial local minima in the error potential surface. Therefore, in the experiments in the previous section, the central frequencies were allowed to change freely in the 2D frequency domain with an iterative structure of a torus surface. However, the range of the bandwidth parameter σ_s ($s \in \{x, y\}$) was limited to be in

$$1.5 \le \sigma_s \text{ (pixel)} \le 0.5W, \qquad (s \in \{x, y\}) \tag{5.12}$$

so that the localization of information could be maintained in both space and frequency domains

under the limitation of using an input subimage support of a predefined size. The lower bound was decided so as to maintain the bandpass nature of

$$0.1(\text{radian/pixel}) \geq \Sigma_s \equiv (2\pi\sigma_s)^{-1}, (s \in \{x, y\}) \tag{5.13}$$

where Σ_s is the standard deviation of the band of the Gabor filter in the frequency domain.

The initial values of the filter parameters

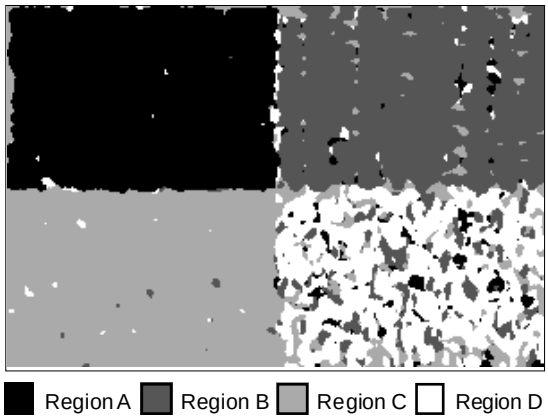


Fig. 5.11. The classification map of the mosaic image in Fig. 5.9(a), produced by a network with relatively wideband filter sets.

It was observed that the kernel parameters of the trained network differed by the initial parameter settings, and each network produced a different classification map. Since the size of the network is usually chosen to be larger than minimal to accomplish the training, there are countless parameter combinations that a network can take to fulfill the training task. In Fig. 5.11, a classification map for the texture mosaic of Fig 5.9(a), obtained by another trained KM Net is shown. The difference in the two networks that produced Fig. 5.9(b) (Case A) and Fig. 5.11 (Case B) are significant in the bandwidth parameters. The averages of the bandwidth parameters were 4.12 for Case A and 2.79 for Case B. Such a difference was caused by the random bandwidth parameter settings at the initial states of the training. In Case A, the bandwidth parameters were randomly chosen from [4.5, 5.5] range. In Case B, the section was shifted to [1.5, 2.5] range. In Case A, it started the training with narrowband filters, whereas in Case B, the filters were relatively wideband at initial state. Both networks still maintained the initial character at the end of their training.

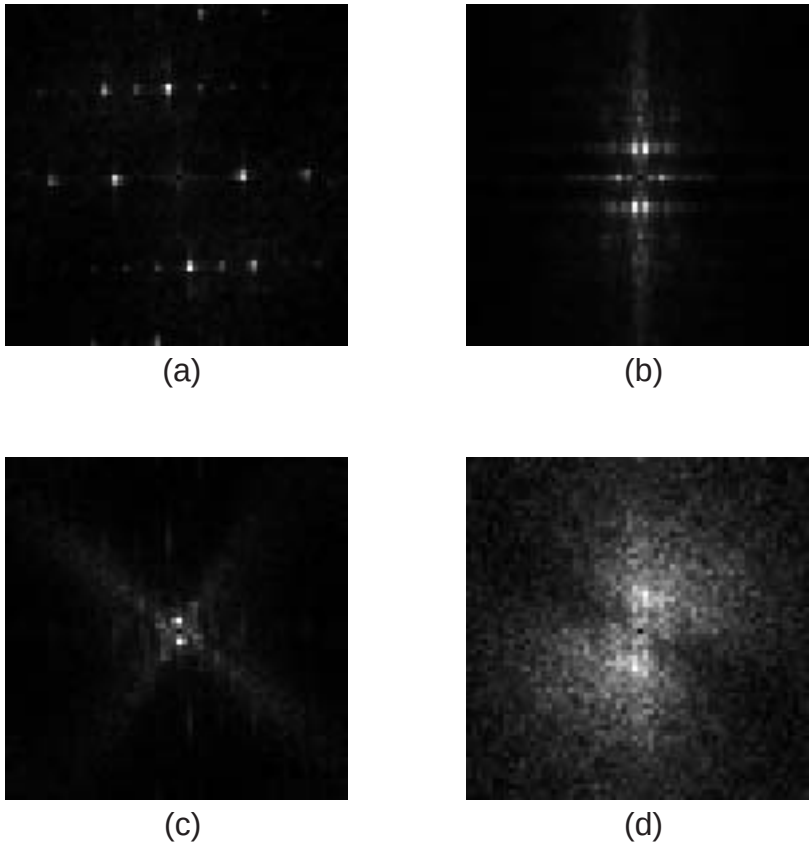


Fig. 5.12. The averaged power spectra of the training sets of the texture regions A~D shown in (a)~(d), respectively.

The averaged power spectra of the training sets are shown in Fig. 5.12. The rate of misclassifications, especially the speckle-like errors as seen in wideband textures such as Class D, had a clear correlation with the value of the bandwidth parameter of the trained network. When a Gabor filter grows wideband, its spatial kernel support will in turn become smaller, eventually causing the response of the filter to be unstable, affected only by local pixel values. This is a restriction caused by employing Gabor filters of single resolution, and may be improved in a multiresolution approach. The instabilies of the filters with small support are significant in wideband textures whose local spectral component emerge stochastically. If the texture regions to be segmented are comparatively larger than the training subimage, the resulting classification map will be much smoother when the training was started from a relatively narrowband state, and gradually turning wider-band to accommodate the spectral instability of the texture class. Speckle-like misclassifications may also be suppressed by using *a priori* information such as the size of the single texture class region; e.g. restricting the ranges that the kernel parameters can take.

For a relatively narrowband texture in which the spectral patterns have distinct peaks that can be used as the “key band” of the class, the differences seen in Figs. 5.9(b) and 5.11 are slight. However, since these texture classes produced error surfaces with narrow minima having very steep

gradients, there were cases when the errors did not converge smoothly. In order to avoid such a problem, a minor change in the parameter modification method was made, and is mentioned in Appendix A.

The initial values of the central frequencies were chosen from the range of $[-\pi/2, \pi/2]$, since the key differences of the spectral patterns tend to appear in lower frequencies.

5.5 Extraction of local phase information

5.5.1 Limitations of the Gabor filter feature extractors

In the KM Net model, the output of a Gabor unit is the magnitude of the convolution of the complex Gabor kernel and the local image, as described in Eq. (5.3). For many texture classes, this method extracts the spectral features that are effective for classification in the subsequent layers. However, there exists a class of textures that are unable to be discriminated using this method, although they are visually different.

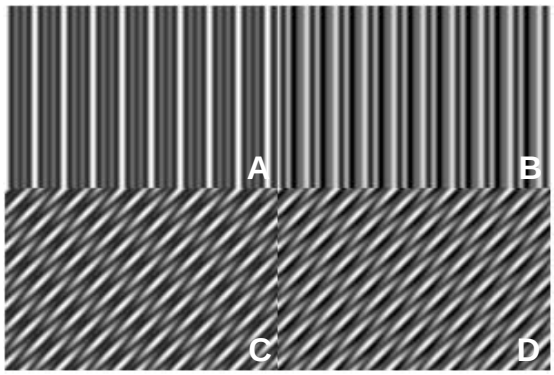


Fig. 5.13. A computer-generated texture mosaic image.

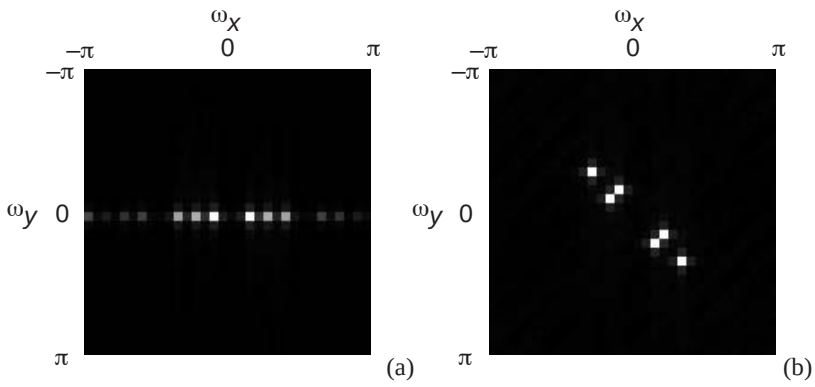


Fig. 5.14. (a) The local power spectra of the texture classes A and B of Fig. 5.13. (b) The local power spectra of the texture classes C and D of Fig. 5.13.

In Fig. 5.13, a mosaic of four synthetic textures is shown. The spatial variation of the graylevel of each texture class was generated according to the following formulae.

Class A :

$$I_A(x, y) = \cos(\alpha x) + \cos(2\alpha x) + \cos(3\alpha x)$$

Class B :

$$I_B(x, y) = \cos(\alpha x) + \cos(2(\alpha x + \phi)) + \cos(3\alpha x)$$

Class C :

$$I_C(x, y) = \cos(\omega_{x1} x + \omega_{y1} y) + \cos(\omega_{x2} x + \omega_{y2} y) + \cos((\omega_{x1} + \omega_{x2})x + (\omega_{x1} + \omega_{x2})y)$$

Class D :

$$I_D(x, y) = \cos(\omega_{x1} x + \omega_{y1} y) + \cos(\omega_{x2} x + \omega_{y2} y + \theta) + \cos((\omega_{x1} + \omega_{x2})x + (\omega_{x1} + \omega_{x2})y)$$

Texture classes A and B both include the same frequency components, of fundamental, second and third harmonic frequencies. Therefore, both textures share the same local power spectrum as shown in Fig. 5.14(a). Since the feature extracted by Gabor filters depend on the local power spectrum, they are indiscriminable using the Gabor filters. The difference of the two textures are in the phase term ϕ of the second harmonic component. Although the observed difference is much slight, there is a similar relation between texture classes C and D; with identical local power spectrums and different phase relation in their frequency components. The power spectrum is shown in Fig. 5.14(b). In order to discriminate these texture classes, characterization of the phase relation among the frequency components is vital.

5.5.2 Higher-order KM Net and extraction of phase information

First, it will be shown that the feature extraction employed in the KM Net is equivalent to obtaining the local second order moment spectrum of the image signal. For the sake of simplicity, the following discussion will be for one dimensional signals only. However, it can be easily extended to the two-dimensional case.

Let us define the convolution kernel as

$$k(x, \omega) = w(x)\, e^{j\omega x}, \tag{5.14}$$

where $w(x)$ is the window function centered at x. The window function is a Gaussian function in a Gabor filter. The output of a Gabor layer unit g can be written as

$$g = \left| i(x) \otimes k(x, \omega) \right| = f_{att}\left(\left| i(x) \otimes k(x, \omega) \right|^2 \right), \tag{5.15}$$

with the attenuation function being $f_{att}(x) = \sqrt{x}$, and the operator '$\otimes$' denoting the spatial convolution. The power of the convolution in Eq. (5.15) can be further expressed as

$$p(x,\,\omega) = |i(x) \otimes k(x,\,\omega)|^2 = \int i(x_1)w(x-x_1)e^{j\omega(x-x_1)}dx_1 \left[\int i(x_2)w(x-x_2)e^{j\omega(x-x_2)}dx_2\right]^*$$

$$= \iint i(x_1)i(x_2)w(x-x_1)w(x-x_2)e^{-j\omega(x_1-x_2)}dx_1dx_2\ . \qquad (5.16)$$

The integral ranges in Eq. (5.16) and the following are from $-\infty$ to $+\infty$. By assuming $\xi = x_1 - x_2$,

$$p(x,\,\omega) = \iint i(x_2+\xi)w(x-x_2+\xi)i^*(x_2)w^*(x-x_2)e^{-j\omega\xi}dx_2d\xi \equiv \int m_2(\xi\,;\,x)e^{-j\omega\xi}d\xi \qquad (5.17)$$

where

$$m_2(\xi\,;\,x) = \int i^*(x_2)w^*(x-x_2)i(x_2+\xi)w(x-x_2+\xi)dx_2\ . \qquad (5.18)$$

Now if the window function is even symmetric, i.e. $w(x) = w(-x)$, it can be put as

$$m_2(\xi\,;\,x) = \int i^*(s+x-\frac{\xi}{2})w^*(s-\frac{\xi}{2})i(s+x+\frac{\xi}{2})w(s+\frac{\xi}{2})ds\quad , \qquad (5.19)$$

by defining $s = x_2 - x + \xi/2$. Thus, $m_2(\xi\,;\,x)$ is the local second order moment of signal i at x, and $p(x,\,\omega)$ in Eq. (5.17) is a local second order moment spectrum of signal i. With this type of feature extraction, the phase information in the signal is lost since the value of $p(x,\,\omega)$ is real and nonnegative. This method is blind to the differences in the signals that share a same local power spectra but have different phase relations among the frequency components.

One method which is known to be effective to extract the relative phase information is the use of higher-order spectra of the signal. There are numerous time-frequency distributions such as the spectrogram and the Wigner distribution [59], and there are several time-multiple-frequency distribution models being their higher-order extensions [60][62]. Since it was shown that the original KM Net extracts the local second order moment spectrum of the input signal, a general model which extracts the n-th order moment spectrum

$$M_n(x,\,\omega_1,...,\,\omega_{n-1}) = \iint m_n(\xi_1,...,\,\xi_{n-1}\,;\,x)e^{-j(\omega_1\xi_1+\,...\,+\,\omega_{n-1}\xi_{n-1})}d\xi_1\,...\,d\xi_{n-1}\,, \qquad (5.20)$$

with $m_n(\xi_1,...,\,\xi_{n-1}\,;\,x)$ being the n-th order windowed moment (autocorrelation) function. The KM Net which extracts the higher-order moment spectral feature will be called the *Higher-Order Kernel Modifying Neural Network* (HOKM Net).

Each higher-order distribution model inherit its second-order version's merits and shortcomings. The higher-order Wigner distribution [60] has the simplest form for extracting the local higher-order moment. However, the spurious response to the so-called cross-frequency terms can be a fatal shortcoming. Since the previous model employed the feature equivalent to windowed power spectrum, it would be natural to seek the windowed local higher-order moment spectrum as the texture feature. Considering the fact that the estimation variance will grow with the order, and also the

computational cost, the third-order spectrum of the image signal will be extracted and will be used.

The local third-order moment spectrum, or the local bispectrum is defined as

$$M_3(x,\ \omega_1, \omega_2) = \iint m_3(\xi_1,\ \xi_2\ ;\ x)e^{-j(\omega_1\xi_1+\omega_2\xi_2)}d\xi_1 d\xi_2 \tag{5.21}$$

with

$$m_3(\xi_1,\ \xi_2\ ;\ x) = \int i^*(u+x)w^*(u)i(u+x+\xi_1)w(u+\xi_1)i(u+x+\xi_2)w(u+\xi_2)du \tag{5.22}$$

where w is the window applied to the data. Since the local bispectrum M_3 is defined as a 2D Fourier transform of the local third-order moment function m_3, it may be possible to define the feature extraction process as the convolution of some kernel and the high order moment function. However, an alternative approach which proves to be easier to embed in the layered neural network form, will be employed as a simple extension to the KM Net model.

In the following, it will be shown that the local bispectrum can be obtained by using the products of three complex Gabor filter outputs. Let us put $v = u + x$, $v_1 = \xi_1 + u + x$ and $v_2 = \xi_2 + u + x$. Then Eq. (5.21) can be rewritten as,

$$\begin{aligned} M_3(x,\ \omega_1, \omega_2) &= \iiint i^*(v)w^*(v-x)i(v_1)w(v_1-x)i(v_2)w(v_2-x)e^{-j\omega_1(v_1-v)}e^{-j\omega_2(v_2-v)}dv\,dv_1\,dv_2 \\ &= \int i^*(v)w^*(v-x)e^{j(\omega_1+\omega_2)(v-x)}dv \int i(v_1)w(v_1-x)e^{-j\omega_1(v_1-x)}dv_1 \int i(v_2)w(v_2-x)e^{-j\omega_2(v_2-x)}dv_2\ . \end{aligned} \tag{5.23}$$

If the symmetry of the window function as $w(x) = w(-x)$ can be assumed, then,

$$M_3(x,\ \omega_1, \omega_2) = \big(i(x) \otimes k(x,\ \omega_1+\omega_2)\big)^* \big(i(x) \otimes k(x,\ \omega_1)\big)\big(i(x) \otimes k(x,\ \omega_2)\big)\,. \tag{5.24}$$

Eq. (5.24) shows that the local bispectral information can be extracted as the product of three complex convolution filter outputs. The Gaussian function will also be used as the window $w(x)$ in the following. However, for the best localization of information in both spatial and bispectral domains, other choice of window function may prove to be superior.

For higher-order spectral features, e.g. local trispectrum, it can be shown in a similar way as in Eqs. (5.21) ~ (5.24), that the products of the windowed kernel convolutions will enable extraction of equivalent measures.

5.6 Bispectral Kernel Modifying Neural Network (BKM Net)

5.6.1 Architecture

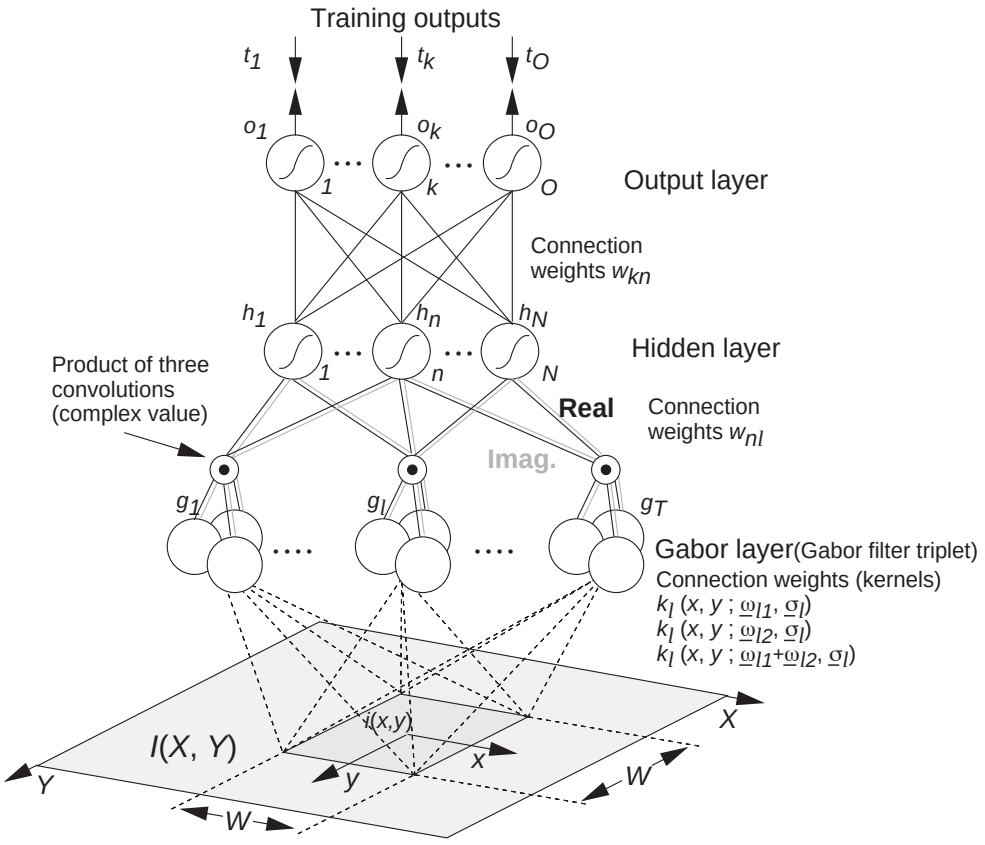


Fig. 5.15. The Bispectral Kernel Modifying Neural Network (BKM Net).

For obtaining the bispectral feature defined in Eq. (5.24) in a neural network model, an extension to the KM Net model, named the Bispectral Kernel Modifying Neural Network (BKM Net) will be proposed. Three Gabor kernels $k(\omega_1)$, $k(\omega_2)$ and $k(\omega_1 + \omega_2)$ will work in a unit (triplet) as shown in Fig. 5.15. This unit will be referred to as bispectral Gabor unit in the following. Parameters ω_1, ω_2 and σ will be modified by training. This amounts to six parameters per bispectral Gabor unit for texture classification application. No attenuation function as in Eq. (5.15) is applied before feeding to the hidden layer in this model. Use of attenuation functions may prove to be beneficial when the feature element range spread to a very wide range.

As another modification to the original KM Net model, the hidden layer will receive the weighted complex feature vectors. One way of handling complex features may be employing a classifier stage consisting of units using complex input-output relations. Layered neural network models using complex signals have been proposed [13]. Instead, the real and imaginary parts of the complex features will be treated as individual real features, and real value units will be used in the

hidden and output layers.

5.6.2 Training rule

The BKM Net will also be trained using a backpropagation-based training rule. The weights between the output and the hidden layers, employ the same modification method as described in Sec. 5.3.2. The weights between the Gabor layer and the hidden layer are also updated using the same strategy. It should be noted, however, that each hidden layer unit has two connections between a Gabor layer unit for real and imaginary part of the filter output, and that the modification rule must be applied to the both weight groups independently.

The bandwidth σ and the central frequencies ω_1 and ω_2 of the l-th bispectral Gabor unit will be modified as

$$\Delta\sigma_l = -\frac{\eta_\sigma}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial\sigma_l} = -\frac{\eta_\sigma}{M}\sum_{m=1}^{M}\left[\frac{\partial E_m}{\partial(Re\,[M_{3l}])}\frac{\partial(Re\,[M_{3l}])}{\partial\sigma_l} + \frac{\partial E_m}{\partial(Im[M_{3l}])}\frac{\partial(Im[M_{3l}])}{\partial\sigma_l}\right] \tag{5.25}$$

and

$$\Delta\omega_{li} = -\frac{\eta_\sigma}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial\omega_{li}} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\left[\frac{\partial E_m}{\partial(Re\,[M_{3l}])}\frac{\partial(Re\,[M_{3l}])}{\partial\omega_{li}} + \frac{\partial E_m}{\partial(Im[M_{3l}])}\frac{\partial(Im[M_{3l}])}{\partial\omega_{li}}\right], \quad (i\in\{1,\ 2\}) \tag{5.26}$$

with $Re[x]$ and $Im[x]$ denoting the real and imaginary parts of x, respectively. See Appendix B for further description of the training measures.

5.7 Application of the BKM Net and discussion

5.7.1 Texture region extraction - Comparison with the KM Net -

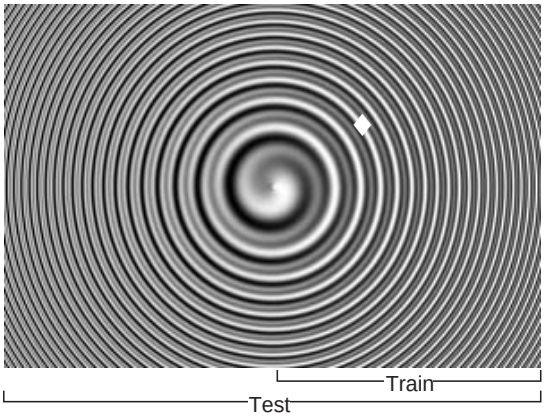


Fig. 5.16. A synthetic image of a texture with a continuously-changing fundamental frequency and the second-order harmonic component. The phase relation of the two components are also changed continuously. Among the right half region which was used to train the networks, texture class A was chosen to be of the texture under the marker '◆'. Class B was “the others”.

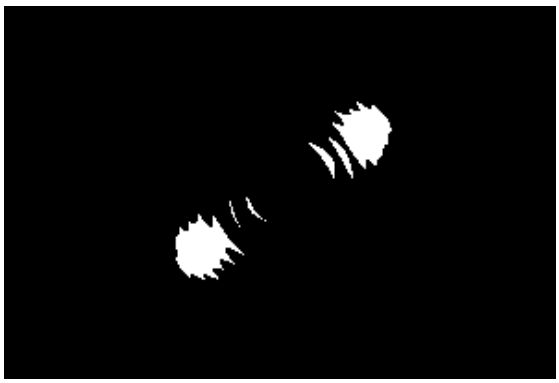


Fig. 5.17. The classification map by the KM Net. Another region in the left half of the image was also classified to be class A.

Fig. 5.18. Classification by the BKM Net. Only the region with the specific frequency component and phase relation was classified to be class A.

In order to evaluate the basic properties of the BKM Net and to compare its feature extraction ability with the original KM Net, a synthetic image shown in Fig. 5.16 will be used. For generating the image, a polar coordinate system $r\theta$ whose origin is at the center of the image, will be defined. The gray level distribution of the image $L(r,\theta)$ is defined as

$$L(r,\ \theta) = \cos(ar^2) + \cos(2ar^2 + \theta)\ .$$

Both networks were trained to extract the texture under the marker "◆" in Fig. 5.16. The training sets were chosen from the right half region of the image. The texture at marker "◆" and the other regions were given as the training sets of classes A and B, respectively. A KM Net of (G, N, O) = (6, 3, 2), and a BKM Net of (T, N, O) = (2, 3, 2) was used. Here, T=2 denotes that two bispectral Gabor units were used, each consisting of a Gabor filter triplet and extracting the local bispectral feature. The classification result by both networks are shown in Figs. 5.17 and 5.18. The KM Net classifies a region in the left half of the image which includes the same frequency components as the training set, to be of class A. However, since both the frequency and the phase relation of the fundamental and the second harmonics has been extracted in the BKM Net, only the trained region is classified to be of class A.

5.7.2 Segmentation of a computer generated image

The BKM Net was also tried for classifying the texture classes in the mosaic of Fig. 5.13. A network of (T, H, O) = (3, 6, 4) was used for classification, which was trained using 12 subimages per texture class. The obtained classification map Fig. 5.19 shows a smooth segmentation of the texture regions, although with some misclassifications at the texture edges where the filter responses tend to be unstable. After training, the bands of the Gabor filters that form the bispectral Gabor units were found be tuned to the harmonic peaks of the texture signals, as seen in Fig. 5.20(a). The average responses of the bispectral Gabor units (local bispectrum) are shown in the bar graphs of Fig. 5.20(b). It clearly shows the different response patterns against the texture class pairs with identical

local power spectra.

Fig. 5.19. Classification map of the texture mosaic in Fig. 5.13, obtained using a BKM Net. Classification rate : 93.9% .

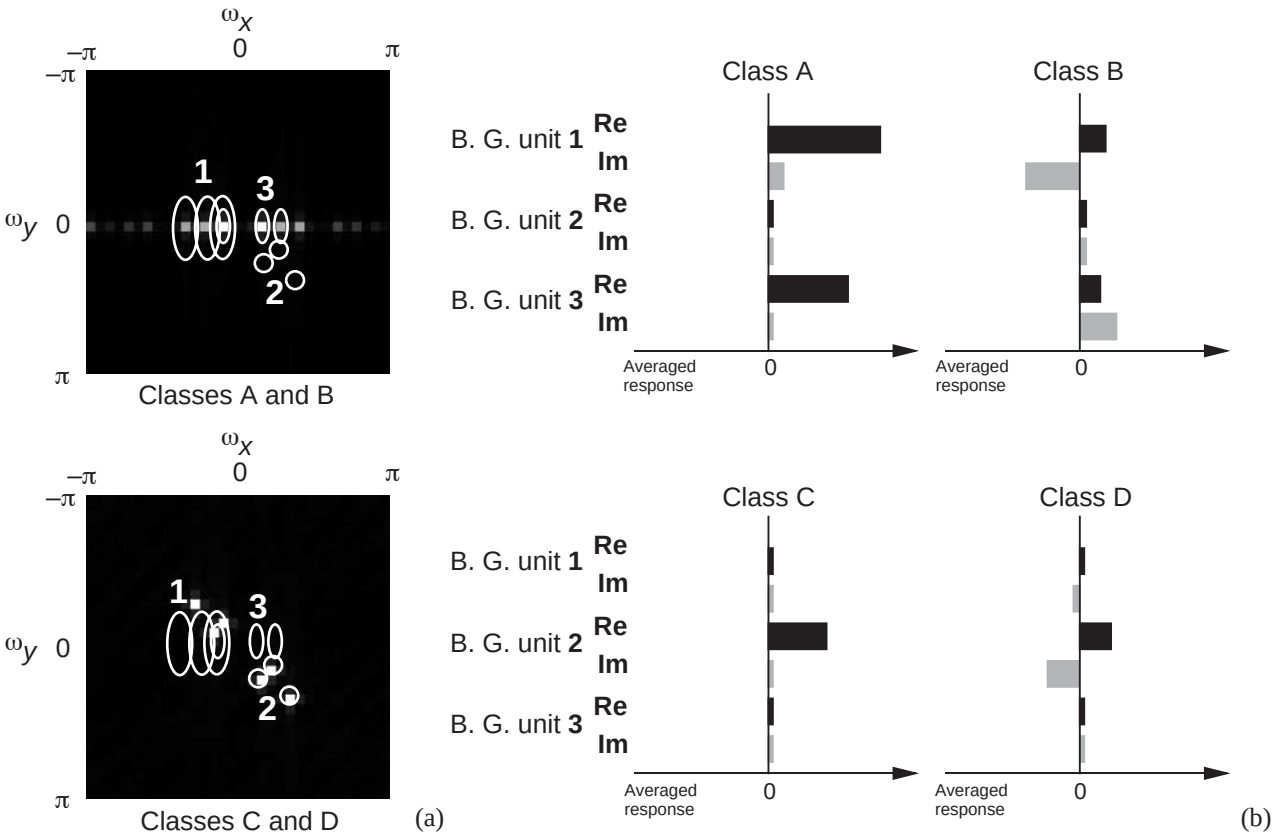


Fig. 5.20. (a) The local power spectra of texture classes, with the bands of the Gabor filter triplets forming the bispectral Gabor units overlayed. The numbers correspond to the bispectral Gabor units. (b) The averaged responses of the products of the bispectral Gabor units (local bispectrum) against the class textures.

5.7.3 Discussion

Bispectral feature extraction and Gaussian noise

A typical motive for using bispectrum in signal feature extraction is in its immunity to additive Gaussian noise [60]. However, in texture segmentation where feature extraction using only one short train of data (subimage) is required, i.e. when no averaged feature from large portion of data is available, the extracted feature will be affected by the additive Gaussian noise. When classification of several sufficiently long data train is of concern, averaged signal feature can be extracted from multiple signal portions during both training and recognition. In such a case, the effect of the Gaussian noise can be eliminated. Even in such cases, the proposed BKM Net will be effective for adaptive selection of frequency bands through training. In the recognition phase, the trained filter parameters can be used for bispectrum estimation from the data portions. After obtaining the averaged bispectral features, the hidden and output layer part can be used as the classifier of the bispectral feature vectors.

Using spectral features of orders higher than three

So far, two types of network models were introduced for texture classification. The two models enabled the extraction of spectral and bispectral features during training. As noted in Sec. 5.5, extraction of spectral features of orders four and higher, is also possible. However, since the size of the data necessary for stable estimation of higher-order spectral features will increase with the order, extraction of features higher than three may not be feasible in texture classification. When longer data samples are available in the recognition phase, as mentioned in the previous subsection, the approach introduced in this paper may also prove to be successful.

5.8 Model switching in the Gabor layer

5.8.1 The inefficiency in the Gabor layer training

By altering the filter parameters in the Gabor layer using the KM Net's training rule, the filters were tuned to bands that show significant difference between the texture classes, as found in the classification of synthetic textures in Sec. 5.4.1. When classifying the real world textures of Fig. 5.9, feature extraction of the same type were possible by automatic tuning of the Gabor filters.

In Fig. 5.21, the Gabor filter bands and their average responses against each texture classes of Fig. 5.9 are shown for a trained KM Net. Although some of the spectral peaks are captured especially in the narrowband classes and the firing patterns of the filter array are clearly discriminable, there are points that can be improved. First, some of the filters are responding very selectively, however, there are filters such as filter #6 that do not contribute to showing contrasts between classes. Such redundant filters can exist when the number of the filters are abundant, however, mostly, they are caused by being caught in local minima in the error potential function.

Second, there are filters that actually extract the same feature. For example, the bands of the filters #7 and #8 are seen to be overlapping, and the responses to the training sets are very similar.

In order to eliminate such inefficiencies in the Gabor layer feature extraction, MS for MLPs tried in Chapter 3, employing Fitess Indices defined by the degree of map inheritance,should be applicable. In the next section, repetitive reduction of the filters by the fusion method and their immediate re-installation during the BP training will be tried, and the contribution to the training speed and the classification rate will be evaluated.

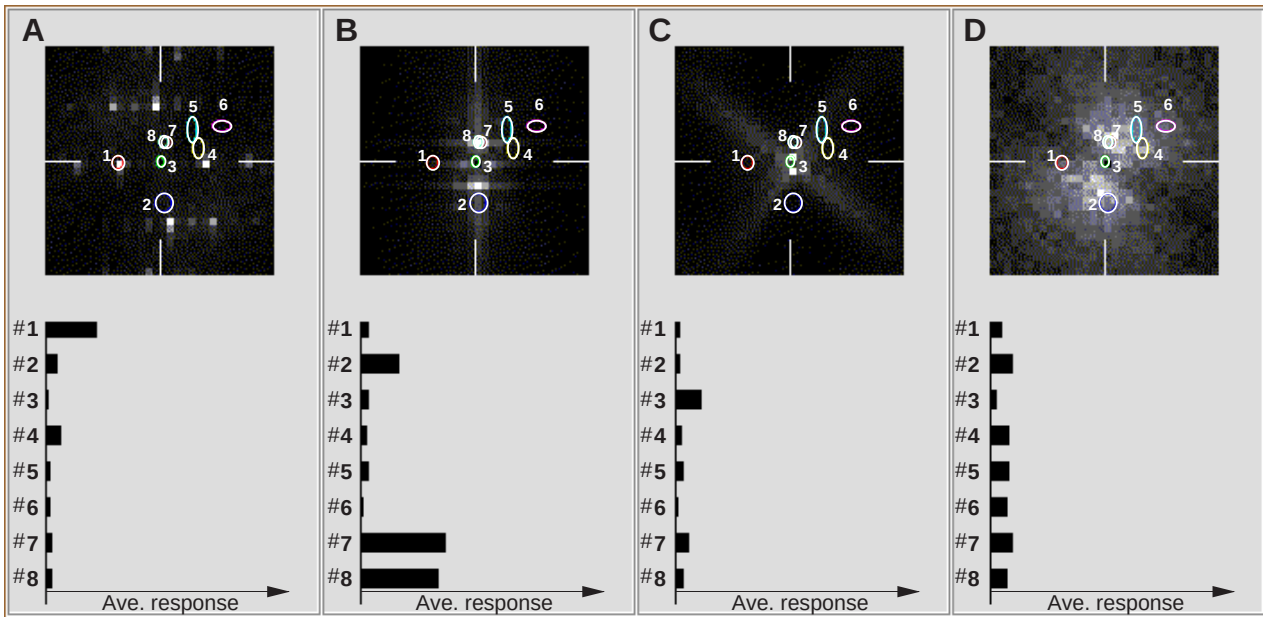


Fig. 5.21. The Gabor filter bands and their average responses against the class training sets of Fig. 5.9. The grayscale image show the average power spectra of the training sets of the texture classes. The filter bands are shown in white ovals.

5.8.2 Model switching by unit fusion in the limited model

Model switching with Fitness evaluation of map inheritance will be applied to the Gabor layer of the KM Net. On using MS, limited model scheduling will be used. In the positive mode, the model will be expanded and immediately reduced to a different map of in the same model. In the negative mode, the model will be reduced, and will be expanded right away. When methods of model expansion and reduction are the same for both switching modes, the double-operation-switching will be identical for the two modes, only the order is different. This scheduling, however, will be beneficial when the generalization ability is in concern. As discussed in Sec. 3.7, fusion of units with low variace will contribute to reducing the null space to the traiing sets. Also, fusion of unit pairs with high correlation will remove dependent bases. If the recruitment of the reduced factors follow this reduction, there is a chance that the re-installed base will become a basis function of better quality. By repeating this operation of MS in the limited model, it is assumed that the functional space spanned by the basis

function set will be kept large, with less null space factors. The features of the KM Net trained with BP with MS will be shown in Table 5.1.

Table 5.1. Features of the training method with MS.

Objective of MS	1. Efficient training 2. Higher generalization ability
General • Network type	KM Net
• MA scheme	MS consulting MS Index. MS applied to the Gabor layer
MS • Scheduling	Limited model
• Fitness Evaluation	Map preservation (inheritance)
• Fitness Index	$I_F(f_N,\ f'_N) = 1 - D(f_N,\ f'_N)$
• Stress Index	$I_S(E, \frac{\partial E}{\partial t}) = \mathrm{sgn}\,(E - E_0)\,\mathrm{sgn}\,(\frac{\partial E}{\partial t} + I_0)$
• MS Index	$I_{MS} = I_F I_S$
• MS candidate	All possible by unit fusion followed by unit installation

5.8.3 Experiment

Learning and classification of four texture classes in Fig. 5.9 was tried with different Model Alteration (MA) strategies using the KM Net. Since the filter responses tend to be fluctuating especially in non-deterministic textures as Class 4, the classification map typically includes speckle-like misclassifications. Such misclassfications can be reduced by spatially smoothing the feature responses in texture classification [65][69]. Instead of smoothing, we used the local feature directly and obtained the classification rate for the whole image to measure the efficiency of the extracted feature. Since the upper bounds of the map distances caused by model switching can be estimated using Theorems 1 and 2, model switching could be applied to both the input layer and the hidden layer simultaneously, always selecting the switch which causes minimum map distance. However, it was considered to be more important to show the difference clearly in a simpler case. Therefore, in this experiment, MA is applied within the Gabor layer only. When the number of filters (channels) was reduced by model switching, a new filter with a random filter parameter set ($\omega_x,\ \omega_y,\ \sigma_x,\ \sigma_y$) was installed. Thus, the network model was always unchanged. This style was taken in order to make a

fair comparison between the KM Net training without model switching. Two types of KM Net configurations were trained and tested using five different MA strategies described in the following.

The procedure of training and MA in the experiments is shown in Fig. 5.22.

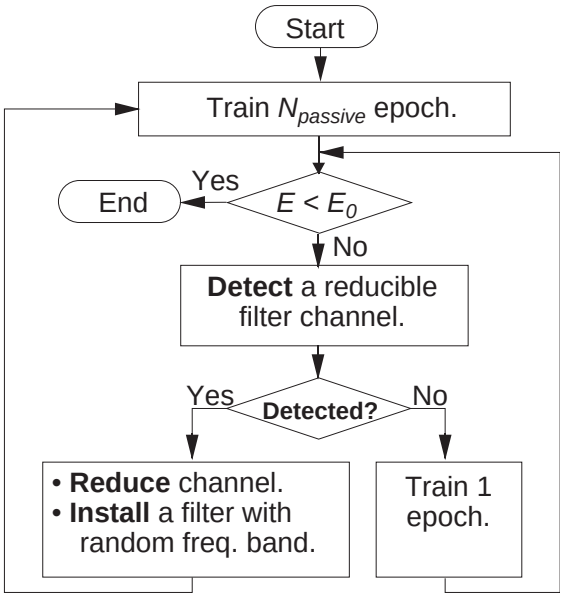


Fig. 5.22. The procedure of training and MA. The detection and reduction methods were varied for comparison of the efficiency of the whole training process.

When the model was altered, no alteration were allowed during the following $N_{passive}$ training epoch to avoid repetitive replacement of the same channel. Two types of KM Net configurations were trained and were tested using four different MA strategies described in the following.

(a) Multichannel filtering : The Gabor filter bands were fixed to the spectral peaks of the textures. Only the connection weights $\{w_{kn}\}$ and $\{w_{nl}\}$ of Fig. 5.5 were trained. In the smaller network with five filters, four filters were tuned to the largest peaks of the four texture classes. The other filter was tuned to the second largest peak of Class 4, which is the most broadband class. In the network with eight filters, first and second largest peaks of four texture classes were characterized. The bandwidth parameters were fixed as, $\sigma_{lx} = \sigma_{ly} = 5.0$ ($l = 1, 2, \ldots, G$) .

(b) KM Net : This is the KM Net with simultaneous BP training of Gabor filter kernels and the connection weights, but not using model switching. Methods (c) ~ (f) in the following employ additional model switching strategies applied to the Gabor filter layer of the KM Net during training.

(c) Detect and remove : Look for a channel j with a minimum σ_j satisfying $\sigma_j < 0.1$. If not found, seek for a pair with a maximum $|r_{ij}|$, satisfying $|r_{ij}| > 0.9$. Remove channel j without weight compensations of Eqs. (3.21) ~ (3.25).

(d) Weight compensation : Same detection criterion as (2), with weight compensations of channel fusion. Fuse channel j with the bias channel when the variance criterion is used. When the similarity criterion is used, fuse channels i and j.

(e) MUB : MS by using the minimum upper bound as map distance. Conditions are, $\varepsilon_{MUB} = 0.1$ and $N_{passive} = 50$. Additionally, a passive epoch will be set to $GN_{passive}$ for the switched channel to avoid repetitive selection, where G denotes the number of Gabor layer units.

(f) EMD : MS by using the effective map distance method. Conditions are, $\varepsilon_{EMD} = 0.02$ and $N_{passive} = 100$.

Methods (a) and (b) are the control cases that do not change the model. Strategies (c) and (d) reproduce the detection and the pruning method employed in [40], where variance and similarity criteria were used separately, whereas in (e) and (f), the proposed unified detection criteria are used.

Table 5.2. Training epochs, classification rates and their standard deviations for two types of KM Nets tried with five different model switching strategies.

Network size (Gabor-hidden-output)		Training epoch	Class. rate	Rate std. dev.
5 - 5 - 4	(1) Mul. ch. filtering	7593	79.0 %	6.6 %
	(2) KM Net (no switch)	5358	81.3 %	6.6 %
	(3) Detect & replace	4287	83.0 %	8.2 %
	(4) Weight Comp.	3855	84.1 %	4.7 %
	(5) MUB	**3827**	**84.7 %**	**5.7 %**
	(6) EMD	**3738**	**86.2 %**	**3.7 %**
8 - 6 - 4	(1) Mul. ch. filtering	4535	81.8 %	6.9 %
	(2) KM Net (no switch)	3374	88.1 %	2.9 %
	(3) Detect & replace	2723	88.6 %	2.9 %
	(4) Weight Comp.	2665	89.1 %	4.0 %
	(5) MUB	**2646**	**89.2 %**	**2.4 %**
	(6) EMD	**2533**	**89.5 %**	**2.7 %**

The networks were trained to gain MSE = 0.002, for unit vector responses assigned to each texture class. After training, the networks were applied to the whole area of the same image of Fig. 5.9, and the classification rates and the standard deviations were obtained. All numbers are averages of 30 trials. For each texture class, 12 training subimages of 32×32[pixel] size were used.

Experimental results are compared in Table 5.2. First, when compared with the conventional multichannel filtering with fixed filter bands, it is clear that the tuning of the Gabor kernels enabled in the training of the KM Net greatly contributes to improve the classification rate. Further, when model switching during training is used, it is observed that detection and replacement of the redundant input channels in (c), and further weight compensations in (d) are both effective to improve the classification rates and the training speeds. The improvement in (c) indicates that providing higher linear independency among the feature channels is beneficial for improvement of generalization ability. The result in (d) shows that preservation of the map while altering the model helps to speed

up the whole training process. Without map preservation, more training epochs will be necessary after each model alteration in order to compensate for the increase of the training error. Although preservation of the map by way of model switching contributes to faster training, an external rule for combining different model switching methods is still necessary, because variance and similarity criteria are used independently in (d).

When the MUB and EMD criteria are used, the classification rate and the speed were further improved for both network configurations, indicating the effectiveness of using the unified detection criterion for less damage to the trained network on switching. In the MUB detection, model switching mainly occurred in the initial phases of the training, as it was predicted in Sec. 3.4. Cases that the training terminated with a pair of Gabor filters extracting an identical frequency band, however, were quite common. In the EMD detection, such a case was scarcely observed, for redundant Gabor units were detected and the model was switched throughout the whole training period.

5.9 Summary

In this chapter, first the scheme of image texture classification by multichannel filtering was reviewed, and the shortcoming of manual feature selection in the conventional method was pointed out. As a means to automate this feature selection process, a novel neural network architecture was introduced. The Kernel Modifying Neural Network (KM Net) which incorporates an array of Gabor bandpass filters and a classifier in a single layered neural network structure, enables to simultaneously tune the filter parameters and the connection weights in the classifier by BP learning for the sake of classification error reduction. Through applications to synthetic and natural texture classification problems, the training rule and the classification ability of the KM Net were confirmed. It proved that, the filter tuning by the training rule could replace the manual feature selection procedure. For the classification of visually different texture classes with identical second order statistical natures, an expansion of the KM Net model was introduced, which extracts the higher-order statistical feature by training. The Bispectral Kernel Modifying Neural Network model, which extracts the localized third-order spectral feature (bispectrum) of the image texture, proved to effectively extract the phase feature in the texture for discrimination. Although the training rule of the KM Net could tune the bands of the Gabor filter array, the same feature extraction efficiency problem arose, as was pointed out in Chapter 2 to be the motive of model switching. At many situations, the filter array bands were tuned to suboptimal frequency band sets. In order to enable an efficient feature extraction in the Gabor layer using a limited number of filters, the model switching by channel fusion was tried during the training. The detection and reinstallation of redundant channels contributed to both the training speed and the classification rate. There was an additional finding that the use of the unified criterion for selection of the switching model, which is based on the map distance defined in Chapter 3, was most effective in the multistrategy switching.

Chapter 6
Defect Classification in Visual Inspection of Semiconductors

6.1 Introduction

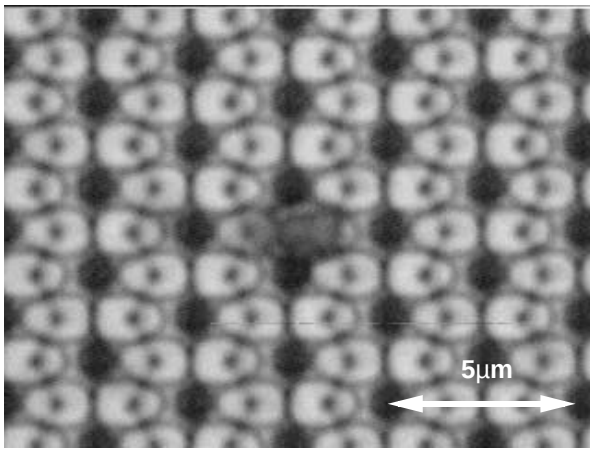


Fig. 6.1. A microscopic image of a defect found on a semiconductor wafer.

Visual inspection of industrial products is an essential stage for quality control. The tasks of inspection which conventionally depended on human experts, are quickly being taken over by machine vision / pattern classification systems. This trend also applies to semiconductor wafer manufacturing processes, where visual inspection plays an important role. The semiconductor chips to be inspected are electrical circuit patterns constructed in a layered structure, on a semiconductor substrate. Since the device geometry of sub-micrometer scale are becoming commonplace, all images used for in-line visual inspection are collected using optical microscopes.

The disorders found on the wafer surface, such as the one shown in Fig. 6.1, are commonly referred to as *defects*. The motive for defect classification is to find out the process stages and the reasons that are causing them. Early detection of the sources of defects are essential in order to maintain high product yield and quality.

In the majority of semiconductor fabs, the visual inspection process of the wafer surface still depend on manual review by human experts. Since the inspection task requires extreme concentration, the time that an inspector can continue the task is quite limited, and still, it tends to be time consuming and inaccurate. The decision instability of an inspector can be quite large against various defect classes, and each inspector relies on different features and strategies [79]. It is reported that the classification accuracies are typically 60~80% [80]. If this stage of visual inspection could be automated, it will greatly contribute to enhance the productivity of the semiconductor fab. However,

the appearances of a known class of defects will typically have a very wide variety in size, shape and color. Also, it is difficult to obtain a training set of a sufficient size that represents the defect class. These conditions make it hard to determine the effective image features. Therefore, only few automatic defect classification systems (ADC) are currently in operation at the production line.

In selecting the feature and determining the strategy of classification, the straightforward way would be to mimic what the human experts do. Information from the expert inspectors are essential, however, the features and the strategies employed by the experts are usually hard to code explicitly. Expert inspectors tend to use multiple strategies, and evaluate the possibilities to reach the final decision in parallel. If we could determine the combinations of several promising defect features through interviews, well known techniques such as vector quantization [23][24], subspace methods based on projection filters, application of the Bayes rule to estimated class probability densities [3] or trainable mapping neural networks[5][23][24] can be used for the classification task.

Semiconductor ADC systems have been developed and reported by several research groups and manufacturers [78][79][80]. In [79], one of the few commercially available ADC systems is introduced. There, die-to-die comparison and single image analysis are used for defect detection in logic products with random patterns, and memory products with repetitive patterns, respectively. The system extracts numerous spatial and textural features from the defect region and applies a set of user-defined fuzzy predicates for defining the defect classes. The paper points out the necessity to train the system for each product and process layer. The approach taken here shares some common approaches and features such as the die-to-die comparison and the need for layer specific construction of the classifiers. However, by using trainable neural networks for classification of the feature vectors, use of both explicit and implicit correlations between the perceptible features and the defect classes are intended.

The task, as seen as a pattern classification problem, has several restrictions inherent to the particular problem. First, the designer does not have the freedom of collecting a sufficient number of, or an appropriate selection of training images. In spite of such a limitation in the training conditions, the dimensionality of the feature space tends to be high, because the decision tends to be dependent to many features in the image. Second, the requirement to the system is to classify the known defect classes without fail and do not make *wild* guesses against unfamiliar types of defects. Such images should be pointed out as unclassifiable and be left for the human expert to see. It is known that problems of the kind can be handled by algorithms whose decisions are based on the distances from the class prototypes, such as the *k*-means algorithm [4], leaning vector quantization and self-organizing feature maps [29]. Here, we will employ an alternative approach, of constructing a class border using localized nonlinear discriminants, implemented in a Hyperellipsoid Clustering Network (HCN) introduced in Chapter 4. In models such as HCNs, it is important to construct a membership function that approximates the class distribution correctly, while maintaining the simplicity.

In this chapter, a system for automatic semiconductor defect classification (ADC) using neural

network classifiers will be introduced. In Sec. 6.2, the defect classes will be defined, and the methods for the defect detection, defect region determination and extraction of the features will be addressed. In Sec. 6.3, MLP networks with predetermined models and HCN networks trained with BP with MS will be applied to the defect classification problems. Several issues and further aims in development of the ADC system will be discussed in Sec. 6.4.

6.2 Automatic defect classifier system (ADC)

6.2.1 Defect classes

The wafer fabrication processes in manufacturing semiconductor products involve several cycles of (a) film deposition, (b) photoresist coating, pattern alignment and developing, (c) pattern etching, and (d) cleaning. There are several types of visible defects occurring in each of these stages that can causing defective products. Among them are, objects of foreign origin, *footprints* of such particles resulting in incorrectly etched patterns, and pattern anomaly due to imperfect exposure or etching processes. In order to maintain high product yield and quality, quick and accurate detection of the causes of the defects are very important. The first step for locating the cause is to classify the types of the defects by visual inspection.

The definition of the defect classes vary according to the manufacturers. However, it is common that the goal classes are combinations of *physical* classes, and *contextual* classes. For example, a physical defect of an incorrectly etched pattern commonly made by an external particle, can occur anywhere on the die. Contextually, it can be of a “short circuit class” if it happens to bridge the pattern. In this case it will be a killer-defect. On the other hand, it can also be of a harmless “cosmetic class” that may be ignorable, if it is found on patternless places or marginal areas.

Here, identification of the physical defect classes which provides most information for locating the cause of the defects, will be tried. Nevertheless, contextual defects are also important since they will help to determine the usability of the particular die as in the above example. Here, further classification to the contextual subclasses will be left for the feature enhancement.

The physical defect classes dealt with and their common appearances will be listed in the following :

(a) Foreign objects (FO)

This class includes defects such that external objects are found on the wafer. If a cleaning process is

scheduled right after the inspection, it is likely that they will be washed away. However, the finding may imply that the source is at the preceding stages, close to the current inspection stage. Defects of FO class tend to appear as small and dark colored regions, typically in near-circular shape.

(b) Embedded foreign objects (EO)

This is the class of defects where one or more processed film layers have been stacked over a foreign object. It is highly likely that the EO class defect is a killer-defect, since it corrupts the patterns formed over it by forcing them to make a heap. They can also cause disturbances in the inter-layer electrical contacts. Defects of this class indicate that the source is more than one wafer process cycle previous to the current inspection stage. Defects of EO class appear slightly larger and irregular-shaped when compared with those of the FO class, because the patterns of the heaped area in the covering layers are deformed by the embedded object. In addition to the characteristic dark color of the particle itself, other colors can be observed as well. Defects of FO and EO classes can appear quite similar, and are sometimes hard to distinguish even for an expert.

(c) Pattern failure (PF)

This class covers all kinds of defects that have pattern deformations without any existence of external objects. The PF class defect can occur in many ways. Most commonly, they are caused by external objects blocking the photo-exposure process or the etching process. Later on, the external objects are washed away, and their footprints become PF class defects. Defects of this class can also be caused by insufficient exposure or etching. Thus they can have a wide variety of size and shape. Since the defect is usually an extra region or a lack in the pattern of a layer, the color of the defect region tends to be one of those observed in the normal patterns.

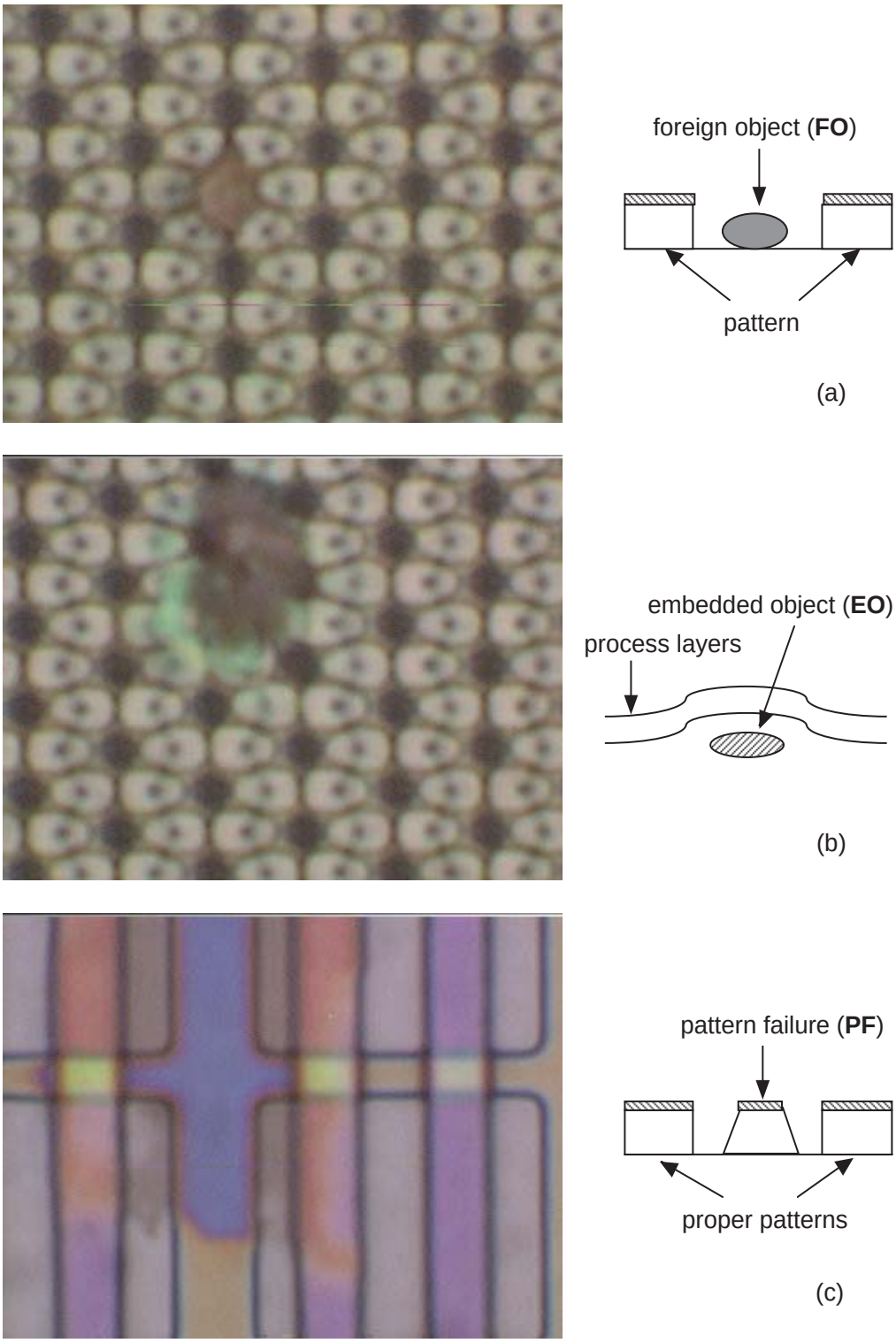


Fig. 6.2. Example images and their cross sections illustrated. (a) Foreign object class (FO). (b) Embedded object class (EO) . (c) Pattern failure class (PF).

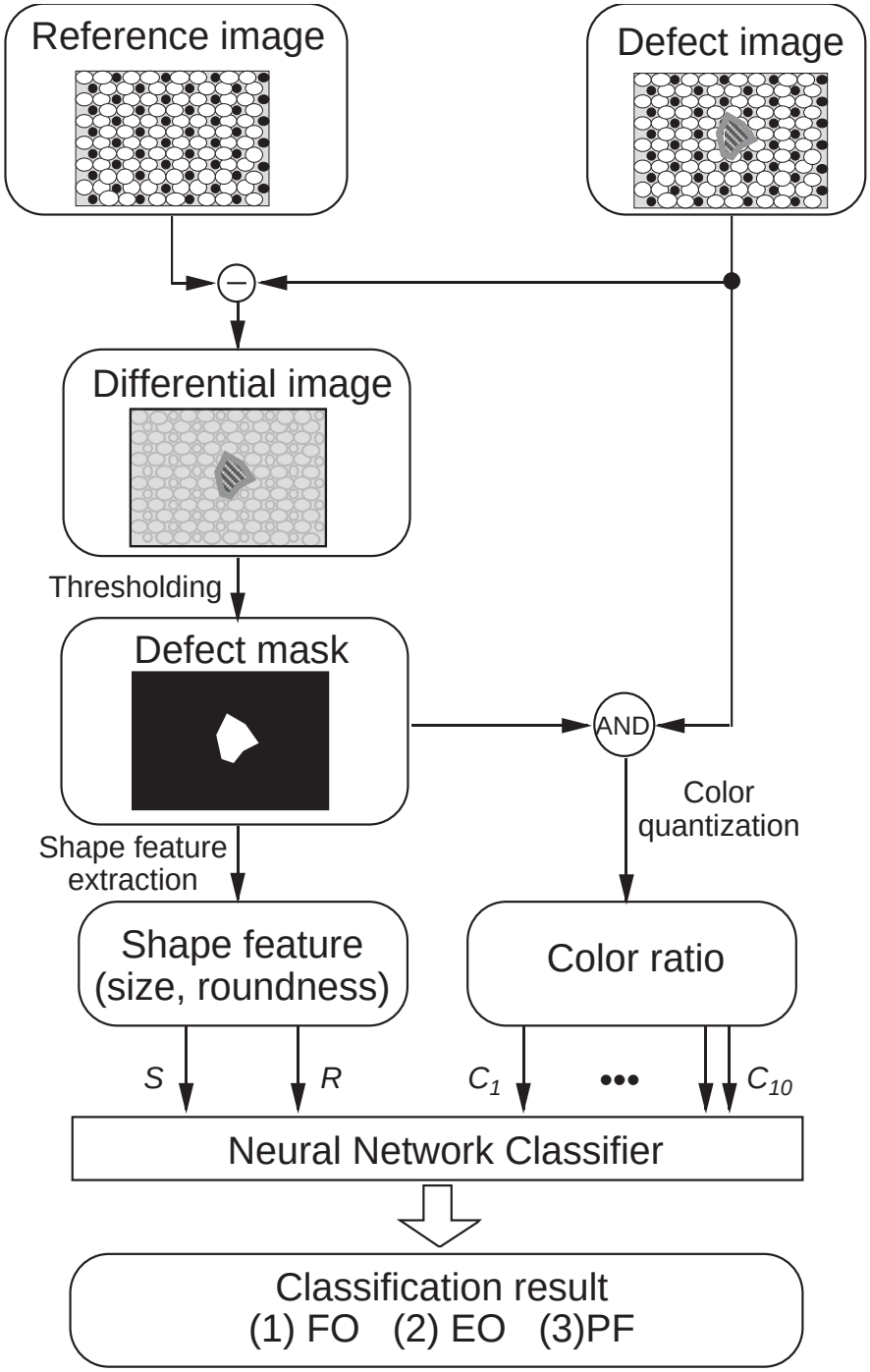


Fig. 6.3. The flow of data in the ADC system.

6.2.2 Detection of defects

The flow of the data in the proposed ADC system is illustrated in Fig. 6.3. Initially, the defects on the patterned wafers are detected by an inspection system. The optical detector scans the surface and compares the detection signal with that of a defectless reference die or an adjacent die on the wafer.

Collection of the defect images is carried out by a *review station* with a two-dimensional moving stage and a microscopic machine vision system. The *defect image* at the detected die coordinate will be recorded, together with the defectless *reference image* from the adjacent die at the same die coordinate. The images are recorded as color bitmap data of 24 (bit/pixel) depth. These image pairs are used for determination of the defect area (*re-detection*) and feature extraction for classification. The size of the images used in this work are 320 (pixel) in width and 240(pixel) in height.

6.2.3 Determination of the defect region

The disordered area in the defect image is determined by the following steps. First, the reference image is subtracted from the defect image. On subtraction, the positional error of the moving stage will be compensated by adjusting the relative positions to where the cross-correlations of the two images are maximum. Further, the intensity information are extracted and rescaled to make a grayscale image of 8 (bit/pixel) depth. Therefore, the normal region will have a level near 127, whereas the graylevel of the defect region will be either higher or lower. In order to make the mask of the defect region, the grayscale level at each pixel is thresholded by the following function of

$$T(p;\,\theta) = \begin{cases} 0\ (\text{off}) & \text{if } (127 - \theta < p < 127 + \theta) \\ 1\ (\text{on}) & \text{otherwise} \end{cases} \tag{6.1}$$

where p is the pixel graylevel and θ is the half width of the graylevel margin acceptable as the normal region. The value of θ determines the detection sensitivity. If θ is small, a larger area including the defect will be captured. Also many pixels in the normal region will be detected due to pattern variations that are practically tolerable. If θ is set too large, small defects can be missed. In this work, the value of θ was empirically set to 20. Even at this setting, some speckle-like signals from normal areas still prevailed. Therefore, only the largest contiguous *on* region was allowed to remain, whereas the others were set to *off* in order to limit the masking to the target defect region. The resulting binary level image was used as the defect region mask.

6.2.4 Feature extraction

For classifying the defects to the three classes of Sec. 6.2.1, the following features were used.

A. Size *S*

The size of a defect can provide some hints to the class information especially for separating FO and EO classes. When defects of FO and EO classes are caused by particles of the same size and shape, the observed region of the EO class defect will always turn out to be larger. This is because the layers covering the EO defect are disturbed by the particle, and the area surrounding the particle will appear in different colors. In the data set we used, the size of the defect region in pixels N_D ranged from 10 to 10^4. However, most of them were within the $10^2 \sim 10^3$ range. In order to scale N_D to a unit range without losing sensitivity to the majority of the data, the size feature S was defined as,

$$S = \frac{1}{a}\log_{10} N_D \ , \qquad (6.2)$$

where a is an additional scaling parameter to keep S within the (0, 1) range. Since the maximum defect pixel counts could reach 320×240 = 76800 (pixels), parameter a was set to 5.0 in the experiments.

B. Roundness *R*

The roundness of the defect region also helps to separate FO and EO classes. As noted in Sec. 6.2.1, many of the foreign objects appear in a smooth circular shape. When they are embedded in film layers, the defect region including the foreign object and the disturbed patterns over it will be observed in a very irregular shape. This difference can be characterized by the roundness of the defect region as defined below.

A common definition of the roundness of a region is

$$R_0 = 4\pi\,(\text{area})/(\text{perimeter})^2 \qquad (0 < R_0 < 1)\ . \qquad (6.3)$$

When using this measure in a digital image, the numbers of pixels in the region (N_D) and the region border (N_B) can be used as the area and the perimeter, respectively. The roundness calculated this way will asymptotically approach the true roundness as N_D grows. However, when the region is small, R_0 can exceed 1.0 due to discretization errors. In order to avoid such cases by scaling, the scaled definition of the region roundness R used in this work was,

$$R = \frac{4\pi\,(\text{area})}{(\text{perimeter})^2} \cong \frac{4\pi N}{(1.5M)^2} \quad (0 < R < 1)\,, \qquad (6.4)$$

where b is the scaling parameter for keep R within (0, 1) range. In the experiments, b was empirically set to 1.5.

C. Quantized color ratios *C*

The color of the defect provides rich information to the human experts. Experts tend to *name* colors to several classes and deduce the defect class in conjunction with the other features. Therefore, it is important to know what type of color is dominant in the defect region.

Among the various colors appearing in the defects, first, there can be groups of colors that have universal correlation with the defect types, such as the dark colors of external objects. We will call these group of predefineable colors as *defect colors*. Defect colors tend to dominate in the FO class defects. Second, there are colors whose interpretation can vary by the layer which is currently inspected. For example, if a particular color is common as the color of the substrate in a layer, a defect region with the same color dominating may imply a break of a pattern, which belongs to the PF class. If the same color is scarcely observed in the normal regions of the layer, the color in the defect may imply an existence of an embedded foreign object, which is of the EO class. In usual products, there were not much variation in the colors that appear in the microscopic image of the normal region. Usually less than ten different classes of colors were perceptible if a person tried to count them. The

group of such typical colors will be called as *normal colors* of a layer. The remaining set of other colors will be named *unfamiliar colors*. Defects of PF class tend to appear mostly in normal colors. However, abnormalities in the film thickness which also belongs to the PF class can manifest itself in a totally unfamiliar color. Defects of EO class tend to have various combinations of defect, normal and unfamiliar colors.

As the existence of a color imply different conclusions layer by layer, it would be hard to design a unified defect classification system for use in all the layers of any product. In this work, the color feature characterization which is a process to determine the prototypes of normal and abnormal colors, will be conducted for each layer and product.

For implementing the *naming* process of the colors, and to convert the pixel by pixel color information to a collective feature of the defect region, the number of the colors were reduced to a lower dimension by using a quantization algorithm. After quantization, the ratio of the quantized colors in the defect region were used as the color feature.

The color reduction algorithm used in this work is the Median Cut Algorithm [15]. The outline of the Median Cut Algorithm for reducing the colors to 2^{γ} prototypes is as follows:

1. Scatter all the pixels in the image to the three dimensional *RGB* space according to their colors.
2. Set i =1
3. Set a *box* in the *RGB* space, each side parallel to one of the *RGB* axes, and tightly enclosing the scattered points. The points in a box and their cardinality will be called the *member pixels* and the *population* of the box, respectively.
4. For each box, find the index among *R*, *G* and *B* having the largest variation, and sort the member pixels along the index.
5. Segregate each box into two boxes at the median point.
6. If $i < \gamma$, increment i and go to 4.
   Otherwise, for each box, set the color of the member pixels to the color at the box center.
7. End.

Seen as a vector quantization algorithm, the center and the enclosed region of the box will be the prototype vector and the neighbor region, respectively.

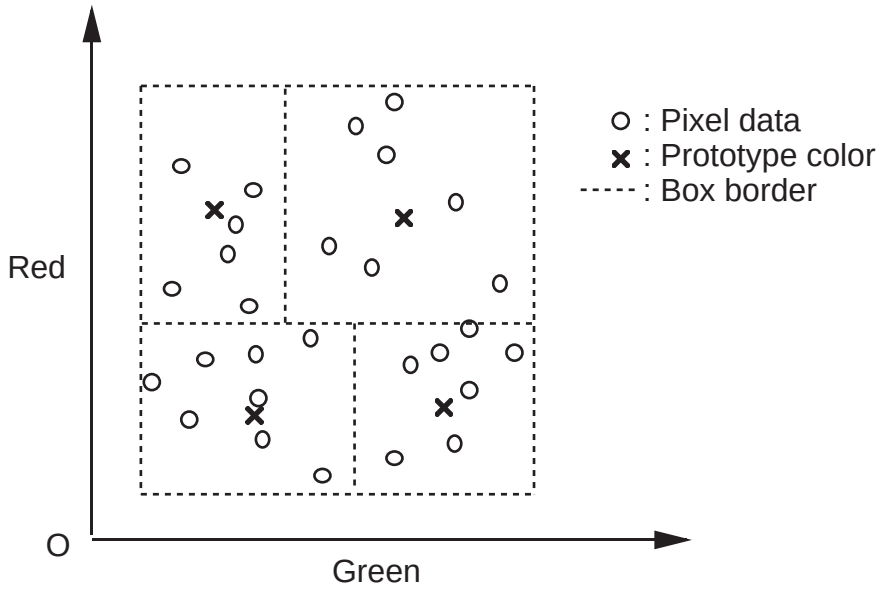


Fig. 6.4. The scheme of color quantization using the Median Cut Algorithm, illustrated using the two axes among the RGB space.

On determining the normal colors and their box neighborhoods for a layer, the Median Cut Algorithm was applied to a collection of reference images. The same algorithm can be applied to a collection of pixels from the defects of a particular type, to register the defect colors. However, since the two color groups are generated independently, there can be an overlapping region among the boxes in the *RGB* space. Post-processing to reduce the overlapping region by manually trimming the box region may be necessary. The prototypes and the box regions of the normal colors and the defect colors were kept to be used for characterizing the new defect images from the same layer.

On characterization of a new defect image, the color of each pixel in the defect region was projected in the *RGB* space to evaluate the membership to the prototype colors; namely to see which prepared boxes of the defect and the normal colors it fell into. For each color c, the ratio feature C_c of a defect was defined as

$$C_c = \frac{\text{(Population of color box } c\text{)}}{\text{(Total number of pixels in the defect)}}$$
$$(0 < C_c < 1,\ c = 1,\ 2,\ \ldots,\ (K_{normal} + K_{defect} + 1)), \tag{6.5}$$

with K_{normal} and K_{defect} being the number of normal and defect color prototypes (boxes), respectively. The merit of using the Median Cut Algorithm for quantization is in that it can point out a color which is unfamiliar to the quantizer; not falling into any of the prepared boxes. In the layer manufacturing process, such colors can be caused by abrupt changes in the film thickness. Since it is hard to make a general connection between such unfamiliar color groups and the defect types, the population of the pixels falling out of all the boxes were counted, to make a single channel of *unfamiliar color* class ratio. There are other merits in using the algorithm, such that the prototypes

and their neighborhoods are reproducible since it does not rely on any random factors. Also, the cluster populations can be kept even as possible. Both of these features do not apply to the case of many common clustering algorithms such as the *k*-means clustering.

6.2.5 Neural network classifiers

Two types of neural networks were used for classification of the feature vectors. The first type is the conventional Multilaer Perceptron (MLP) network, trained by batched BP. The second type is the Hyperellipsoid Clustering Network (HCN) introduced in Chapter 4. It is assumed that the network model characterizing a localized data cluster is suitable for this application. In Fig. 6.5, the distribution of the feature vectors of the defect data set used in the experiments is shown. The data points are projected to a feature subspace spanned by two among the twelve features. Although the classes are not clearly separable within the subspace, it is clear that each class forms a locally concentrated cluster, leaving space without any input data. In contrast ot the MLP, recognition of the rejection class will be attempted in HCN.

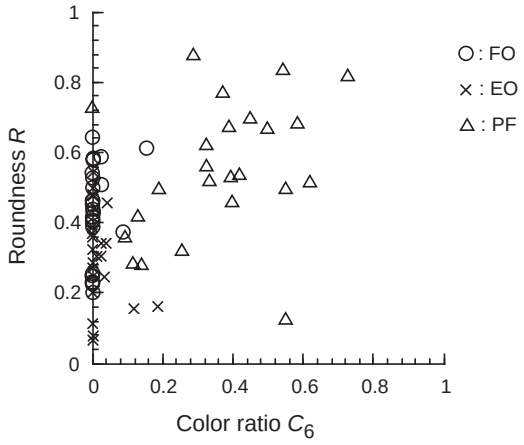


Fig. 6.5. Distribution of the feature vectors of the data set used in the experiments. The 12 dimensional vectors are projected to a 2 dimensional subspace spanned by the pixel ratio of the sixth prototype color C_6 and the roundness R.

6.3 Experiment

A collection of defect images obtained from the same process layer of a product was used for evaluating the classification system. The set consisted of 33 FO class, 36 EO class and 24 PF class images. Defect images with typical appearances for the defect classes were manually selected to form the training set. The remains of the image data were used as the test set for evaluation except for experiment C. The class information for all the images were provided by an expert inspector. Besides the classification rates, the membership thresholding in HCN will also be tested.

The features used for classification were the size S, roundness R and the color ratio C_c (c =

1, ..., 10). The ten colors consisted of eight normal colors determined by applying the Median Cut Algorithm to the reference images, one defect color for typically dark external objects and another channel representing the unfamiliar colors that do not fall inside the box-shaped neighborhood of the known colors. Therefore, the total number of feature channels was twelve. The box neighbor region of the defect color was manually tuned to reduce the overlap between the boxes of the normal colors. When a pixel fell into the overlapping region, it was counted as a population of the defect color only.

A. Classification with a small training set

First, classification of the images using a very small training set was tried. This is a simulation of an extreme case when only very small number of defect images are available. Only two samples were used for each class to train the networks. The average classification rates for 20 training trials each are shown in Tables 6.1 and 6.2, for the MLP and the HCN networks, respectively. The averaged confusion matrix counting the images of the row class classified to the column class is provided as well for error analysis. The configurations of the networks and the training conditions were as follows.

MLP

Units (input, hidden, output)	(12, 3, 3) .
Training gain	$\eta = 2.0$.
Momentum gain	$\alpha = 0.9$.
Initial weights	Selected randomly from (–0.1, 0.1) .
Error criterion to stop training	0.001 (average/output unit) .

HCN

Units (input, hidden, output)	(12, 3, 3) .
Training gains	$\eta_\mu = 0.001$, $\eta_H = 0.5$, $\eta_r = 0.01$, $\eta_w = 0.1$.
Momentum term gains	$\alpha_\mu = \alpha_H = \alpha_r = 0.9$.
Penalty term gains	$\phi_H = 5.0\times10^{-6}$, $\phi_r = 1.0\times10^{-6}$.
Membership threshold	$\theta_m = 0.5$.
Initial parameters are randomly selected from the following sections except for $\boldsymbol{\mu}_n$.	
	w_{kn} : (–0.1, 0.1) .
	H_{nst} : (diagonal element $s = t$) (2.5, 3.5) * .
	H_{nst} : (non-diagonal element $s \neq t$) (–0.1, 0.1) * .
	r_n : (0.5, 1.0) .
	$\boldsymbol{\mu}_n$: (k-means clustering) .
Error criterion to stop training	0.04 (average/output unit) .

* Initial weight matrix $\boldsymbol{H}_n$ is set to start the training from a near-hypersphere status.

It is noteworthy in Tables 6.1 and 6.2 that the classification rates are high in spite of the small number of training images. This fact justifies the selection of the feature elements and indicate good separability of the clusters within the feature space. However, the relatively high confusion between the FO class and the EO class reflects the difficulty of discriminating the two, even to the eye of an expert (and possible confusions in the oracle information as well). The performance for both classifiers were almost the same when no membership thresholding was used in the HCN. When the thresholding was used with θ_m set to 0.5, most of the nondiagonal elements besides the FO-EO confusion were suppressed.

Table 6.1. The classification rate and the confusion matrix for the MLP classifier trained with a small training set.

True \ Estimation	FO	EO	PF	Correct (%)	Error (%)
Foreign Object (FO) 31 test sets	18.1	12.9	0.0	58.4	41.6
Embedded Object (EO) 34 test sets	13.0	20.0	1.1	58.9	41.1
Pattern Failure (PF) 22 test sets	4.2	0.5	17.6	80.0	20.0
Average rates (weighted)				64.1	35.9

Table 6.2. The classification rate and the confusion matrix for the HCN classifier trained with a small training set. The numbers in bold typefaces are for the cases when membership thresholding was applied.

True \ Estimation	FO	EO	PF	Unknown	Correct (%)	Error (%)
Foreign Object (FO) 31 test sets	15.8 **12.1**	14.4 **10.2**	0.8 **0.0**	0.0 **8.7**	51.0 **38.7**	49.0 **32.9**
Embedded Object (EO) 34 test sets	10.2 **4.7**	20.1 **13.3**	3.8 **0.1**	0.0 **16.1**	59.1 **39.1**	41.0 **14.1**
Pattern Failure (PF) 22 test sets	1.1 **0.1**	1.2 **0.7**	19.8 **14.3**	0.0 **6.9**	90.0 **65.0**	10.2 **3.6**
Average rates (weighted)					64.0 **45.5**	36.1 **18.1**

B. Classification with a larger training set

Next, the number of training pairs was increased to cover more intra-class variation. The numbers of the training samples were (FO, EO, PF) = (12, 11, 5).

In determination of the model of HCN, the *k*-means initialization of the hidden layer kernels and gradual reduction of the hidden layer by fusion were tried. Since it was found that best generalization could be achieved with the model with 5 hidden units, this configuration was selected. On training, MS strategy of limited model scheduling was taken. All the settings were the same as in the former experiment, except for the network size. The network size were as follows

MLP

Units (input, hidden, output) (12, 7, 3) .

HCN

Units (input, hidden, output) (12, 5, 3) .

The rates of the classification are shown in Tables 6.3 and 6.4. The rates were improved with less EO-FO confusion. When membership thresholding is turned off, the HCN based on Mahalanobis distance measure which was more suited to the problem, outperformed the MLP employing hyperplane class borders. When θ_m was set to 0.1, the error was suppressed to the same rate as observed in Experiment A, with approximately 30% improvement in the correct classification rate.

Table 6.3. The classification rate and the confusion matrix for the MLP classifier trained with a larger training set.

True \ Estimation	FO	EO	PF	Correct (%)	Error (%)
Foreign Object (FO) 21 test sets	16.1	5.0	0.0	76.7	23.3
Embedded Object (EO) 25 test sets	8.1	16.7	0.4	66.8	33.2
Pattern Failure (PF) 19 test sets	2.0	0.2	16.9	88.9	11.1
Average rates (weighted)				76.4	23.5

Table 6.4. The classification rate and the confusion matrix for the HCN classifier trained with a larger training set. The numbers in bold typefaces are for the cases when membership thresholding was applied.

True \ Estimation	FO	EO	PF	Unknown	Correct (%)	Error (%)
Foreign Object (FO) 21 test sets	15.3 **15.3**	5.1 **5.1**	0.7 **0.7**	0.0 **0.0**	72.9 **72.9**	21.1 **21.1**
Embedded Object (EO) 25 test sets	6.1 **6.0**	17.9 **17.3**	1.1 **0.1**	0.0 **1.7**	71.6 **69.2**	28.4 **24.4**
Pattern Failure (PF) 19 test sets	1.4 **1.4**	0.2 **0.0**	17.5 **16.4**	0.0 **1.3**	92.1 **86.3**	7.9 **7.4**
Average rates (weighted)					78.0 **75.4**	20.0 **18.4**

C. Leave-one-out evaluation with HCN using MS

Highest classfication rates were obtained by using HCN with MS evaluated in the leave-one-out method. Here, HCN network initialized by placing all the kernels at the training inputs were trained using MS developed in Chapter 4. The model typically converged to models with hidden layer units in the range of 9 to 14. The membership threshold θ_m was set to 0.5. By using rejection, it is found that the non-diagonal elements (errors) in the confusion matrix could be drastically reduced.

Table 6.5. The classification rate and the confusion matrix for the HCN classifier trained with a larger training set. The numbers in bold typefaces are for the cases when membership thresholding was applied.

True \ Estimation	FO	EO	PF	Unknown	Correct (%)	Error (%)
Foreign Object (FO)	32 **32**	1 **0**	0 **0**	0 **1**	97.0 **97.0**	3.0 **0.0**
Embedded Object (EO)	2 **1**	32 **30**	2 **0**	0 **5**	88.9 **83.3**	11.1 **2.8**
Pattern Failure (PF)	2 **0**	0 **0**	22 **21**	0 **3**	91.7 **87.5**	8.3 **0.0**
Average rates (weighted)					92.5 **89.2**	7.5 **1.1**

6.4 Discussion

In [79], it is pointed out that the rate of re-detection which depend on the accuracy of the moving stage and the auto-focusing system of the review station is also an important factor affecting the overall performance. Since all the numbers reported here assume that the defects were successfully re-detected, the overall classification rate can be lower according to the precision of the inspection hardware. Additionally, there were several cases that selecting the largest contiguous region failed to capture the whole defect region. When the defect is of the EO class, line pattern can cross over them, which separates the region to non-contiguous parts. Preparation of an additional preprocessing stage for correct extraction of the defect region is under development.

A recent paper [80] introduces an ADC system in use for repetitive (memory) patterns. Phenomenal defect classification is dealt with by extracting various defect features including color, size, texture, etc. In addition, it classifies the contextual defect class addressed in Sec. 6.2.1, by using the relation between the image pixel label and the defect pixel label. The former label represents the local identity (such as compositions) and the latter is determined by a distance-based classification among the known label identity feature spaces.

The number of features that were used in this work is quite limited when compared with the other works. What we intended here is to find out whether the essential features that experts recognize to be beneficial can be appropriately coded to mimic the expert. This goal was successfully achieved in view of the separability of the classes and even the inheritances of the traits seen in the confusion of the FO and EO classes. At the same time, rejection of unknown inputs by the “tight clustering” and membership thresholding in the HCN was also successfully implemented. Therefore, we can now proceed to add other defect features. As the dimension of the feature increases, there will be more freedom in the feature space, thereby opening more “empty space” for the training set of the predetermined classes. The HCN model, which may be seen as a flexible compromise between the classification methods based on discrimination borders (e.g. MLP[5]) and nonparametric methods based on distances from examples (e.g. PNN[22]) should perform better in such cases.

Ability of incremental learning, to increase the training set size without starting the training all over again, should be important when the system is to be used in the fab. The idea of Growing RBF nets in [21][25] may be applicable, to train or split the nearest kernel function which consists the membership function of the class of the added training pair. Alternatively, assigning an independent small kernel to the new training input as in the PNN case, and to further seek the possibility of unifying it to an existing hyperellipsoid in the style of proposed model switching may prove to be more suitable to limit the distribution of the membership function.

6.5 Summary

In this chapter, A system for semiconductor defect classification using neural network classifiers was introduced. The outline of the ADC system, including the procedure of defect re-detection, the types of features based on the experts' knowledge and their extraction process were elucidated. As for the classifier, the Hyperellipsoid Clustering Network (HCN) architecture with membership thresholding and training with model switching was used in comparison with the conventional MLP network.

In the experiments, the HCN model proved to be more suitable for a flexible multidimensional clustering with a smaller training set, together with a tunable margin of membership thresholding for rejection of unfamiliar inputs. It was also found that the use of MS during the training of HCN could be used for efficient generation of the membership functions, in place of external initialization processes such as the *k*-means algorithm. By using the HCN classifier, a classification rate comparative to those of the human experts could be achieved.

Chapter 7
Conclusion

In this thesis, a novel scheme of layered neural network training called Model Switching (MS) was introduced. Model switching being a scheme for determination of the occasion of model alteration in accordance with the fitness of the candidates, opened up a novel search path of the map and the model in the stepwise training process such as the BP. For controlling the MS events, MS Index which reflects both the progress of training in the immediate map, and the fitness of the new map was introduced. It was shown that MS by MS Index could be applied to wide range of layered network types, allowing efficient training, small model implementation and higher generalization ability. Among them were, pruning and self model adjustment of the hidden layer in MLPs and RBF networks, network model adjustment in a network in which first and second order discriminants co-reside, and fusion and re-installation of the feature extracting filters in a special network for image texture classification.

In Chapter 2, the general idea of an operation named *Model Switching* (MS) for enabling the simultaneous determination of the occasion, the model and the initial map was introduced. After giving the definition of MS, a method of MS consulting an index named the *Model Switching Index* (MS Index) will be introduced. The MS Index is defined as a function of the Stress Index reflecting the training progress of the immediate model, and the Fitness Index reflecting the suitability of the new initial map candidate at each moment of training. Then the variations in the scheduling of MS, and the types of Fitness Index were introduced.

In Chapter 3, BP training with MS was introduced to MLPs. As the Fitness Index for selecting the initial maps, degree of map inheritance was evaluated. First, a method of measuring the degree of map inheritance, named *map distance* was defined. Further, the Fitness Index using the *map distance* measure was defined for use in the network training with MS. Then, newly proposed methods for model alteration, named *unit fusion*, *splitting* and *installation* were introduced. In order to simplify the map distance evaluation process, an approximate method for estimating the map distance caused by the unit fusion was introduced using the theorems giving the upper bounds of the map distances. The proposed methods for MA was applied to network pruning. The unit fusion method was compared with several well known pruning methods. In the experiments, the superiority of the unit fusion method in map preservation was shown. Then, the proposed MS according to MS Index preferring map preserving candidates was applied to the BP training. There, it was shown that the whole training process of obtaining a small trained network, will be much efficient when MS by MS Index was used, in comparison with the conventional MA strategies. Further, the relation between the generalization ability of the network and the selection of the initial map in the new model using the proposed MA methods and map distance, was discussed. There, it was shown that altering the

model by unit fusion has a positive effect to improving the generalization ability of the network. As a variant of the proposed training method by MS, a method which allows a smooth change in the model by controlling the linear independence of the responses of the hidden layer units was introduced. There, it was shown that the training process will smoothly converge to a small network attaining the target training error.

In Chapter 4, implementation of MS to the training process of RBF networks and hybrid networks were discussed. There, in addition to the map distance, the fitness index which also evaluates the nature of the class discrimination borders determined by the initial map in the new model was used.First, the Hyperellipsoid Clustering Network (HCN) which is an RBF type network, was introduced, aimed at efficient characterization of the localized clusters and recognition of the rejection class. It was shown that a recognition system can be constructed, which points out the inputs that are distant from known training sets to be originating from an unknown class, in a small model having far less hidden layer units in comparison to the size of the training set. Next, use of MS in the training process of a hybrid network with both first and second order hidden units were attempted. By combining the method with unit fusion and installation methods introduced in Chapter 3, an algorithm for BP training with model switching for the hybrid network was introduced. The algorithm and the network was applied to an artificial classification problem requiring first and second order class borders, and it proved that better generalization could be achieved owing to an efficient characterization of the classification borders when compared with networks with uniform hidden layer unit types.

In Chapter 5, the Kernel Modifying Neural Network (KM Net) model for image texture classification, which incorporates the filter array for feature extraction in a single layered network structure, was introduced. By training the filters together with the classification layers, it was shown that the classification rates could be improved in comparison to the conventional multichannel filtering methods with fixed filters. The applicability of the network to various texture classification problems were shown. For compensating the ability of the network for extraction of higher-order features, the Bispectral Kernel Modifying Neural Network for higher-order statistical feature extraction was introduced. There, its ability to utilize the phase information among the frequency for classification was verified. MS was applied to the filter array in the KM Net in the style of fusion and re-installation for efficient feature selection. It was shown that by applying the MS method introduced in Chapter 3, both the training speed and the generalization ability could be improved.

In Chapter 6, a system for semiconductor defect classification was introduced. The details of the classification system including defect classes, methods for defect detection and the extracted features were delineated. In the classification experiment, HCN with MS introduced in Chapter 4 was used. There, it was pointed out that MS contributes to drastic simplification of the model and adequate recognition of unfamiliar inputs, thereby reducing the error rate. It was also shown that a classification rate comparative to those of the human experts could be achieved.

Appendix A
Modification of the Gabor kernel by the Kernel Modifying Network's training.

Modification measures to the kernel parameters (Eqs. (5.9) and (5.10)) will be derived here. From Eq. (5.1), the output of the Gabor layer unit can be written as

$$g_{lm} = A_l\sqrt{D_{lm}} \tag{A.1}$$

where

$$A_l = \frac{A}{2\pi\sigma_{lx}\sigma_{ly}}, \tag{A.2}$$

$$A_l = \frac{A}{2\pi\sigma_{lx}\sigma_{ly}} D_{lm} = (\sum_x \sum_y C_{lm})^2 + (\sum_x \sum_y S_{lm})^2, \tag{A.3}$$

$$C_{lm} = i_m(x, y)\exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_{lx}^2} + \frac{y^2}{\sigma_{ly}^2}\right)\right\}\cos(\omega_{lx}x + \omega_{ly}y) \tag{A.4}$$

and

$$S_{lm} = i_m(x, y)\exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_{lx}^2} + \frac{y^2}{\sigma_{ly}^2}\right)\right\}\sin(\omega_{lx}x + \omega_{ly}y)\,. \tag{A.5}$$

The modification to the center frequencies in each training epoch can be written as,

$$\Delta\omega_{ls} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial\omega_{ls}(t)} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial g_{lm}}\frac{\partial g_{lm}}{\partial\omega_{ls}}\,. \tag{A.6}$$

The first partial derivative inside the summation of the right hand side of Eq. (A.6) can be obtained from the classification error backpropagated from the hidden layer as [5],

$$\frac{\partial E_m}{\partial g_{lm}} = \sum_{n=1}^{N}\delta_{nm}w_{nl}\,. \tag{A.7}$$

The second partial derivative can be further resolved as

$$\begin{aligned}\frac{\partial g_{lm}}{\partial\omega_{ls}} &= \frac{dg_{lm}}{dD_{lm}}\frac{\partial D_{lm}}{\partial\omega_{ls}} = \frac{A_l^2}{2g_{lm}}\frac{\partial D_{lm}}{\partial\omega_{ls}} \\ &= \frac{A_l^2}{g_{lm}}\left\{(\sum_x\sum_y C_{lm})(\sum_x\sum_y(-sS_{lm})) + (\sum_x\sum_y S_{lm})(\sum_x\sum_y(-sC_{lm}))\right\}.\end{aligned} \tag{A.8}$$

Therefore, combining Eqs. (A.6), (A.7) and (A.8), it can be written as,

$$\Delta\omega_{ls} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\left[(\sum_{n=1}^{N}\delta_{nm}w_{nl})\frac{A_l^2}{g_{lm}}\left\{(\sum_x\sum_y C_{lm})(\sum_x\sum_y(-sS_{lm})) + (\sum_x\sum_y S_{lm})(\sum_x\sum_y(-sC_{lm}))\right\}\right] \quad (s\in\{x, y\})\,. \tag{A.9}$$

The modification to the bandwidth parameter can be derived in a similar way as,

$$\Delta\sigma_{ls} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial\sigma_{ls}(t)} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\frac{\partial E_m}{\partial g_{lm}}\frac{\partial g_{lm}}{\partial\sigma_{ls}} = -\frac{\eta_\omega}{M}\sum_{m=1}^{M}\left[(\sum_{n=1}^{N}\delta_{nm}w_{ml})\frac{\partial g_{lm}}{\partial\sigma_{ls}}\right]. \tag{A.10}$$

The partial derivative inside the summation of the right hand side of Eq. (A.10) can be written as

$$\frac{\partial g_{lm}}{\partial \sigma_{ls}} = \frac{\partial A_l}{\partial \sigma_{ls}}\sqrt{D_{lm}} + A_l \frac{\partial}{\partial \sigma_{ls}}(\sqrt{D_{lm}}) = \frac{-g_{lm}}{\sigma_{ls}} + \frac{A_l^2}{2g_{lm}} \frac{\partial D_{lm}}{\partial \sigma_{ls}}$$
$$= \frac{-g_{lm}}{\sigma_{ls}} + \frac{A_l^2}{g_{lm}} \left\{ (\sum_x \sum_y C_{lm})(\sum_x \sum_y (\frac{s^2}{\sigma_{ls}^3} C_{lm})) + (\sum_x \sum_y S_{lm})(\sum_x \sum_y (\frac{s^2}{\sigma_{ls}^3} S_{lm})) \right\}. \quad \text{(A.11)}$$

Thus, from Eqs. (A10) and (A11),

$$\Delta\sigma_{ls} = -\frac{\eta_\sigma}{M} \sum_{m=1}^{M} \left[(\sum_{n=1}^{N} \delta_{nm} w_{nl}) \left[\frac{-g_{lm}}{\sigma_{ls}} + \frac{A_l^2}{g_{lm}} \left\{ (\sum_{x,y} C_{lm})(\sum_{x,y} (\frac{s^2}{\sigma_{ls}^3} C_{lm})) + (\sum_{x,y} S_{lm})(\sum_{x,y} (\frac{s^2}{\sigma_{ls}^3} S_{lm})) \right\} \right] \right]$$
$$(s \in \{x, y\}). \quad \text{(A.12)}$$

Use of parameter modification measures calculated according to Eqs. (A.9) and (A.12) generally lead the training to the minima of the error surface. This was more successful in relatively wideband texture classes. For texture classes where the energy is concentrated to several narrow bands, the error surface tend to have a steep gradient, causing an unstable training process. This is caused by the absolute values of the modification measures growing very large according to the very steep descent in the error surface. On actual application of the training law, the kernel parameters were modified by

$$\Delta\omega'_{ls}(t) \equiv F[\Delta\omega_{ls}(t);\ \gamma_\omega] = \gamma_\omega \tanh(\frac{2\Delta\omega_{ls}(t)}{\gamma_\omega}) \quad \text{(A.13)}$$

and

$$\Delta\sigma'_{ls}(t) \equiv F[\Delta\sigma_{ls}(t);\ \gamma_\sigma] = \gamma_\sigma \tanh(\frac{2\Delta\sigma_{ls}(t)}{\gamma_\sigma}), \quad \text{(A.14)}$$

so that the change will be soft limited in order to stabilize the training process. Parameters γ_ω and γ_σ were empirically determined in a classification problem of pure-monotonic texture classes, to enable a smooth error convergence even for the severest cases having a very steep and and narrow error minima. The momentum term [5] was also added to the modification measures in Eqs. (5.5), (5.7), (5.9) and (5.10), in order to accelerate the training.

Appendix B
Modification of the Gabor kernel by the Bispectral Kernel Modifying Network's training.

Modification measures to the kernel parameters of the BKM Net (Eqs. (5.25) and (5.26)) will be derived in detail. However, the discussion will be of the one dimensional signal case, since expansion to two dimension is straightforward.

The output of the bispectral Gabor unit $M_3(x,\ \omega_1, \omega_2)$, defined in (5.24) will reappear here as,

$$M_3(x,\ \omega_1, \omega_2) = \left(i(x) \otimes k(x,\ \omega_1 + \omega_2)\right)^* \left(i(x) \otimes k(x,\ \omega_1)\right)\left(i(x) \otimes k(x,\ \omega_2)\right), \tag{5.24}$$

where i is the image and k is the filter kernel defined as

$$k(x,\ \omega) = w(x)\, e^{j\omega x}\ . \tag{5.14}$$

Since the image is a discrete signal, Eq. (5.24) can be written as,

$$M_3 = \left(\sum_u i(x-u)k(u,\ \omega_1 + \omega_2)\right)^* \left(\sum_u i(x-u)k(u,\ \omega_1)\right)\left(\sum_u i(x-u)k(u,\ \omega_2)\right), \tag{B.1}$$

where the summations are extended over the effective areas of the window w. If a moving coordinate system v whose origin is always at the center of the local subimage (and also the center of the window function w) is defined, Eq. (B.1) will be,

$$M_3 = \left(\sum_v i(v)k(-v,\ \omega_1 + \omega_2)\right)^* \left(\sum_v i(v)k(-v,\ \omega_1)\right)\left(\sum_v i(v)k(-v,\ \omega_2)\right). \tag{B.2}$$

Since the image i is real and the window w is a real even function,

$$M_3 = \left(\sum_v i(v)k(-v,\ \omega_1 + \omega_2)\right)\left(\sum_v i(v)k(v,\ \omega_1)^*\right)\left(\sum_v i(v)k(v,\ \omega_2)^*\right) \tag{B.3}$$

holds. By defining $i'(v) \equiv i(v)w(v)$, it can be further written as,

$$\begin{aligned} M_3 &= \left(\sum_v i'(v)e^{j(\omega_1+\omega_2)v}\right)\left(\sum_v i'(v)e^{-j\omega_1 v}\right)\left(\sum_v i'(v)e^{-j\omega_2 v}\right) \\ &= \left\{\sum_v i'(v)\cos[(\omega_1+\omega_2)v] + j\sum_v i'(v)\sin[(\omega_1+\omega_2)v]\right\} \\ &\quad \times \left\{\sum_v i'(v)\cos(\omega_1 v) - j\sum_v i'(v)\sin(\omega_1 v)\right\}\left\{\sum_v i'(v)\cos(\omega_2 v) - j\sum_v i'(v)\sin(\omega_2 v)\right\} \end{aligned} \tag{B.4}$$

$$\begin{aligned} &= \{C_0 + j\,S_0\}\{C_1 - j\,S_1\}\{C_2 - j\,S_2\} \\ &= (C_0C_1C_2 + S_0S_1C_2 + S_0C_1S_2 + C_0S_1S_2) + j\,(S_0C_1C_2 - C_0S_1C_2 - C_0C_1S_2 - S_0S_1S_2) \\ &= Re[M_3] + j\ Im[M_3]\ , \end{aligned} \tag{B.5}$$

where each summation is denoted by a single character such as, $C_0 \equiv \sum_v i'(v)\cos[(\omega_1+\omega_2)v]$ and $S_1 \equiv \sum_v i'(v)\sin(\omega_1 v)$, etc.

The factors in the modification measures of Eqs. (5.25) and (5.26) will be listed in the following using the notations of Eq. (B.5).

$$\frac{\partial E_m}{\partial(Re[M_{3l}])} = \sum_n \delta_{nm} w_{Rnl} \tag{B.6}$$

$$\frac{\partial E_m}{\partial(Im[M_{3l}])} = \sum_n \delta_{nm} w_{Inl} \tag{B.7}$$

Here, δ_{nm}, w_{Rnl} and w_{Inl} denote the *error*backpropagated from hidden layer unit n, real connection and the imaginary connection between the hidden and bispectral Gabor layers, respectively.

$$\begin{aligned} \frac{\partial(Re[M_{3l}])}{\partial\omega_{l1}} &= \{-(vS_o)C_1 + (vC_o)S_1 - C_o(vS_1) + S_0(vC_1)\}C_2 \\ &\quad + \{(vC_o)C_1 + (vS_o)S_1 - S_o(vS_1) - C_0(vC_1)\}S_2 \end{aligned} \tag{B.8}$$

$$\begin{aligned} \frac{\partial(Im[M_{3l}])}{\partial\omega_{l1}} &= \{(vC_o)C_1 + (vS_o)S_1 - S_o(vS_1) - C_0(vC_1)\}C_2 \\ &\quad + \{(vS_o)C_1 - (vC_o)S_1 + C_o(vS_1) - S_0(vC_1)\}S_2 \end{aligned} \tag{B.9}$$

$$\begin{aligned} \frac{\partial(Re[M_{3l}])}{\partial\omega_{l2}} &= \{-(vS_o)C_2 + (vC_o)S_2 - C_o(vS_2) + S_0(vC_2)\}C_1 \\ &\quad + \{(vC_o)C_2 + (vS_o)S_2 - S_o(vS_2) - C_0(vC_2)\}S_1 \end{aligned} \tag{B.10}$$

$$\frac{\partial(Im[M_{3l}])}{\partial \omega_{l2}} = \{(vC_o)C_2 + (vS_o)S_2 - S_o(vS_2) - C_0(vC_2)\}C_1 + \{(vS_o)C_2 - (vC_o)S_2 + C_o(vS_2) - S_0(vC_2)\}S_1 \quad \text{(B.11)}$$

In Eqs. (B.8) ~ (B.11), notations of $(vC_0) \equiv \sum_v v\, i'(v)\cos[(\omega_1 + \omega_2)v]$ and $(vS_1) \equiv \sum_v v\, i'(v)\sin(\omega_1 v)$ etc. are used.

The framework of the BKM Net can accomodate any type of window functions as long as

$$\int |w(x)|^2\, dx = 1 \quad \text{(B.12)}$$

holds for energy preservation. As the Gabor kernels defined in Eq. (5.1) was used as well, the factors in the modification of the bandwidth parameter (again, in the one dimensional case) were as follows.

$$\frac{\partial(Re[M_{3l}])}{\partial \sigma_l} = \frac{-3}{\sigma_l} Re[M_{3l}] + \left[(\frac{v^2}{\sigma_l^3}C_o)(C_1C_2 - S_1S_2) + C_o\{(\frac{v^2}{\sigma_l^3}C_1)C_2 + C_1(\frac{v^2}{\sigma_l^3}C_2) - (\frac{v^2}{\sigma_l^3}S_1)S_2 - S_1(\frac{v^2}{\sigma_l^3}S_2)\} + (\frac{v^2}{\sigma_l^3}S_o)(S_1C_2 + C_1S_2) + C_o\{(\frac{v^2}{\sigma_l^3}S_1)C_2 + S_1(\frac{v^2}{\sigma_l^3}C_2) + (\frac{v^2}{\sigma_l^3}C_1)S_2 + C_1(\frac{v^2}{\sigma_l^3}S_2)\} \right] \quad \text{(B.13)}$$

$$\frac{\partial(Im[M_{3l}])}{\partial \sigma_l} = \frac{-3}{\sigma_l} Im[M_{3l}] + \left[(\frac{v^2}{\sigma_l^3}C_o)(S_1C_2 + C_1S_2) - C_o\{(\frac{v^2}{\sigma_l^3}S_1)C_2 + S_1(\frac{v^2}{\sigma_l^3}C_2) + (\frac{v^2}{\sigma_l^3}C_1)S_2 + C_1(\frac{v^2}{\sigma_l^3}S_2)\} + (\frac{v^2}{\sigma_l^3}S_o)(C_1C_2 - S_1S_2) + S_o\{(\frac{v^2}{\sigma_l^3}C_1)C_2 + C_1(\frac{v^2}{\sigma_l^3}C_2) - (\frac{v^2}{\sigma_l^3}S_1)S_2 - S_1(\frac{v^2}{\sigma_l^3}S_2)\} \right] \quad \text{(B.14)}$$

Here, notations of $(\frac{v^2}{\sigma_l^3}C_0) \equiv \sum_v \frac{v^2}{\sigma_l^3} i'(v)\cos[(\omega_1 + \omega_2)v]$ and $(\frac{v^2}{\sigma_l^3}S_1) \equiv \sum_v \frac{v^2}{\sigma_l^3} i'(v)\sin[\omega_1 v]$ etc. are used.

References

Neural networks and pattern recognition

[1] E. Parzen, “On estimation of a probability density function and mode,” Annals of Mathematical Statistics, vol. 33, pp. 1065-1076, 1962.

[2] M. L. Minsky and S. A. Papert, Perceptrons, Expanded edition, MIT Press 1988.

[3] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, 1972.

[4] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, 1973.

[5] D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Parallel distributed processing, vol. 1, MIT Press 1986.

[6] R. Courant and D. Hilbert, Methods of Mathematical Physics, Springer, 1937.

[7] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of Control, Signals and Systems, vol. 2, pp. 303-314, 1989.

[8] K. Funahashi, “On the approximate realization of continuous mappings by neural networks,” Neural Networks, vol. 2, pp. 183-192, 1989.

[9] S. Huang and Y. Huang, “Bounds on the number of hidden neurons in multilayer perceptrons,” IEEE Trans. on Neural Networks, vol. 1, no. 1, pp. 47-55, 1991.

[10] P. Baldi and K. Hornik, “Neural networks and pricipal component analysis: learning from examples without local minima,” Neural Networks, vol. 2, pp. 53-58, 1989.

[11] M. Gori and A. Tesi, “On the problem of local minima in backpropagation,” IEEE Trans. PAMI, vol. 14, no. 1, pp. 76-86, 1992.

[12] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” AIChE-Journal, vol. 37, no. 2, pp. 233-243, 1991.

[13] H. Leung and S. Haykin, “The complex backpropagation algorithm,” IEEE Trans. Sig. Proc., vol. 39, no. 9, 1991.

[14] E. Oja, Subspace Methods of Pattern Recognition, Wiley, 1983

[15] P. Heckbert, “Color image quantization for frame buffer display,” Computer Graphics, vol. 16, no. 3, pp. 297-307, 1982.

[16] C. L. Giles and T. Maxwell, “Learning, invariance and generalization in high-order neural networks,” Applied Optics, vol. 26, no. 23, pp. 4972-4978, 1987.

[17] K. Tanaka, M. Yamamura and S. Kobayashi, “On the capability of the higher-order back-propagation Network,” Trans. SICE, vol. 28, no. 1, pp. 125-134, 1992 (In Japanese).

[18] M. Ishikawa, “Structural learning,” Neural-Networks,vol. 9 no. 3 pp. 509-521, 1996.

[19] J. E. Moody and C. J. Darken, “Fast learning in networks of locally-tuned processing units,” Neural Computation, vol. 1, pp. 281-294, 1989.

[20] T. Poggio and F. Girosi, “Networks for approximation and learning,” Proceedings of the IEEE, vol. 78, pp. 1481-1497, 1990.

[21] N. B. Karayiannis and G. W. Mi, “Growing radial basis neural networks: merging supervised and unsupervised learning with network growing techniques,” IEEE Trans. Neural Networks, vol. 8, no. 6, pp. 1492-1506, 1997.

[22] D. F. Specht, “Probabilistic neural networks,” Neural Networks, vol. 3, pp. 109-118, 1990.

[23] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley 1990.

[24] S. Haykin, Neural Networks - A Comprehensive Foundation, Macmillan, 1994.

[25] K. Ishida, K. Kameyama and Y. Kosugi, “Adaptive constitution method of hidden layer in neural network with RBFs,” Technical Report of IEICE, NC95-112, pp. 83-90, 1996 (In Japanese).

[26] M. Hirahara and N. Oka, “A hybrid model composed of a multilayer perceptron and a radial basis function network,” Proc. IJCNN, vol. 2, pp. 1353-1356, (1993).

[27] S. Geva and J. Sitte, “A constructive method for multivariate function approximation by multilayer perceptrons,” IEEE Trans. Neural Networks, vol. 3, no. 4, pp. 621-624, 1992.

[28] D. Cohen and J. Shawe-Taylor, “Feedforward networks - a tutorial”, in J. G. Taylor and C. L. T. Mannion (Ed.), New developments in neural computing, pp. 1-13, Adam Hilger (1989).

[29] T. Kohonen, Self-organization and associative memory, Springer (1988).

[30] M. C. Mozer and P. Smolensky, “Skeletonization : a technique for trimming the fat from a network via relevance assessment”, Advances in Neural Information Processing 1, D. S. Touretzky, Ed. Morgan Kaufmann, pp. 177-185, (1989).

[31] E. D. Karnin, “A simple procedure for pruning back-propagation trained neural networks”, IEEE Trans. Neural Networks, vol. 1, pp. 239-242, (1990).

[32] J. Sietsma and R. J. F. Dow, “Neural net pruning - why and how?”, Proc. IEEE Int. Conf. Neural Networks, vol. 1 (San Diego, Calif.), pp. 177-185, (1988).

[33] J. Sietsma and R. J. F. Dow : “Creating artificial neural networks that generalize”, Neural Networks, vol. 4, no. 1, pp. 67-79 (1991).

[34] K. Kameyama and Y. Kosugi, “Neural network pruning by fusing hidden layer units,” Transactions of IEICE, vol. E74, no. 12, pp. 4198 - 4204, 1991.

[35] Y. LeCun, J. S. Denker and S. A. Solla, “Optimal brain damage,” Advances in Neural Information Processing Systems, vol. 2, pp. 598-605, Morgan Kaufmann, 1990.

[36] B. Hassibi, D. G. Stork and G. J. Wolff, “Optimal brain surgeon and general network pruning,” Proc. International Conf. on Neural Networks, vol. 1, pp. 293-299, 1993.

[37] G. Castellano, A. M. Fanelli and M. Pelillo, “An iterative pruning algorithm for feedforward neural networks,” IEEE Trans. Neural Networks, vol. 8, no. 3, pp. 519-531, 1997.

[38] R. Reed, “Pruning algorithms – a survey,” IEEE Trans. Neural Networks, vol. 4, no. 5, pp. 740-747, 1993.

[39] Y. Hirose, K. Yamashita and S. Hijiya, “Back-propagation algorithm which varies the number of hidden units,” Neural Networks, vol. 4, pp. 61-66, 1991.

[40] T. Oshino, J. Ojima and S. Yamamoto, “Method for gradually reducing a number of hidden units on back propagation learning algorithm,” Trans. IEICE, vol. J76-D-II, no. 7, pp. 1414-1424, 1993 (In Japanese).

[41] S. Yamamoto et al. “Gradual reduction of hidden units in the back propagation algorithm, and its application to blood cell classification,” Proc. IJCNN, vol. 3, pp. 2085 - 2088, 1993.

[42] K. Kameyama and Y. Kosugi, “Automatic fusion and splitting of artificial neural elements in optimizing the network Size,” Proc. International Conf. on Systems, Man and Cybernetics, vol. 3, pp.1633 - 1638, 1991.

[43] S. E. Fahlman and C. Lebiere, “The cascade-correlation learning architecture,” Advances in Neural Information Processing Systems, vol. 2, pp. 524-532, Morgan Kaufmann, 1990.
[44] S. E. Fahlman and C. Lebiere, “The cascade-correlation learning architecture,” CMU Technical Report CMU-CS-90-100, 1991.
[45] R. Kothari and K. Agyepong, “On lateral connections in feed-forward neural networks,” Proc. ICNN, vol. 1, pp. 13-18, 1996.
[46] I. Valova and Y. Kosugi, “MR brain image classification by multimodal perceptron tree neural network,” Neural Networks for Signal Processing, vol. 7, pp. 189-198, 1997.
[47] S. A. Harp and T. Samad, “Genetic synthesis of neural network architecture,” in L. Davis Ed. Handbook of Genetic Algorithms, Chap. 15, Van Nostrand Reinhold, 1991.
[48] H. Kitano, “Designing neural networks using genetic algorithms with graph generation system,” Complex System, vol. 4, no. 4, 1990.
[49] H. Kitano, “Neurogenetic learning : an integrated method of designing and training neural networks using genetic algorithms,” Tech. Rep. Carnegie Mellon Univ., 1992.
[50] H. Akaike, “A new look at the statistical model identification,” IEEE Trans. Automatic Control, vol. AC-19, no. 6, pp. 716-723, 1974.
[51] T. Kurita, “A method to determine the number of hidden units of three layered neural networks by information criteria,” Trans. IEICE, vol. J73-D-II, no. 11, pp. 1872-1878, 1990.
[52] D. B. Fogel, “An information criterion for optimal neural network selection,” IEEE Trans. Neural Networks, vol. 2, no. 5, pp. 490-497, 1991.
[53] T. Onoda, “Experimental analysis of generalization capability based on information criteria,” Proc. IJCNN, vol. 1, pp. 114-119, 1996.
[54] H. Ogawa, “Neural network theory as an inverse problem,” Journal of IEICE, vol. 73, no. 7, pp. 690-695, 1990.
[55] H. Ogawa and K. Yamasaki, “A theory of over-learning,” Trans. IEICE, vol. J76-D-II, no. 6, pp. 1280-1288, 1993.
[56] H. Ogawa and J. Funada, “A realization method of optimally generalizing neural network based on error minimization,” Technical Report of IEICE, NC93-127, pp. 115-122, March, 1994.
[57] H. Ogawa, “Neural networks and generalization ability,” Techinical Report of IEICE, NC95-8, pp. 57-64, May, 1995.
[58] A. Hirabayashi and H. Ogawa, “Admissibility of memorization learning with respect to projection learning in the presence of noise,” Proc. ICNN’96, pp. 335-340, 1996.

Signal and image processing

[59] L. Cohen, “Time-frequnecy distributions – a review,” Proc. IEEE, vol. 77, no. 7, pp. 941 - 981, 1989.
[60] C. L. Nikias and A. P. Petropulu, Higher-Order Spectra Analysis – A Nonlinear Signal Processing Framework –, Prentice Hall 1993.
[61] B. Widrow and S. D. Stearns, Adaptive Signal Processing, Prentice Hall 1985.
[62] N. L. Gerr, “Introducing a third-order wigner distribution,” Proc. IEEE, vol. 76, no. 3, pp. 290 - 292, 1988.

[63] J. S. Bendat and A. G. Piersol, Random data : Analysis and measurement procedures, Wiley-Interscience, (1971).

[64] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," J. Opt. Soc. Am. A, vol. 2, no. 7, pp. 1160 - 1169, July 1985.

[65] A. C. Bovik, M. Clark and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," IEEE Trans. PAMI, vol. 12, no. 1, pp. 55 - 73, Jan. 1990.

[66] D. Dunn, W. E. Higgins and J. Wakeley, "Texture segmentation using 2-D Gabor elementary functions," IEEE Trans. PAMI, vol. 16, no. 2, pp. 130 - 149, Feb. 1994.

[67] J. G. Daugman, "High confidence visual recognition of persons by a test of statistical independence," IEEE Trans. PAMI, vol. 15, no. 11, pp. 1148 - 1161, Nov. 1993.

[68] P. P. Raghu, R. Poongodi and B. Yegnanarayana, "A combined neural network approach for texture classification," Neural Networks, vol. 8, no. 6, pp. 975 - 987, 1995.

[69] A. K. Jain and K. Karu, "Learning texture discrimination masks," IEEE Trans. PAMI, vol. 18, no. 2, pp. 195 - 205, Feb. 1996.

[70] A. Teuner, O. Pichler and B. J. Hosticka, "Unsupervised texture segmentation of images using tuned matched Gabor filters," IEEE Trans. Image Processing, vol 4, no. 6, pp. 863 - 870, June 1995.

[71] A. Biem, E. McDermott and S. Katagiri, "Discriminative feature extraction application to filter bank design," Proc. IEEE Workshop of Neural Net. for Sig. Pro., pp. 273-282, 1996.

[72] B-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," IEEE Trans. Signal Processing, vol. 40, no. 12, 1992.

[73] R. Picard, C. Graczyk, S. Mann, J. Wachman, L. Picard, and L. Campbell, Vision Texture 1.0, 1995.

Biological tissue characterization using ultrasound

[74] S. Hasegawa, K. Hayashi, T. Sato and K. Kameyama, "Simultaneous imaging system of three kinds of parameters of nonlinearity for medical diagnosis," Acoustical Imaging, vol. 21, pp. 391 - 397, 1994.

[75] K. Kameyama, T. Sato, K. Hayashi, S. Hasegawa and I. Kodaira, "Information fusion of images of linear and nonlinear parameters for efficient medical diagnosis," Ultrasound in Medicine and Biology, vol. 20, supplement 1, p. 74, 1994.

[76] T. Sato, Y. Mochida, K. Fujii, I. Yu. Demin, K. Y. Jhang, K. Kobayashi, M. Kato and K. Kameyama, "Bispectral analysis applied for measurement of nonlinear characteristics of vibration propagation in soft tissues," Proceedings of IEEE Signal Processing Workshop on Higher Order Statistics, pp. 369 - 373, 1993.

[77] K. Kameyama, T. Inoue, I. Yu Demin, K. Kobayashi and T. Sato, "Acoustical tissue nonlinearity characterization using bispectral analysis," Signal Processing (Elsevier), vol. 53, issue 2-3, pp. 117-131, 1996.

Semiconductor manufacturing and inspection

[78] M. H. Bennett, “Automatic defect classification: status and industry trends,” Proc. SPIE, vol. 2439, pp. 210-220, 1995.

[79] L. Breaux and B. Singh, “Automatic defect classification system for patterned semiconductor wafers,” Proc. Int’l Symposium on Semiconductor Manufacturing, pp. 68-73, 1995.

[80] P. B. Chou, A. R. Rao, M. C. Struzenbecker, F. Y. Wu and V. H. Brecher, “Automatic defect classification for semiconductor manufacturing,” Machine Vision and Applications, vol. 9, no. 4, pp. 201-214, 1997.

Publications

A. Journal papers

1. Keisuke Kameyama and Yukio Kosugi (Ch. 3)
   Neural Network Pruning by Fusing Hidden Layer Units
   Transactions of IEICE, vol. E74, no. 12, pp. 4198 - 4204, 1991

2. Keisuke Kameyama, Kenzo Mori and Yukio Kosugi (Ch. 5)
   Texture Segmentation Using A Kernel Modifying Neural Network
   IEICE Transactions on Information and Systems, vol. E80-D, no. 11, pp. 1092-1101, 1997

3. Keisuke Kameyama, Yukio Kosugi, Tatsuo Okahashi and Morishi Izumita (Ch. 6)
   Automatic Defect Classification in Visual Inspection of Semiconductors Using Neural Networks
   IEICE Transactions on Information and Systems, vol. E81-D, no. 11, pp. 1261-1271, 1998

4. Keisuke Kameyama and Yukio Kosugi (Ch. 2, 3, 5)
   Neural Network Model Switching for Efficient Feature Extraction
   (IEICE Transactions on Information and Systems採録決定　印刷中)

5. Ivan Fanany, Hideki Mawatari, Keisuke Kameyama, Benjamin Kusumoputro and Yukio Kosugi
   Bispectrum Nonlinear Vector Quantization to Speaker Verification
   (IEICE Transactions on Information and Systems投稿中) (Ch. 5)

B. Conference proceedings

1. Keisuke Kameyama and Yukio Kosugi (Ch. 3)
   Automatic Fusion and Splitting of Artificial Neural Elements in Optimizing the Network Size
   Proceedings of the International Conference on Systems, Man and Cybernetics 1991, vol. 3, pp.1633 - 1638, 1991

2. K. Kameyama, T. Sato, S. Hasegawa, K. Hayashi, I. Kodaira (Ch. 5)
   Information Fusion of Images of Linear and Nonlinear Parameters for Efficient Medical Diagnosis
   Ultrasonic Imaging, vol. 16, no. 1, p. 40, 1994

3. Seiya Hasegawa, Katsuhiko Hayashi, Takuso Sato and Keisuke Kameyama (Ch. 5)
   Simultaneous Imaging System of Three Kinds of Parameters of Nonlinearity for Medical Diagnosis
   Acoustical Imaging (Plenum), vol. 21, pp. 391-397, 1994

4. K. Kameyama, T. Sato, K. Hayashi, S. Hasegawa and I. Kodaira (Ch. 5)
   Information Fusion of Images of Linear and Nonlinear Parameters for Efficient Medical Diagnosis
   Ultrasound in Medicine and Biology, vol. 20 supplement 1, p.74, 1994

5. Keisuke Kameyama, Kenzo Mori and Yukio Kosugi (Ch. 5)
A Neural Network Incorporating Adaptive Gabor Filters for Image Texture Segmentation
Proceedings of 1997 International Conference on Neural Networks (Houston), vol. 3, pp. 1523-1528, 1997

6. Keisuke Kameyama and Yukio Kosugi (Ch. 5)
Spectral and Bispectral Feature Extraction Neural Network for Texture Classification
Proc. SPIE, Statistical and Stochastic Methods in Image Processing II, vol. 3167, pp. 93 - 103, 1997

7. Keisuke Kameyama and Yukio Kosugi (Ch. 2, 3, 5)
Model Switching by Channel Fusion for Network Pruning and Efficient Feature Extraction
Proceedings of International Joint Conference on Neural Networks 1998, pp. 1861-1866, 1998

8. Keisuke Kameyama and Yukio Kosugi (Ch. 6)
Semiconductor defect classification using Hyperellipsoid Clustering Neural Networks and model switching
Proceedings of International Joint Conference on Neural Networks 1999, In press, 1999

C. Domestic conferences and meets

1. Keisuke Kameyama and Yukio Kosugi (Ch. 3)
Neural Network Learning Using Multi Channel Neurons
信学技報, PRU90, pp. 7 - 14, 1990

2. 石田和子, 亀山啓輔, 小杉幸夫 (Ch. 4)
RBFを含むニューラルネット中間層の適応的構成法
信学技報, NC95-112(1996-02), pp. 83-90, 1996

3. Keisuke Kameyama, Kenzo Mori , Tomohisa Suzuki and Yukio Kosugi (Ch. 5)
Texture Classification by Adaptive Extraction of Second and Third-Order Statistical Features,
信学技報, NC96-197 (1997-03), pp. 323-330, 1997

4. 橋本直久, 亀山啓輔, 小杉幸夫, 岡橋卓夫, 泉田守司 (Ch. 6)
２次制限型高次ニューラルネットワークによるLSI欠陥像認識機構の構築,
信学技報, NC96-188 (1997-03), pp. 251-258, 1997

5. Keisuke Kameyama and Yukio Kosugi (Ch. 4)
An algorithm for model determination in a layered network with non-uniform hidden layer unit set
信学技報, NC98-96(1999-02), pp. 141-148, 1999

D. Other publications

Journal papers

1. 亀山啓輔, 小杉幸夫
インピーダンス分布再構成におけるニューラルネットワークの適用
電子情報通信学会論文誌, vol. J75-D-II, no. 2, pp. 435-439, 1992

2. 小杉幸夫，亀山啓輔
層間荷重に解を得る誤差逆伝搬法の逆問題への適用
電子情報通信学会論文誌, vol. J77-D-II, no. 3, pp. 600 - 605, 1994

3. Iren Valova, Keisuke Kameyama and Yukio Kosugi
Image Decomposition by Answer-in-Weights Neural Network
IEICE Transactions on Information and Systems, vol. E78-D, no. 9, pp. 1221-1224, 1995

4. M. Kato, T. Sato, K. Kameyama and H. Ninoyu
Esitimation of the Stress Distribution in Metal Using Nonlinear Acoustoelasticity
J. Acoust. Soc. Am., vol. 98, no. 3, pp. 1496-1504, 1995

5. Takehiko Ogawa, Keisuke Kameyama, Roman Kuc and Yukio Kosugi
Source Localization with Network Inversion Using an Answer-in-Weights Scheme
IEICE Transactions on Information and Systems, vol. E79-D, no. 6, pp. 608-619, 1996

6. Keisuke Kameyama, Toshikazu Inoue, Igor Yu Demin, Koichi Kobayashi and Takuso Sato
Acoustical Tissue Nonlinearity Characterization Using Bispectral Analysis
Signal Processing (Elsevier), vol. 53, issue 2-3, pp. 117-131, 1996

7. 加藤勝，佐藤拓宋，亀山啓輔，小杉幸夫
変形の確率モデルによる金属中の応力状態の評価
非破壊検査，vol. 46，no. 4，pp. 293 - 298，1997

8. Masaru Kato, Takuso Sato, Keisuke Kameyama and Yukio Kosugi
Spatial Localization of Strress-perturbing Wave Generated by an Electromagntic Acoustic Transducer
IEEE Trans. on Ultrasonics, Ferroelectrics and Frequency Control, vol. 44, no. 5, pp. 1132-1139, 1997

Conference proceedings

1. Yukio Kosugi, Keisuke Kameyama, Joichi Bita, Nobuyuki Shitara and Kintomo Takakura
Neural Network Approaches to the Flowcytometric Data Analysis
Flow cytometry and image analysis for clinical applications, Elsevier (Proc. Int'l. Symp. on Flow Cytometry and Image Analysis for Clinical Applications 1990), pp. 181 - 186, 1991

2. Keisuke Kameyama, Masahiko Iwata, Takuji Sakai, Kyung-Young Jhang and Takuso Sato
Acousto-Optical Third Order Correlator
Proceedings of IEEE Signal Processing Workshop on Higher Order Statistics, pp. 81 - 85, 1993

3. Keisuke Kameyama, Masashi Sakamoto, Hirofumi Akagi, Kyung-Young Jhang and Takuso Sato
Robot Vision System Using High Order Correlation Analysis
Proceedings of IEEE Signal Processing Workshop on Higher Order Statistics, pp. 86 - 90, 1993

4. T. Sato, Y. Mochida, K. Fujii, I. Yu. Demin, K. Y. Jhang, K. Kobayashi, M. Kato and K. Kameyama
Bispectral Analysis Applied for Measurement of Nonlinear Characteristics of Vibration

Propagation in Soft Tissues
Proceedings of IEEE Signal Processing Workshop on Higher Order Statistics,
pp. 369 - 373, 1993

5. K. Fujii, T. Sato, K. Kameyama, K. Kobayashi
Theoretical treatment of propagation characteristics of vibration in soft tissues and a method to obtain precise hardness images in vivo
Ultrasonic Imaging, vol. 15, no. 2, pp. 165 - 166, 1993

6. K. Hayashi, T. Sato, S. Hasegawa, K. Kameyama
Ultrasonic imaging system combined with perturbing waves for observation of dynamic and fine structure characteristics of soft tissues
Ultrasonic Imaging, vol. 15, no. 2, p. 166, 1993

7. Masaru Kato, Takuso Sato, Keisuke Kameyama and Hideyuki Ninoyu
Nondestructive Imaging of Stress Distribution in Metal Using Nonlinear Elasto-Acoustics
Acoustical Imaging (Plenum), vol. 21, pp. 621-626, 1994

8. Katsunori Fujii, Takuso Sato, Keisuke Kameyama, Toshikazu Inoue, Katsunori Yokoyama and Koichi Kobayashi
Imaging System of Precise Hardness Distribution in Soft Tissue in vivo Using Forced Vibration and Ultrasonic Detection
Acoustical Imaging (Plenum), vol. 21, pp. 253-258, 1994

9. M. Sakamoto, K. Kameyama, K. Kuwano and T. Sato
Movement Tracer System Using Non-parallel Multiple Line Detectors and High Order Correlation Analysis
Proc. IAPR Workshop on Machine Vision Applications, pp. 178 - 181, 1994

10. I. Yu. Demin, T. Sato, Y. Mochida, K. Fujii, K. Y. Jhang, K. Kobayashi, M. Kato and K.Kameyama
Nonlinear characteristics of propagation of low frequency vibration in soft tissues and its measurements using bispectral analysis
Ultrasonic Imaging, vol. 16, no. 1, p. 39 1994

11. I. Yu. Demin, K. I. Dzhang, K. Kobayashi, K. Kameyama, M. Kato, K. Fudzhi and T. Sato
Using low-frequency acoustic waves for diagnosis of mild biological tissues
Akusticheskii Zurnal, vol. 41, no. 3, p 508, 1995.

12. Yukio Kosugi, Yusuke Suganami, Naoko Uemoto, Keisuke Kameyama, Mikiya Sase, Toshimitsu Momose and Junichi Nishikawa
CCE-Based Index Selection for Neuro Assisted MR-Image Segmentation
Proc. IEEE 1996 International Conf. on Image Processing (Lausanne), pp. 249-252, 1996

Domestic conferences and meets

1. 小杉幸夫，亀山啓輔
逆問題解法のためのAnswer-inWeights Scheme
信学技報，NC91，pp. 223 - 228，1992

2. 藤井勝紀，佐藤拓宋，亀山啓輔，小林浩一
生体軟組織中での低周波振動の伝播特性の定式化とその医用診断への応用
第４０回応用物理学関係連合講演会予稿集，vol. 1，p. 42，1993

3. 林勝彦，佐藤拓宋，亀山啓輔
摂動波を併用した超音波映像系による生体組織のダイナミック特性の計測
第４０回応用物理学関係連合講演会予稿集，vol. 1，p. 42，1993

4. 加藤一憲，亀山啓輔，佐藤拓宋
金属材料の非線形音弾性特性と塑性変形のモデル化
第４０回応用物理学関係連合講演会予稿集，vol. 1，p. 43，1993

5. 岩山政彦，亀山啓輔，佐藤拓宋
超音波光変調器を用いた３次相関関数解析器の高Ｓ／Ｎ化と広帯域化
第４０回応用物理学関係連合講演会予稿集，vol. 1，p. 42，1993

6. 坂本真志，亀山啓輔，佐藤拓宋
高次相関解析を用いたロボットビジョンの高精度化
第４０回応用物理学関係連合講演会予稿集，vol. 1，p. 42，1993

7. 亀山啓輔，藤井勝紀，小林浩一，佐藤拓宋
生体軟組織の硬度マップの高精度形成法
Jpn. J. Med. Ultrasonics, vol. 21 [Supplement 1], p. 281, 1994

8. 井上俊一，亀山啓輔，小林浩一，佐藤拓宋
生体軟組織中での低周波振動の非線形伝搬特性の計測と映像形成
Jpn. J. Med. Ultrasonics, vol. 21 [Supplement 1], p. 277, 1994

9. 小平格，亀山啓輔，小林浩一，佐藤拓宋
線形および非線形性の映像の結合による合成像の形成
Jpn. J. Med. Ultrasonics, vol. 21 [Supplement 1], p. 282, 1994

10. 長屋俊輔，亀山啓輔，小杉幸夫
強化学習モデルを用いた移動ロボットの適応的進路制御
信学技報，NC95-69(1995-11)，pp. 29-36，1995

11. 菅波雄介，亀山啓輔，小杉幸夫，佐瀬幹哉，百瀬敏光，西川潤一
ＭＲ援用ＰＥＴ画像処理における組織分類ニューラルネット
信学技報，MBE95-95(1995-10)，pp. 45-52，1995

12. 加藤勝，安藤一隆，佐藤拓宋，亀山啓輔，二之湯秀幸
非線形音弾性を用いた金属中の応力分布の推定
超音波による非破壊評価シンポジウム講演論文集，pp. 13-20, 1995

13. 加藤勝，佐藤拓宋，安藤一隆，亀山啓輔，小杉幸夫
変形の確率的モデルによる金属中の応力状態の評価
日本非破壊検査協会平成７年度秋季大会講演論文集，pp. 413-420, 1995

14. 鈴木啓之，亀山啓輔，小杉幸夫
低周波振動と超音波検出を用いた生体軟組織異方性の映像形成
信学技報，US95-87(1996-01)，pp. 85-92，1996

15. 加藤勝，佐藤拓宋，亀山啓輔，小杉幸夫
電磁誘導型トランスデューサによる応力摂動波の空間的局在化
第３回超音波による非破壊評価シンポジウム講演論文集，pp. 47-54, 1996

16. 加藤勝，佐藤拓宋，亀山啓輔，小杉幸夫
電磁誘導型トランスデューサによる応力摂動波の空間的局在化
第６回音弾性材料評価研究会抄録，pp. 54-61, 1996

17. 馬渡秀樹，亀山啓輔，小杉幸夫
バイスペクトルを用いたニューラルネットワーク援用話者認識システムに関する研究，
信学技報，NC96-146 (1997-03)，pp. 255-262，1997

18. 森健造，亀山啓輔，小杉幸夫
作業仮説にもとづいた問題解決手法の適応的探索
信学技報，NC97-152(1998-03)，pp. 103-110，1998

19. 久保博紀，亀山啓輔，小杉幸夫
画像からの曲面推定問題におけるネットワークインバージョンの適用
信学技報，NC97-169(1998-03)，pp. 233-240，1998

20. 石田隆志，亀山啓輔，小杉幸夫
操作領域限定型Feature Mapによる画像処理
信学技報，NC97-170(1998-03)，pp. 241-248，1998

21. 高石秀樹，森谷直哉，亀山啓輔，小杉幸夫
幾何学的特徴量に基づいた地理画像の非線形写像とそのネットワーク表現
信学技報，NC98-94(1999-02)，pp. 125-132，1999

Review papers

1. 佐藤拓宋，佐々木修己，鈴木孝昌，亀山啓輔
位相共役効果を用いた光計測・センシング（解説論文）
計測と制御，vol. 32，no. 11，pp. 895 - 901, 1993