# GENERALIZATION CAPABILITY OF NEURAL NETWORKS

Thesis by Chuanyi Ji

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1992

(Defended October 25, 1991)

*This thesis is dedicated to the memory of my father,*

*and to my mother and daughter.*

## Acknowledgement

I would like to express my gratitude to my thesis advisor, Professor Demetri Psaltis, for all his advice and encouragement in the course of this work. I appreciate, especially, his way of transmitting enthusiasm, creativity and high standards in doing research to his students.

I thank Professor Edward Posner for kindly reading my preprints and his insights; and Professor Yaser Abu-Mostafa for the helpful discussions as well as his concise way of presenting ideas. I would also like to show my appreciation to Professor David Rutledge for all his help.

I am grateful to Mrs. Su Mckinly and Mrs. Hellen Carrier for their kindly administrative helps.

My special thanks also go to Professor Saleem Kassam at University of Pennsylvania for all his help and encouragement.

Many thanks go to my friends and colleagues at Caltech. Among them, I enjoyed collaborations with Alan Yamamura and Dr. Robert Snapp, which resulted in parts of the work in Chapter 5; and many helpful discussions with Dr. Amir Atiya, Dr. Mark Neifield, Charlie Stirk, Dr. Timothy Brown, John Miller, Subrata Rakshit, Dr. Scott Hudson, Dr. Santosh Venkatesh, Yu-fai Wong, Dr. Cheolhoon Park and Robert Denkewalter. I appreciate friendships from Dr. Claire Gu, Dr. Zhaoping Li, Dr. Ken Hsu, Hsin-Yu Li, Dr. Steven Lin, Yong Qiao, Dr. Nabel Retz, Dr. David Brady, Dr. Mok Fai, Annette Grot, Kevin Kurtis, David Marks and Ruth Erlanson.

I can never emphasis enough how much appreciation I have for my family, for my parents Xiaogung Ji and Hongju He, for their unwavering support and encouragement through the years; and for my brother Chuanshu Ji for his significant influence and encouragement. To my daughter Sophia J. Li, her love, understanding and cooperation contribute to my motivations and the completeness of the thesis. To my husband Mo Li, I deeply appreciate his constant encouragement and support in every way, and all his understanding and sharing.

## Abstract

The generalization capability of feedforward multilayer neural networks is investigated from two aspects: the theoretical aspect and the algorithmic aspect.

In the theoretical part, a general relation is derived between the so-called VC-dimension and the statistical lower epsilon-capacity, and then applied to two cases. First, as a general constructive approach, it is used to evaluate a lower bound of the VC-dimension of two layer networks with binary weights and integer thresholds. Second, how the sample complexity may vary with respect to distributions is investigated through analyzing a particular network which separates two binary clusters. Bounds for the capacity of two layer networks with binary weights and integer thresholds are also obtained.

In the algorithmic part, a network reduction algorithm is developed to study generalization in learning analog mappings. It is applied to control a two-link manipulator to draw characters. The network addition-deletion algorithm is described to find an appropriate network structure during learning. It is used to study the effect of sizes of networks on generalization, and applied to various classification problems including hand written digits recognition.

# Contents

# Chapter 1

# Introduction

Artificial neural networks consist of nonlinear threshold units connected with one another through interconnections called weights. A special type of neural network, feedforward multilayer neural networks, are of particular interest in this thesis, since it has been shown that with enough resources (units) they are capable of approximating any nonlinear mapping [8] [2].

Neural network models of computation are suitable for those problems which can not be solved by a well-defined formula or algorithm, due to the learning capability of neural networks. A commonly used learning model is learning from examples, or specifically, supervised learning. In a setting of supervised learning, what is usually available is a training set, which consists of a finite number of desired input-outout pairs. A neural network model can be developed by modifying its parameters so that it can absorb some useful information about an unknown environment. However, a network thus obtained is not guaranteed to be able to approximate the true underlying mapping, since it is possible for a resulting network to simply behave like a look-up table, which can remember training samples perfectly, but fail to respond well to new samples belonging to the same problem but which it has never seen before. Therefore, to make neural networks useful models of information processing, it is crucial to investigate the problem of generalization.

Loosely speaking, generalization can be identified as the ability of networks to have persistent performance on unseen inputs. Rigorously, it can be defined in a probabilistic setting using the so-called PAC learning model which was introduced into the machine learning community by Valliant [4], and into the neural network community by Baum and Hassler [1]. This theory answers two questions: a) whether generalization is possible, b) when it can happen. According to this theory, a single quantity called the VC-dimension [5], which is a distribution-free and network parameter-independent, can be used to characterize the generalizing capability of a class of networks with the same structure. There are, however, two of the questions given below which the existing theory has not been able to answer yet, and which are of interest for this thesis.

1) How does generalization relate to memorization, where the memorizing capability of a network is characterized by its information capacity [10] [8]?

3) How can one find networks that generalize well?

To contribute answers to these questions, the work presented in this thesis investigates generalization from two aspects: the theoretical aspect in the first three chapters, and the computational aspect in the Chapters 5 and 6.

The theoretical results focus on the VC-dimension, the statistical information capacity and their relations, to investigate how degrees of freedom of networks relate to generalization. Specifically, in Chapter 2 a general relationship is developed for feed-forward multilayer networks in general, which bridges together the VC-dimension, a distribution and network independent quantity, with the statistical epsilon-capacity [7], which depends on the distribution of samples and choices of weights. The relationship indicates that generalization happens after memorization, and provides a general constructive methodology for finding a lower bound for the VC-dimension of a class of networks of interest in general. It is applied to find a lower bound on the VC-dimension of two layer networks with binary weights and integer thresholds. In Chapter 3, as another application of this general relation, how the sample bound for generalization varies with respect to distributions is studied through analyzing a specific network which classifies samples belonging to two binary clusters. Chapter 4 contributes to evaluations of the capacity of two layer networks with binary weights and integer thresholds.

The algorithmic part concentrates on the development of learning algorithms which can actually find networks that generalize well. Guidelines for these algorithms are provided by the implications of the theoretical results, which suggest that in order to obtain an optimal network that can approximate the underlying mapping, it is necessary to search for an appropriate network structure during training. Particularly, in Chapter 5, a network reduction algorithm and its modified versions are developed to deal with learning analog mappings. The algorithm is applied to training a network to control a two-link manipulator which draws characters. A network addition-deletion algorithm is developed in Chapter 6, which allows the search of network structures in a general fashion by finding number of layers, number of units on each layer and possible local connectivity patterns. Generalization in terms of learning binary mappings is investigated in this chapter by applying the algorithm on various classification problems including hand written character recognitions.

Finally, the conclusion summarizes the results and states some open problems and possible future directions.

# Bibliography

[1] E. Baum and D. Haussler,“ What Size Net Gives Valid Generalization?” *Neural Computation*, 1(1), 151-160, 1989.

[2] G. Cybenko, “ Approximations by Superpositions of A Sigmoidal Function,” Math. Control, Signals, Systems, Vol. 2, 303-314, 1989

[3] R.J . McEliece, E.C . Posner, E.R . Rodemich, S.S . Venkatesh, “The Capacity of the Hopfield Associative Memory,” *IEEE Trans. Inform. Theory*, Vol. IT-33, No. 4, 461-482, July 1987.

[4] L. Valliant, “ A Theory of Learnable,” *CACM 27*, 1134-1142, 1984.

[5] V.N . Vapnik and A.Ya. Chervonenkis, “On Uniform Convergence of Relative Frequency of Event to Their Probabilities,” *Theor. Probability Appl.*, 16(1971), 264-280.

[6] S.S . Venkatesh, *Ph.D. Thesis, Caltech*, 1986.

[7] S.S . Venkatesh and D. Psaltis, “Epsilon Capacity of Neural Networks,” to appear on *IEEE Trans. on Information Theory.*

[8] K. Hornik, M, Stinchcombe and H. White, “ Multi-layer Feedforward Networks Are Universal Approximators,” to appear in *Neural Networks.*

# Chapter 2

# The VC-Dimension vs. The Statistical Capacity of Multilayer Networks with Binary Weights

## 2.1 Introduction

The information capacity and the VC-dimension are two important quantities that characterize multilayer feedforward neural networks. The former characterizes their memorization capability, while the latter represents the sample complexity needed for generalization. Although intuitively they seem to be interrelated, the precise relationships are basically unknown. Discovering those relationships is of importance for obtaining a better understanding of the fundamental properties of multilayer networks in learning and generalization.

In this work we show that the VC-dimension of feedforward multilayer neural networks, which is a quantity independent of the distribution of the samples as well as choices of network parameters, can be lower bounded (in order) by a distribution and network dependent quantity, the statistical lower epsilon-capacity $C_\epsilon^-$, when the samples are drawn from two classes: $\Omega_1(+1)$ and $\Omega_2(-1)$. The only requirement on the distribution from which samples are drawn is that the optimal classification error achievable, the Bayes error $P_{be}$, is greater than zero. Then we will show that the VC-dimension $d$ and the statistical lower epsilon-capacity $C_\epsilon^-$ are related by

$$C_\epsilon^- \leq Ad, \tag{2.1}$$

where $\epsilon = P_{eo} - \epsilon'$ for $0 < \epsilon' \leq P_{eo}$; or $\epsilon = P_{be} - \epsilon'$ for $0 < \epsilon' \leq P_{be}$. Here $P_{eo}$ represents the optimal error rate achievable on the class of classifiers considered. It is obvious that $P_{eo} \geq P_{be}$. The relation given in equation (2.1) is non-trivial if $P_{be} > 0$, $P_{eo} \leq \epsilon'$ or $P_{be} \leq \epsilon'$ so that $\epsilon$, the error tolerance, is a non-negative quantity. $Ad$ is called the universal sample bound for generalization, where $A < \frac{128 ln \frac{1}{\epsilon}}{\epsilon'^2}$ is a positive constant. That is, when the sample complexity exceeds $Ad$, all the networks of the same architecture for all distributions of the samples can generalize with almost probability 1 for $d$ large. A special case of interest, in which $P_{be} = \frac{1}{2}$, corresponds to random assignments of samples. In this case, the class conditional distributions of the samples completely overlap with each other. Then $C_\epsilon^-$ characterizes the memorizing capability of networks. Therefore, the relationship in equation (2.1) relates the sample complexity that is necessary for generalization with three things: the random lower epsilon-capacity of a network, which indicates that generalization happens after memorization; the statistical lower epsilon-capacity of the optimal classifier in a class of classifiers; and the Baysian classifier, which implies the importance of appropriate choices of network structures.

Although the VC-dimension is a key parameter in generalization, there exists no systematic way of finding it. The relationship we have obtained, however, brings concomitantly a constructive method of finding a lower bound for the VC-dimension of multilayer networks. That is, if the weights of a network are properly constructed using random samples drawn from a chosen distribution, the statistical lower epsilon-capacity can be evaluated and then utilized as a lower bound for the VC-dimension. In this paper we will show how this constructive approach contributes to finding a lower bound of the VC-dimension of two layer $(N-2L-1)$ networks with binary interconnections, where $N$ and $L$ are the number of input and hidden units respectively.

It has been shown by Baum and Haussler [1] that the VC-dimension $d_{2r}$ for two layer $(N-L-1)$ feedforward networks with analog weights satisfies the relation $O(W) \leq d_{2r} \leq O(WlnL)$, where $W$ is the total number of weights and $L$ is the number of hidden units. Discrete weights, however, are of primary interest for hardware implementations. Furthermore , the number of bits needed to describe weight values is one of the important quantities to characterize network complexity. Although such networks with discrete weights have been implemented successfully in hardware and various simulations have been done, very few theoretical results exist so far concerning the generalization of multilayer networks with discrete weights. In what follows we show that the general relationship between the VC-dimension and the statistical lower epsilon-capacity can be applied to show that a lower bound for the VC-dimension of $(N-2L-1)$ networks with binary weights is $O(\frac{W}{lnL})$. Since it can be easily shown that the VC-dimension $d_b$ is upper-bounded by $O(W)$, $d_b$ satisfies the relation $O(\frac{W}{lnL}) \leq d_b \leq O(W)$.

## 2.2 A Relationship Between the VC-Dimension and the Statistical Capacity

### 2.2.1 The Statistical Lower Epsilon-Capacity $C_\epsilon^-$

The concept of the statistical lower epsilon-capacity was introduced by McEliece et al. [10] for the Hopfield associative memory. It was later extended to the concept of the epsilon-capacity by Venkatesh and Psaltis [15] who considered a single threshold element in which an epsilon fraction of samples were allowed to be classified incorrectly. For multilayer feedforward networks, the statistical lower epsilon-capacity is defined similarly as follows.

**Definition 2.1** *Consider a network $s$ whose weights are constructed from $M$ random samples belonging to two classes. Let $\hat{r}(s) = \frac{Z}{M}$, where $Z$ is the total number of samples classified incorrectly by the network $s$. Let the random variable $\hat{r}(s)$ is the training error rate. Let*

$$P_\epsilon(M) = \Pr(\hat{r}(s) \leq \epsilon), \tag{2.2}$$

*where $0 \leq \epsilon \leq 1$. Then the statistical lower epsilon-capacity $C_\epsilon^-$ is the maximum $M$ such that $P_\epsilon(M) \geq 1 - \eta$, where $\eta$ can be arbitrarily small for sufficiently large $N$.*

### 2.2.2 The VC-Inequality and A Universal Bound for Generalization

Consider a class $S$ of feedforward networks with a fixed structure which consist of threshold elements and one output unit. Then the Vapnik-Chervonenkis inequality [11] can be expressed as

$$\Pr(\sup_{s\in S} \mid \hat{r}(s) - P_e(x|s) \mid > \epsilon') \leq 4\Phi(2M,d)e^{-\frac{\epsilon'^2 M}{8}}, \tag{2.3}$$

where $\Phi(2M,d) = \sum_{i=1}^{d}\binom{2M}{i}$, $d$ is the VC-dimension of $S$, and $s$ is one network in $S$. $P_e(x|s)$ is the true probability of incorrect classification of a new sample $x$ by the network $s$. $\hat{r}(s)$, which is an estimator of $P_e(x|s)$, is the ratio between the number of samples classified incorrectly by $s$ to the total $M$ samples drawn independently from some distribution. Since $\Phi(2M,d) = 2^{2M}$ for $2M \leq d$; and $\Phi(2M,d) < 1.5\frac{(2M)^d}{d!}$ for $2M > d$ [13], we have

$$\Pr(\sup_{s\in S} \mid \hat{r}(s) - P_e(x|s) \mid > \epsilon') \leq h(2M;d,\epsilon'), \tag{2.4}$$

where

$$h(2M;d,\epsilon') = \begin{cases} 1; & \text{if either } 2M \leq d, \text{ or } 6\frac{(2M)^d}{d!}e^{-\frac{\epsilon'^2 M}{8}} \geq 1 \text{ for } 2M > d, \\ 6\frac{(2M)^d}{d!}e^{-\frac{\epsilon'^2 M}{8}}; & \text{otherwise.} \end{cases}$$

In the following theorem we will show that $h(2M;d,\epsilon')$ has one sharp transition point occurring at $O(d)$ for large $d$. When the number of samples $M$ exceeds this critical point, $h(2M;d,\epsilon')$ is almost zero. That is, $\mid \hat{r}(s) - P_e(x|s) \mid \leq \epsilon'$ with almost probability one, for all $s \in S$, which represents the occurrence of generalization.

**Theorem 2.1** *For large d*

$$h(2M;d,\epsilon') \begin{cases} = 1, & \text{if } M \leq Ad \\ \leq Be^{-\alpha d}, & \text{if } M \geq Ad \end{cases}$$

*where the constant $A > \frac{1}{2}$ satisfies the equation*

$$ln(2A) + 1 - \frac{\epsilon'^2}{8}A = 0, \tag{2.5}$$

$B \approx \frac{1.5}{\sqrt{2\pi}}$ *and* $\alpha \approx \gamma ln2A$.

Proof:

a). For $2M \leq d$, $h(2M; d, \epsilon') = 1$, since $\Phi(2M, d) = e^{Mln4}$ which is bigger than $e^{-\frac{\epsilon'^2 M}{8}}$ for $\epsilon' > 0$.

b). For $2M > d$ and $d$ large, by Stirling's formula, we have $d! \sim \sqrt{2\pi} e^{-d} d^{d+\frac{1}{2}}$. Then

$$1.5\frac{(2M)^d}{d!} e^{-\frac{\epsilon'^2 M}{8}} \sim \frac{1.5}{\sqrt{2\pi}} e^{dg(M;d,\epsilon')}, \tag{2.6}$$

where

$$g(M; d, \epsilon') = lnM - lnd + 1 + ln2 - \frac{\epsilon'^2 M}{8d}. \tag{2.7}$$

Let $M_0 = A'd$ and insert it into equation (2.7), where $A'$ is a constant. We then obtain

$$\begin{aligned} f(A') &= g(M_0; d, \epsilon') \\ &= lnA' + 1 + ln2 - \frac{\epsilon'^2 A'}{8}, \end{aligned} \tag{2.8}$$

for $A' > \frac{1}{2}$. Since $d$ is assumed large, we show in what follows, that for $A' > A$, where $A > \frac{1}{2}$ is the unique root of $f(A') = 0$, that $f(A')$ is negative. That is, $e^{dg(M_0;d,\epsilon')}$ decreases exponentially in $d$.

The uniqueness of the root in $(\frac{1}{2}, +\infty)$ for $f(A') = 0$ is true since $f(A')$ is convex and has a unique maximum $f(\frac{8}{\epsilon'^2}) = ln(\frac{16}{\epsilon'^2})$, where $f(A') \to -\infty$ when $A' \to +\infty$ and $f(\frac{1}{2}) = 1 - \frac{\epsilon'^2}{16}$, which is greater than zero. It is easy to check that this root is in $(\frac{16ln\frac{1}{\epsilon'}}{\epsilon'^2}, \frac{128ln\frac{1}{\epsilon'}}{\epsilon'^2})$. For any given $\gamma > 0$ small, when $M = (1+\gamma)Ad$ $e^{d(lnM - lnd + 1 + ln2 - \frac{\epsilon'^2 M}{8d})} \approx e^{-\alpha d}$, with $\alpha = \frac{\epsilon'^2 \gamma A}{8} - ln(1+\gamma)$, which is approximately equal to $\gamma lnA > 0$. Then for $M \geq (1+\gamma)Ad$, $h(2M; d, \epsilon') \leq Be^{-\alpha d}$, where $B \approx \frac{1.5}{\sqrt{2\pi}}$.

Q.E.D

Since the VC-inequality holds for all sample distributions and all the networks of the same structure, we call this sharp transition point $Ad$ the universal sample bound for generalization.

### 2.2.3 A Relationship between The VC-Dimension and $C_\epsilon^-$

To relate the VC-dimension, which is a distribution-free and network-parameter-independent quantity, with the statistical lower epsilon-capacity which depends on the distribution of samples and weights, we take two major steps shown in the theorem below. One is to focus on a special distribution; another is to consider a specific network. Then the relation can be drawn through a connection between the two sharp transitions characterized by the statistical lower epsilon-capacity and the universal sample bound for generalization.

**Theorem 2.2** *Let samples belonging to two classes $\Omega_1(+1)$ and $\Omega_2(-1)$ be drawn independently from some distribution. The only requirement on the distributions considered is that the Bayes error $P_{be}$ satisfies $0 < P_{be} \leq \frac{1}{2}$. Let $S$ be a class of feedforward multilayer networks with a fixed structure consisting of threshold elements and $s_1$ be one network in $S$, where the weights of $s_1$ are constructed from $M$ (training) samples drawn from one distribution as specified above. For a given distribution, let $P_{eo}$ be the optimal error rate achievable on $S$ and $P_{be}$ be the Bayes error rate. Then*

$$\Pr(\hat{r}(s_1) < P_{eo} - \epsilon') \leq h(2M; d, \epsilon'), \tag{2.9}$$

*and*

$$\Pr(\hat{r}(s_1) < P_{be} - \epsilon') \leq h(2M; d, \epsilon'), \tag{2.10}$$

*where $\hat{r}(s_1)$ is equal to the training error rate of $s_1$. (It is also called the resubstitution error estimator in the pattern recognition literature.) These relations are nontrivial if $P_{eo} > \epsilon'$, $P_{be} > \epsilon'$ and $\epsilon' > 0$ small.*

Proof:

Since the VC-inequality holds for all distributions of samples, it is certainly valid for the class of distributions with $P_{be} > 0$. Then for one distribution in this class, we have, for any given $\epsilon' > 0$ and all the networks in $S$,

$$\Pr(\sup_{s \in S} \mid \hat{r}(s) - P_e(x|s) \mid > \epsilon') \leq h(2M; d, \epsilon'). \tag{2.11}$$

Furthermore, for any particular $s_1 \in S$,

$$\mid \hat{r}(s_1) - P_e(x|s_1) \mid \leq \sup_{s \in S} \mid \hat{r}(s) - P_e(x|s) \mid, \tag{2.12}$$

so that

$$\Pr(\mid \hat{r}(s_1) - P_e(x|s_1) \mid > \epsilon') \leq \Pr(\sup_{s \in S} \mid \hat{r}(s) - P_e(x|s) \mid > \epsilon'). \tag{2.13}$$

It is noted that $s_1$ is constructed using $M$ samples and $\hat{r}(s_1)$ is the error estimator evaluated using the same sample set.

The event $\mid \hat{r}(s_1) - P_e(x|s_1) \mid > \epsilon'$ occurs when one of the following cases is true: $\hat{r}(s_1) < P_e(x|s_1) - \epsilon'$ or $\hat{r}(s_1) > P_e(x|s_1) + \epsilon'$. Since these two events are exclusive, we have

$$\begin{aligned} \Pr(\mid \hat{r}(s_1) - P_e(x|s_1) \mid > \epsilon') &= \Pr(\hat{r}(s_1) < P_e(x|s_1) - \epsilon') \\ &+ \Pr(\hat{r}(s_1) > P_e(x|s_1) + \epsilon'). \end{aligned} \tag{2.14}$$

Then

$$\Pr(\hat{r}(s_1) < P_e(x|s_1) - \epsilon') \leq \Pr(\mid \hat{r}(s_1) - P_e(x|s_1) \mid > \epsilon'). \tag{2.15}$$

Furthermore, the relation $P_e(x|s_1) \geq P_{eo} \geq P_{be}$ is always true since $P_{eo}$ is the optimal probability of error achievable on $S$ and $P_{be}$ is the Bayes error rate which is optimal for all classifiers for the given distribution. Therefore

$$\begin{aligned} \Pr(\hat{r}(s_1) < P_{eo} - \epsilon') &\leq \Pr(\hat{r}(s_1) < P_e(x|s_1) - \epsilon') \\ &\leq h(2M; d, \epsilon'), \end{aligned} \tag{2.16}$$

and

$$\begin{aligned}\Pr(\hat{r}(s_1) < P_{be} - \epsilon') &\leq \Pr(\hat{r}(s_1) < P_e(x|s_1) - \epsilon') \\ &\leq h(2M; d, \epsilon'), \end{aligned} \tag{2.17}$$

where $h(2M; d, \epsilon')$ is given in Theorem 2.1, and $d$ is the VC-dimension.

Q.E.D.

Since $\Pr(\hat{r}(s_1) < P_{be} - \epsilon')$ has the same form as given by equation (2.2) in defining the statistical lower epsilon-capacity, Theorem 2.2 suggests that the statistical lower epsilon-capacity and the VC-dimension can be quantitatively related through a relation shown in the theorem below.

**Theorem 2.3** *Asymptotically (for a large network such that $M$ and $d$ as functions of the network size are also large), the lower epsilon-capacity $C_\epsilon^-$ of a network $s_1$ is a lower bound for the VC-Dimension. Specifically,*

$$C_\epsilon^- \leq Ad, \tag{2.18}$$

*where $Ad$ is the universal sample bound given in Theorem 2.1 for $A < \frac{128 ln \frac{1}{\epsilon'}}{\epsilon'^2}$, $\epsilon = P_{eo} - \epsilon'$ for $0 < \epsilon' \leq P_{eo}$, or $\epsilon = P_{be} - \epsilon'$ for $0 < \epsilon' \leq P_{be}$. That is, $C_\epsilon^-$ is at most of the same order as $d$.*

Proof:

Assume $C_\epsilon^- > Ad$, where either $\epsilon = P_{eo} - \epsilon'$ or $\epsilon = P_{be} - \epsilon'$. Then by the definition of the lower epsilon-capacity, either $\Pr(\hat{r}(s_1) < P_{eo} - \epsilon') \geq 1 - \eta$ or $\Pr(\hat{r}(s_1) < P_{be} - \epsilon') \geq 1 - \eta$, where $\eta$ is sufficiently small when the size of the network is sufficiently large. However, from equation (2.4), we have either $\Pr(\hat{r}(s_1) < P_{eo} - \epsilon') < \eta$ or $\Pr(\hat{r}(s_1) < P_{be} - \epsilon') < \eta$, where $\eta \leq O(e^{-\alpha d})$ which is small when $M > Ad$. Then a contradiction occurs. Therefore, we must have $C_\epsilon^- \leq Ad$.

Q.E.D.

To interpret this relation, let us examine the range of $\epsilon$ and $\epsilon'$ in equation (2.18). Since $\epsilon'$, which is initially given in inequality (2.3), represents the generalization error between the training error rate and the true probability of error, it is usually quite small. For most of practical problems, $P_{be}$ is small also. If the structure of the class of networks is properly chosen so that $P_{eo} \approx P_{be}$, then $\epsilon = P_{eo} - \epsilon'$ will be a small quantity. Although the epsilon-capacity is a valid quantity depending on $M$ for any network in the class, for $M$ sufficiently large, the meaningful networks to be considered through this relation is only a small subset in the class whose true probability of error is close to $P_{eo}$. That is, this small subset contains only those networks which can approximate the best classifier contained in this class.

For a special case in which samples are assigned randomly to two classes with equal probability, we have a result stated in Corollary 2.1.

**Corollary 2.1** *Let samples be drawn independently from some distribution and then assigned randomly to two classes $\Omega_1(+1)$ and $\Omega_2(-1)$ with equal probability. This is equivalent to the case that the two class conditional distributions have complete overlap with one another. That is, $\Pr(x \mid \Omega_1) = \Pr(x \mid \Omega_2)$. Then the Bayes error is $\frac{1}{2}$. Using the same notation as in the above theorem, we have*

$$C^{-}_{\frac{1}{2}-\epsilon'} \leq Ad. \tag{2.19}$$

Although the distributions specified here give an uninteresting case for classification purposes, we will see later that the random statistical epsilon-capacity in inequality (2.19) can be used to characterize the memorizing capability of networks, and to formulate a constructive approach to find a lower bound for the VC-dimension.

## 2.3 A Lower Bound of the VC-Dimension of Two Layer Networks with Binary Weights

### 2.3.1 A Constructive Methodology

One of the applications of this relation is that it provides a general constructive approach to find a lower bound for the VC-dimension for a class of networks. Specifically, using the relationship given in equation (2.19), the procedures can be described as follows.

1). Select a distribution.

2). Draw samples independently from the chosen distribution, and then assign them randomly to two classes.

3). Evaluate the lower epsilon-capacity and then use it as a lower bound for the VC-dimension.

In this section we give an example that demonstrates how this general approach can be applied to find a lower bound for the VC-dimension. Specifically, a statistical approach is used to find a lower bound for the lower epsilon-capacity of a two-layer network with binary weights as a lower bound for the corresponding VC-dimension.

### 2.3.2 Construction of The Network

The construction we consider is motivated by the one used by Baum in finding the capacity for two layer networks with analog weights. Such a network whose weights and thresholds required arbitrary accuracy can dichotomize any assignment of $NL$ samples in general position in $N$ dimensional space, which gives a lower bound for the

VC-dimension for such networks with real weights. Although this particular network will fail if the accuracy of the weights and the thresholds is reduced, the idea of using the grandmother-cell type of network can be adopted to construct our network.

We consider a two layer binary network with $2L$ hidden threshold units and one output threshold unit as shown in Figure 2.1.

The weights at the second layer are fixed and equal to $+1$ and $-1$ alternately. The hidden units are allowed to have integer thresholds in $[-N, N]$, and the threshold for the output unit is zero. The weights at the first layer are constructed from random samples as described as follows. Let $\vec{X}_l^{(m)} = (x_{l1}^{(m)}, ..., x_{lN}^{(m)})$ be a $N$ dimensional random vector, where $x_{li}^{(m)}$'s are independent random variables taking $(+1)$ and $(-1)$ with equal probability $\frac{1}{2}$, $0 \leq l \leq L$, and $0 \leq m \leq M$.

Consider the $l$th pair of hidden units. The weights at the first layer for this pair of hidden units are equal and constructed using $M$ samples which are assigned randomly to the same class. Let $w_{li}$ denote the weight from the $i$th input to these two hidden units

$$w_{li} = sgn(\alpha_l \sum_{m=1}^{M} x_{li}^{(m)}), \tag{2.20}$$

where $sgn(x) = 1$ if $x > 0$, and $-1$ otherwise. $\alpha_l$'s , $1 \leq l \leq L$, are independent random variables which take on two values $+1$ or $-1$ with equal probability. They represent the assignments of the $LM$ samples into two classes $\Omega_1(+1)$ and $\Omega_2(-1)$.

The thresholds for these two units are different and are given as

$$t_{l\pm} = \alpha_l \lfloor (1 \mp k)\sqrt{\frac{2}{\pi}}\frac{N}{\sqrt{M}} \rfloor, \tag{2.21}$$

where $t_{l\pm}$ correspond to the thresholds for the units with weight $+1$ and $-1$ at the

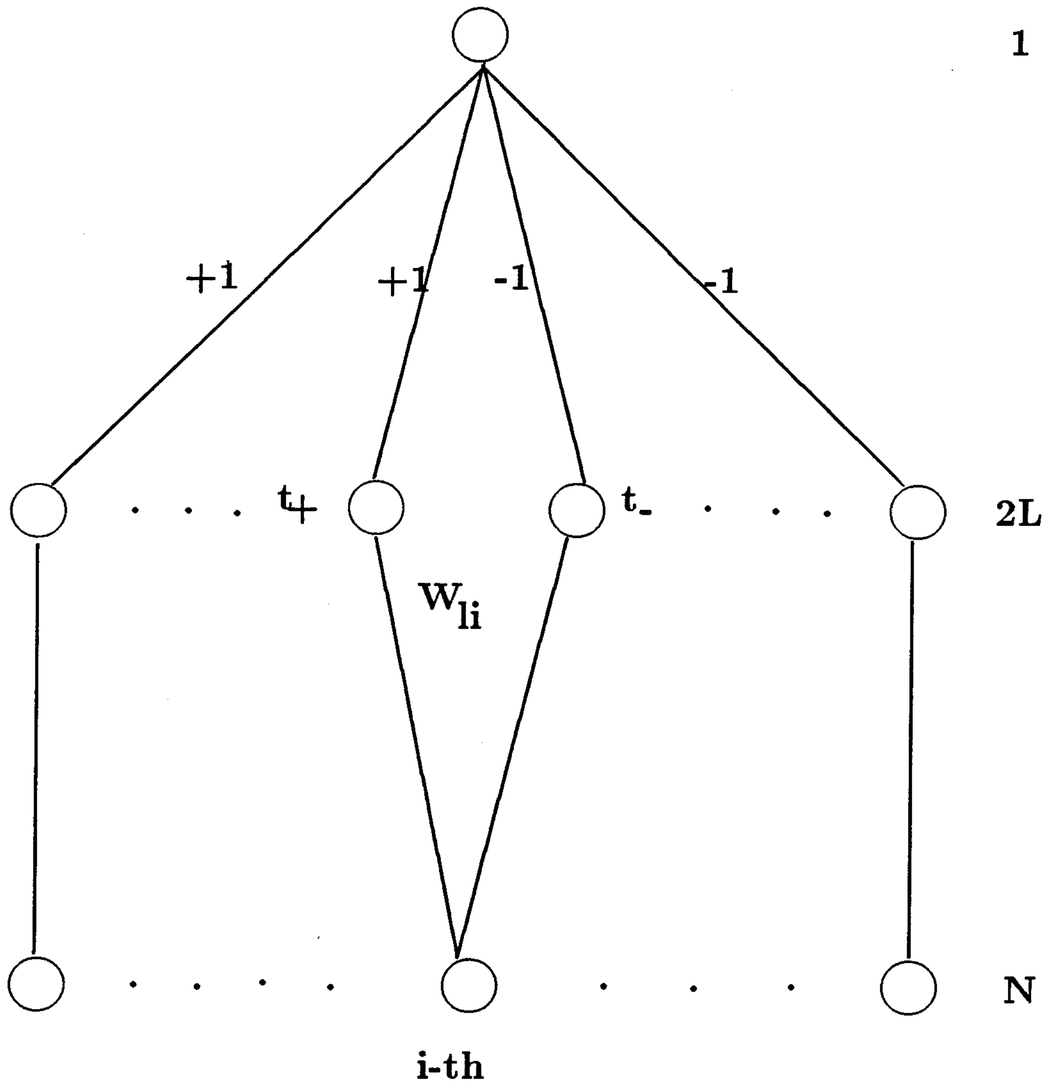


Figure 2.1: Two layer networks with binary weights and integer thresholds

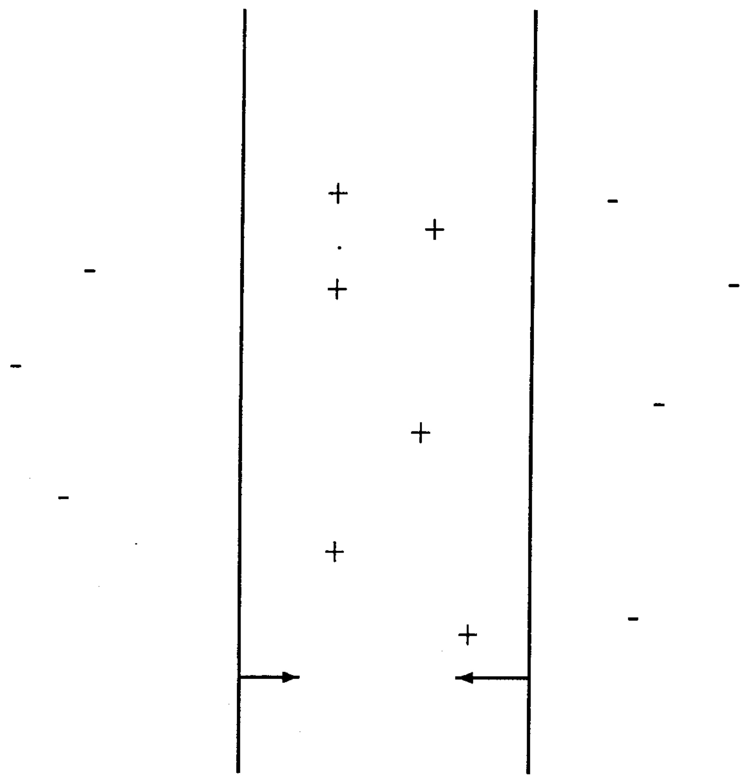


Figure 2.2: Two Parallel Hyperplanes Formed by One Pair of Hidden Units.

second layer respectively. Since the samples, which are independent and equally probable binary random variables, are randomly assigned to two classes, the Bayes error is $\frac{1}{2}$ in this case.

Before we go into the rigorous analysis to find a lower epsilon-capacity, we first explain how the constructed network roughly works. Consider a set of given samples and assignments. As shown in Figure 2.2, each pair of hidden units forms two parallel hyperplanes separated by the two thresholds. When $\alpha_l = 1$, this pair will have a presynaptic input $+2$ to the output unit for the samples stored in this pair which fall in between the planes represented by "$+$"s in the figure, since they lie on the positive sides of both planes. The other pairs of hidden units will have the presynaptic inputs approximately 0 to the output unit to most of these samples which are not stored in those pairs represented by "-"s in the figure, and thus will most likely fall outside of those pairs of parallel planes. When $\alpha_l = -1$, a similar outcome will occur except $+2$ will be replaced by $-2$. When the samples are random, with a certain probability they will fall either between a pair of parallel hyperplanes or outside. That is why we need to go through such a statistical analysis.

### 2.3.3 Probability of Error for One Sample

As the first step in evaluating the epsilon-capacity, we compute $P_{e1}$, which is the probability of incorrect classification of one random sample stored. By the construction of the network, the total presynaptic input to the output unit due to the $j$-th pair of hidden units, which is the difference between the outputs of two threshold hidden units, can have two possible values $+2$ or $0$ for $\alpha_l = 1$. For $\alpha_l = -1$, this pair can take the values $-2$ or $0$. Since $\alpha_l$ is random and can take $1$ and $-1$ with equal probability, each pair can take three possible values $(+2)$, $(-2)$ and $0$, for $1 \leq l \leq L$. Without loss of generality, we now consider the case that $\vec{X}_1^{(1)}$ is fed through the network and $\alpha_1$ is given to be 1. Let $y_1^{(1)}$ denote the output of the network. Then $P_{e1} = \Pr(\alpha_1 y_1^{(1)} < 0)$. If $y_1^{(1)} = 0$, $y_1^{(1)}$ is assigned to either class 1 or class 2 with equal probability $\frac{1}{2}$. Let $s_l^{(1)}$ be the presynaptic input to the output unit due to the $l$th pair of hidden units. And let $n_+^{(1)}$ and $n_-^{(1)}$ be the number of $+2$'s and $-2$'s of the presynaptic inputs of the output units given by the pairs with $2 \leq l \leq L$. Then

$$\begin{aligned}
P_{e1} &= \Pr(y_1^{(1)} = -1) + \frac{1}{2}\Pr(y_1^{(1)} = 0) \\
&= \Pr(s_1^{(1)} = 2)\Pr(n_-^{(1)} - n_+^{(1)} \geq 2) + \Pr(s_1^{(1)} = 0)\Pr(n_-^{(1)} - n_+^{(1)} \geq 1) \\
&+ \frac{1}{2}\Pr(s_1^{(1)} = 2)\Pr(n_-^{(1)} - n_+^{(1)} = 1) \\
&+ \frac{1}{2}\Pr(s_1^{(1)} = 0)\Pr(n_-^{(1)} - n_+^{(1)} = 0). \qquad (2.22)
\end{aligned}$$

To obtain an explicit expression for $P_{e1}$, we need to evaluate each individual term in the above equation.

The presynaptic input to the two hidden units in the first pair equals

$$z_1^{(1)} = \sum_{i=1}^{N} w_{1i} x_{1i}^{(1)}. \tag{2.23}$$

Since different terms in the summation (2.23) are independent, by the large deviation theorem [4], for $N$ sufficiently large, with $\mid t_{1+} - \mathbf{E}z_1^{(1)} \mid < O(N^{\frac{1}{2}})$ and $\mid t_{1-} - \mathbf{E}z_1^{(1)} \mid < O(N^{\frac{1}{2}})$, we have

$$\begin{aligned} \Pr(s_1^{(1)} = 2) &= \Pr(t_{1+} < z_1^{(1)} < t_{1-}) \\ &= 1 - 2Q(-k\sqrt{\frac{N'}{M}}), \end{aligned} \tag{2.24}$$

$$\Pr(s_1^{(1)} = 0) = 2Q(-k\sqrt{\frac{N'}{M}}), \tag{2.25}$$

where $N' = \frac{2}{\pi}N$ and $Q(-x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-x} e^{\frac{-u^2}{2}} du$.

Similarly we can obtain for $2 \le l \le L$

$$\begin{aligned} \Pr(s_l^{(1)} = 2) &= \Pr(t_{1-} < z_1^{(1)} < t_{1+}) \\ &= Q(-(1-k)\sqrt{\frac{N'}{M}}) - Q(-(1+k)\sqrt{\frac{N'}{M}}), \end{aligned} \tag{2.26}$$

$$\begin{aligned} \Pr(s_l^{(1)} = -2) &= \Pr(s_l^{(1)} = -2 | \alpha_l = -1)\Pr(\alpha_l = -1) \\ &= \Pr(s_l^{(1)} = 2), \end{aligned} \tag{2.27}$$

$$\Pr(s_l^{(1)} = 0) = 1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}}). \tag{2.28}$$

Let $n_0^{(1)}$ denote the number of pairs which have zero presynaptic inputs to the output neuron. Then due to the mutual independence of $s_l^{(1)}$ for $2 \le l \le L$, the event

$(n_0^{(1)} = k_0, n_-^{(1)} = k_1, n_+^{(1)} = k_2)$ has a multinomial distribution, i.e.

$$\begin{aligned} &\Pr(n_0^{(1)} = k_0, n_-^{(1)} = k_1, n_+^{(1)} = k_2) \\ = \; &\frac{(L-1)!}{k_0!k_1!k_2!}\Pr^{k_0}(s_2^{(1)} = 0)\Pr^{k_1}(s_2^{(1)} = -2)\Pr^{k_2}(s_2^{(1)} = 2), \end{aligned} \tag{2.29}$$

where $k_0, k_1$ and $k_2$ are non-negative integers, and we can choose $l = 2$ without loss of generality. For $M \ll N$, $L$ sufficiently large and $L\Pr(s_l^{(1)} = 2) \overset{L\to\infty}{\longrightarrow} c$, for $2 \le l \le L$, where c is a constant, the multinomial distribution can be approximated by the Multiple Poisson distribution [4]. That is,

$$\Pr(n_-^{(1)} = k_1, n_+^{(1)} = k_2) = e^{-2\lambda}\frac{\lambda^{(k_1+k_2)}}{k_1!k_2!}, \tag{2.30}$$

where $0 < k < 1$, and

$$\lambda = \frac{(L-1)}{2}[Q(-(1-k)\sqrt{\frac{N'}{M}}) - Q(-(1+k)\sqrt{\frac{N'}{M}})]. \tag{2.31}$$

When $L$ is sufficiently large

$$\begin{aligned} \Pr(n_-^{(1)} = n_+^{(1)}) &= e^{-2\lambda}\sum_{k=0}^{\lfloor\frac{L-1}{2}\rfloor}\frac{\lambda^{2k}}{k!^2} \\ &\approx e^{-2\lambda}I_0(2\lambda), \end{aligned} \tag{2.32}$$

$$\begin{aligned} \Pr(n_-^{(1)} = n_+^{(1)} + 1) &= e^{-2\lambda}\sum_{k=0}^{\lfloor\frac{L-1}{2}\rfloor}\frac{\lambda^{2k+1}}{k!(k+1)!} \\ &\approx e^{-2\lambda}I_1(2\lambda), \end{aligned} \tag{2.33}$$

since

$$\sum_{k=0}^{\infty}\frac{\lambda^{2k}}{k!(k+n)!} = \lambda^{-n}I_n(2\lambda), \tag{2.34}$$

where $I_n(x) = i^{-n}J_n(ix)$ for $x$ being a real variable and $J_n(x)$ the $n$th order Bessel function.

Because

$$\Pr(n_-^{(1)} - n_+^{(1)} \ge 2) = \frac{1}{2}[1 - \Pr(n_-^{(1)} = n_+^{(1)}) - 2\Pr(n_-^{(1)} - n_+^{(1)} = 1)], \tag{2.35}$$

$$\Pr(n_-^{(1)} - n_+^{(1)} \ge 1) = \frac{1}{2}[1 - \Pr(n_-^{(1)} = n_+^{(1)})], \tag{2.36}$$

plugging equations (2.36) (2.35) (2.28) (2.26) into the equation (4.3), we can obtain

$$\begin{aligned} P_{e1} &= \frac{1}{2} - \frac{1}{2}\Pr(n_-^{(1)} = n_+^{(1)}) - \frac{1}{2}\Pr(n_-^{(1)} = n_+^{(1)} + 1) \\ &+ Q(-k\sqrt{\frac{N'}{M}})[\Pr(n_-^{(1)} = n_+^{(1)} + 1) + \Pr(n_-^{(1)} = n_+^{(1)})]. \end{aligned} \tag{2.37}$$

When $\lambda$ is large, we have the approximation [5]

$$I_n(2\lambda) \sim \frac{e^{2\lambda}}{\sqrt{4\pi\lambda}}. \tag{2.38}$$

Using this result in equation (2.37), we have

$$P_{e1} = \frac{1}{2} - \frac{1}{\sqrt{4\pi\lambda}} + 2Q(-k\sqrt{\frac{N'}{M}})\frac{1}{\sqrt{4\pi\lambda}} + o(\frac{1}{\sqrt{\lambda}}), \tag{2.39}$$

where $\lambda = \frac{(L-1)}{2}[Q(-(1-k)\sqrt{\frac{N'}{M}}) - Q(-(1+k)\sqrt{\frac{N'}{M}})]$, and $0 < k < 1$.

We will show in the next section how $P_{e1}$ can be used in evaluating the lower epsilon-capacity.

## 2.4 Probability of Error for All Samples Stored

Let

$$\hat{r} = \frac{1}{LM}\sum_{l=1}^{L}\sum_{m=1}^{M} I(A_{lm}), \tag{2.40}$$

where $I(A) = 1$ if event A occurs, and $I(A) = 0$ otherwise. $A_{lm}$ represents the event that the $m$th sample stored at the $l$th pair is classified incorrectly. $\vec{X}_l^{(m)}$ is classified incorrectly with probability $P_{e1}$, if $\alpha_l y_l^{(m)} < 0$; and with probability $\frac{1}{2}$ if $y_l^{(m)} = 0$. Then $\hat{r}$, used as before, is a random variable representing the training error rate. To evaluate the statistical epsilon-capacity for this network, we need to consider the probability $\Pr(\hat{r} < \epsilon)$ for $\epsilon = \frac{1}{2} - \epsilon'$, which is the probability that no more than $(\frac{1}{2} - \epsilon')LM$ out of $LM$ samples stored are classified incorrectly. Let

$$Z_L = \frac{1}{\sqrt{L}}\sum_{l=1}^{L} Z_{Ml}, \tag{2.41}$$

where $Z_{Ml} = \frac{1}{\sqrt{M}}\sum_{m=1}^{M} Y_l^{(m)}$ for $Y_l^{(m)} = \frac{1}{\sigma}(I(A_{lm}) - P_{e1})$, $\sigma^2 = P_{e1}(1 - P_{e1})$. Then $\Pr(Y < \epsilon LM) = \Pr(Z_L < \hat{\epsilon}\sqrt{LM})$, where $\hat{\epsilon} = \frac{1}{\sigma}(\frac{1}{2} - \epsilon' - P_{e1})$. To find a lower epsilon-capacity, we need to use a large-deviation theorem to evaluate the probability $\Pr(Z_L < \hat{\epsilon}\sqrt{LM})$. Since for any $l \in [1, L]$, $Z_{Ml}$ is a summation of $M$ dependent random variables, we first show in the following lemma that the joint distributions of the dependent random varibles $\{I(A_{lm})\}$ for $m \in [1, M]$ can be approximated by the products of individual distributions plus a term which is negligible.

**Lemma 2.1** *When $M \sim O(\frac{N}{lnL})$ and $L$, $N$ large ($L$ and $N \to \infty$), we have*

$$\begin{aligned} P(1,1) &= \Pr(I(A_{lm_1}) = 1, I(A_{lm_2}) = 1) \\ &= P_{e1}^2 + O(\frac{1}{M^2}), \end{aligned} \tag{2.42}$$

$$\begin{aligned} P(1,1,1) &= \Pr(I(A_{lm_1}) = 1, I(A_{lm_2}) = 1, I(A_{lm_3}) = 1) \\ &= P_{e1}^3 + O(\frac{1}{M^2}), \end{aligned} \tag{2.43}$$

$$\begin{aligned} P(1,1,1,1) &= \Pr(I(A_{lm_1}) = 1, I(A_{lm_2}) = 1, I(A_{lm_3}) = 1, I(A_{lm_4}) = 1) \\ &= P_{e1}^4 + O(\frac{1}{M^2}), \end{aligned} \tag{2.44}$$

*where $P_{e1} = \Pr(I(A_{lm_1}) = 1)$, $1 \le l \le L$, $1 \le m_i \le M$ and $m_i \ne m_j$ for $i \ne j$ and $i$, $j \in [1, ..., 4]$.*

The proof is given in the appendix to this chapter.

Using the results given by this lemma, we now show that $Z_L$ is asymptotically normal with mean zero and standard deviation 1.

**Lemma 2.2** *For $M \sim O(\frac{N}{lnL})$ and $N$, $L$ large such that $\lambda$ goes to a constant when $N$ and $L \to \infty$, $Z_L \sim N(0,1)$, where $\lambda = \frac{(L-1)}{2}[Q(-(1-k)\sqrt{\frac{N'}{M}}) - Q(-(1+k)\sqrt{\frac{N'}{M}})]$. Equivalently,*

$$\mathbf{E}(e^{sZ_L}) = e^{s^2}(1+\eta), \tag{2.45}$$

*where $\eta \sim O(\frac{1}{\sqrt{L}}) \to 0$ when $N$, $L \to \infty$.*

Proof:

First, as shown in the proof of Lemma 2.1, the $Z_{Ml}$'s, $1 \le l \le L$, are a mutually independent and identically distributed triangular array of exchangeable random variables. The $Y_l^{(m)}$'s, for $1 \le m \le M$, however, are dependent. To show that the $Z_L$ is asymptotically normal, it suffices to show that $\mathbf{E}(Z_{Ml}^4)$ are bounded when $L$, $N \to +\infty$ and $M \sim O(\frac{N}{lnN})$. Without loss of generality, we let $l = 1$ and then obtain

$$\mathbf{E}(Z_{M1}^4) = A_{ML} + (4T_1 - 6T_2)M + T_2M^2 + O(\frac{1}{M}), \tag{2.46}$$

where

$$A_{ML} = 4\mathbf{E}([Y_1^{(1)}]^3 Y_1^{(2)}) + 6\mathbf{E}([Y_1^{(1)}]^2 [Y_1^{(2)}]^2) - 12T_1 + 11T_2, \tag{2.47}$$

$$T_1 = \mathbf{E}({Y_1^{(1)}}^2 Y_1^{(2)} Y_1^{(3)}), \tag{2.48}$$

$$T_2 = \mathbf{E}(Y_1^{(1)} Y_1^{(2)} Y_1^{(3)} Y_1^{(4)}). \tag{2.49}$$

Using the results in equation (2.42), equation (2.43) and equation (2.44) in Lemma 2.1, we can obtain

$$T_1 \;=\; \frac{1}{\sigma^4}[(1-2P_{e1})P(1,1,1) + (5P_{e1}^2 - 2P_{e1})P(1,1) + (P_{e1}^3 - 3P_{e1}^4)]$$

$$= O(\frac{1}{M^2}), \tag{2.50}$$

$$\begin{aligned} T_2 &= \frac{1}{\sigma^4}[P(1,1,1,1) - 4P(1,1,1)P_{e1} + 6P(1,1)P_{e1}^2 - 3P_{e1}^4] \\ &= O(\frac{1}{M^2}). \end{aligned} \tag{2.51}$$

Then for $L$ and $N \to \infty$, and $M \sim O(\frac{N}{lnN})$, $\mathbf{E}({Z_{M1}}^4) \sim O(1)$. The Gaussian hypothesis is satisfied. That is, $\mathbf{E}(e^{sZ_L}) = e^{s^2}(1+\eta)$, where $\eta \sim O(\frac{1}{\sqrt{L}}) \to 0$ when $N$, $L \to \infty$.

Q.E.D.

The Gaussian approximation obtained here leads to our main theorem.

**Theorem 2.4** *Let* $\hat{\epsilon} = \frac{1}{\sigma}(\frac{1}{2} - \epsilon' - P_{e1})$. *For* $M = \frac{(1-k)^2N}{\pi(ln\frac{4\sqrt{\pi}\epsilon'^2}{9}L - \frac{1}{2}lnlnL)}$, $\hat{\epsilon} > 0$ *small and any* $\gamma \geq 0$ *arbitrarily small, we have* $\Pr(Z_L \geq \hat{\epsilon}\sqrt{LM}) \sim (1-\eta')$, *where* $\eta' \sim O(\frac{1}{(LM)^\alpha}) \to 0$ *for* $N, L \to \infty$, *and* $\alpha \geq 1$. *That is, a lower bound* $C^{-'}_{\frac{1}{2}-\epsilon'}$ *for the epsilon-capacity* $C^{-}_{\frac{1}{2}-\epsilon'}$ *(*$C^{-'}_{\frac{1}{2}-\epsilon'} \leq C^{-}_{\frac{1}{2}-\epsilon'}$*) can be obtained as*

$$\begin{aligned} C^{-}_{\frac{1}{2}-\epsilon'} &= \frac{(1-k)^2NL}{\pi(ln4\sqrt{\pi}\epsilon'^2L - \frac{1}{2}lnlnL)} \\ &\approx \frac{(1-k)^2N}{\pi ln4\sqrt{\pi}\epsilon'^2L}, \end{aligned} \tag{2.52}$$

*where* $\epsilon' > 0$ *small, and* $0 < k < 1$.

Proof:

The technique used here is motivated by that in [12].

By the Markov inequality, for any $v > 0$ and any non-negative random variable $Y$,

$$\Pr(Y \geq v) \leq \frac{\mathbf{E}(Y)}{v}. \tag{2.53}$$

It is easy to check from equations (2.31) (2.39) that when $M = \frac{(1-k)^2N}{\pi(ln4\sqrt{\pi}\epsilon'^2L - \frac{1}{2}lnlnL)}$, $\lambda \rightarrow \frac{1}{16\pi\epsilon'^2}$ for $L \rightarrow \infty$, and $\hat{\epsilon} \approx \frac{1}{\sigma}\epsilon' > 0.$, where $\sigma^2 = O(1)$ for $\lambda$ big So for our case, $Y = e^{Z_L}$ and $v = e^{\hat{\epsilon}\sqrt{LM}}$. Then using the result in Lemma 2.2, we can get

$$\begin{aligned}
\Pr(Z_L \geq \hat{\epsilon}\sqrt{LM}) &= \Pr(e^{Z_L} \geq e^{\hat{\epsilon}\sqrt{LM}}) \\
&\leq \inf_{r\geq 0} e^{-r\hat{\epsilon}\sqrt{LM}}\mathbf{E}(e^{rZ_L}) \\
&\leq (1+\eta)\inf_{r\geq 0} e^{-r\hat{\epsilon}\sqrt{LM}}e^{r^2} \\
&= (1+\eta)\inf_{r\geq 0} e^{(r-\frac{\hat{\epsilon}\sqrt{LM}}{2})^2 - \frac{\hat{\epsilon}^2LM}{4}} \\
&= (1+\eta)e^{-\frac{\hat{\epsilon}^2LM}{4}} \\
&= (1+\eta)e^{-\frac{\epsilon'^2LM}{4\sigma^2}} \\
&\rightarrow 0 \qquad (2.54)
\end{aligned}$$

when $L$ and $N \rightarrow \infty$.

Then $\Pr(Z_L < \hat{\epsilon}\sqrt{LM}) \sim 1 - \eta'$, where $\eta' \sim O(e^{-\frac{\epsilon'^2LM}{4\sigma^2}})$. Therefore, we obtain a lower bound $C^{-'}_{\frac{1}{2}-\epsilon'}$ for the lower epsilon-capacity $C^{-}_{\frac{1}{2}-\epsilon'}$:

$$\begin{aligned}
C^{-'}_{\frac{1}{2}-\epsilon'} &= \frac{(1-k)^2NL}{\pi(ln\frac{4\sqrt{\pi}\epsilon'^2}{9}L - \frac{1}{2}lnlnL)} \\
&\sim O(\frac{W}{lnL}). \qquad (2.55)
\end{aligned}$$

Q.E.D

To verify the theoretical result obtained in euqation (2.55), Monte Carlo simulations have been done to estimate the probability $\Pr(\hat{r} < \epsilon)$ as a function of $M$, where $\epsilon = \frac{1}{2} - \epsilon'$. In the simulation, we chose $N = 400$, $L = 500$, $k = .4$ and $\epsilon' = .1$. The resulting histogram plotted in Fig. 2.3 shows that the estimated $\Pr(\hat{r} < \epsilon) = 1$ for $M \leq 22$, that is, $LM < 11000$, where the points are the simulated probabilities averaged over 5 runs. Using the expression given in equation (2.55), we have a calculated lower bound $C^{-'}_{\frac{1}{2}-.1}$ for the lower epsilon-capacity, $C^{-'}_{\frac{1}{2}-.1} \approx 8600$. That is, when $M \approx 17.2$, $\Pr(\hat{r} < \epsilon)$ should be approximately 1. This agrees well with the simulations.

## 2.5 An Upper Bound

Since the total number of possible mappings of two layer $(N - 2L - 1)$ networks with binary weights and integer thresholds ranging in $[-N, N]$ is bounded by $2^{W+L\log 2N}$, the VC-dimension $d_b$ is upper bounded by $W + L\log 2N$, which is in the order of $W$. Then $d_b \leq O(W)$. By combining both the upper and lower bounds, we have

$$O(\frac{W}{lnL}) \leq d_b \leq O(W). \tag{2.56}$$

## 2.6 Discussion of the Methodology

How good is this general methodology for use in practice?

In the first place it is very general. That is, it can be applied for any feedforward network structure with one threshold unit at the output layer. Secondly, it is a constructive method- one can actually store random samples according to the construction of the network. How tight the resulting lower bound is usually depends on how good the construction of the network is and whether rigorous analysis can be carried out for the particular network considered. It should be noted that any analysis in evaluating the lower epsilon-capacity is usually quite complicated.

Specifically, to comment on the application of this general methodology in this paper, we mention independent work by Littlestone [9]. In his work he showed that the VC-dimension of so-called DNF expressions with $N$ variables and $K$ term was on the order of $KN$. Since any DNF expression can be implemented by a two layer

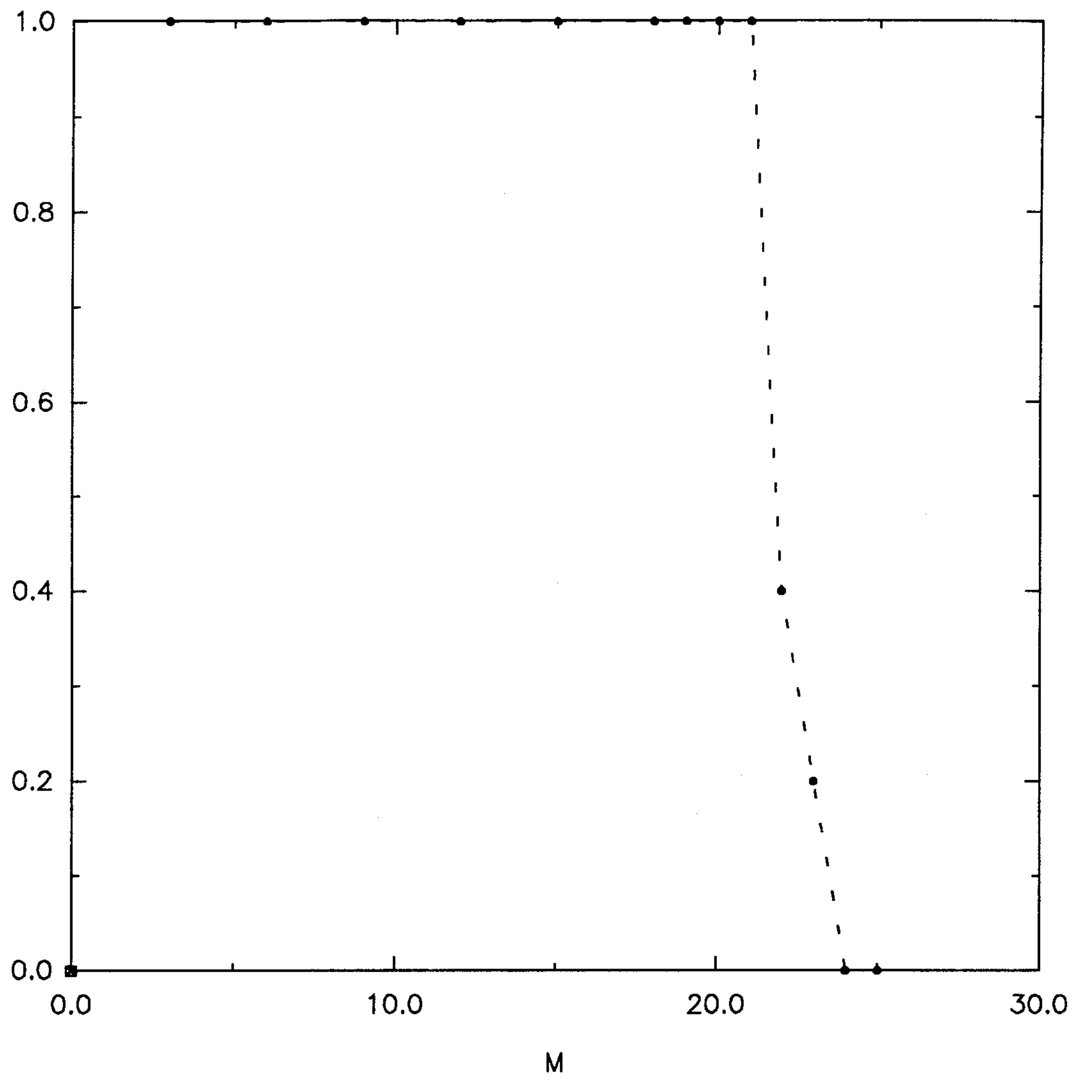


Figure 2.3: Monte Carlo simulation of $\Pr(\hat{r} \leq \epsilon)$. The vertical axis: $\Pr(\hat{r} \leq \epsilon)$

network of threshold units with binary weights and integer thresholds for hidden and the output units, which essentially implements "and" and "or" gates for the VC-dimension of the corresponding two-layer $(N-K-1)$ nets with binary weights and integer thresholds is also $O(KN)$, which is $O(W)$, $W$ being the number of weights of the networks. This result is tighter than ours by a factor of $lnL$. We believe that this $lnL$ factor is due to the limitations of the grandmother-cell type of networks used in our construction. However, since our analysis for the lower bound is constructive, it provides an explicit way to actually store random samples in the weights. On another occasion [7]where we have applied the constructive approach to a single neuron with discrete weights, we can show that this method is able to find a lower bound which is of the same order as the upper bound for the VC-dimension.

## 2.7 Conclusion

We have drawn a general relationship between the VC-dimension and the statistical lower epsilon-capacity. It provides a new view on the sample complexity for generalization. Specifically, its implications to learning and generalization can be summarized as follows.

1) For random assignments of the samples ($P_{be} = \frac{1}{2}$), since the statistical lower epsilon-capacity charaterizes the memorizing capability of networks and it is upper bounded by the universal sample bound for generalization, the relationship confirms that generalization occurs after memorization.

2) For cases where the Bayes error is smaller than $\frac{1}{2}$, the relationship indicates that an appropriate choice of a network structure is very important. If a network structure is properly chosen so that the optimal achievable error rate $P_{eo}$ is close to the Bayes error $P_{eb}$, than the optimal network in this class is the one which has

the largest lower epsilon-capacity. Since a suitable structure can hardly be chosen a priori due to a lack of knowledge about the underlying distribution, searching for network structures as well as weight values becomes necessary. Similar idea has been addressed by Devroye [3] in pattern recognition and by Vapnik [13] for structural minimization. It has also been used as guidelines in learning algorithms for finding a structure for multilayer networks during training[2] [6][14].

This relationship also brings a general constructive methodology of finding a lower bound for the VC-dimension of networks. We have applied this relation to obtain a lower bound for the VC-dimension of two layer networks with binary interconnections. We have shown that the VC-dimension $d_b$ of two layer networks with binary weights, integer thresholds for the hidden units and zero threshold for the output unit satisfies the relation $O(\frac{W}{lnL}) \leq d_b \leq O(W)$. This result demonstrates the capability of the networks in learning and generalization and gives strong theoretical support to the usage of multilayer networks with binary interconnections.

# Bibliography

[1] E. Baum and D. Haussler," What Size Net Gives Valid Generalization?" *Neural Computation*, 1(1), 151-160, 1989.

[2] J.S . Denker, Y. LeCun and S.A . Solla, "Optimal Brain Damage," *Advances in Neural Information Processing Systems*, Vol. 2, 598-605, 1989.

[3] L. Devroye, "Automatic Pattern Recognition: A Study of Probability of Error," *IEEE Trans. on Pattern Recognition and Machine Intelligence*, Vol. 10, No. 4, 530-543, July 1988.

[4] W. Feller, *An Introduction to Probability Theory and Its Applications*, New York: John Wiley and Sons, 1968.

[5] I.S . Gradshteyn and I.M . Ryzhik, *Table of Integrals, Series and Products*, Academic Press, 1980.

[6] C. Ji and D. Psaltis, "A Probabilistic Algorithm for Developing Network Structure," Submitted to *Neural Computation.*

[7] C. Ji and D. Psaltis, in preparation.

[8] M.G . Kendall, "Proof of Relations Connected with the Tetrachoric Series and Its Generalization," 196-198.

[9] N. Littlestone, " Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," *Machine Learning 2:* 285-318, 1988.

[10] R.J . McEliece, E.C . Posner, E.R . Rodemich, S.S . Venkatesh, "The Capacity of the Hopfield Associative Memory," *IEEE Trans. Inform. Theory*, Vol. IT-33, No. 4, 461-482, July 1987.

[11] J. Nadel, "Study of A Growth Algorithm for Neural Networks," *Int. J. Neural Syst.*, Vol. 1, 55-59, 1989.

[12] C.M . Newman, "Memory Capacity in Neural Network Models: Rigorous Lower Bounds," *Neural Networks*, Vol. 1, 223-238, 1988.

[13] V.N . Vapnik, *Estimation of Dependences Based on Empirical Data*, New York: Springer-Verlag, 1982.

[14] V.N . Vapnik and A.Ya. Chervonenkis, "On Uniform Convergence of Relative Frequency of Events to Their Probabilities," *Theor. Probability Appl.*, 16(1971), 264-280.

[15] S.S . Venkatesh and D. Psaltis, "Epsilon Capacity of Neural Networks," in press in *IEEE Trans. on Pattern Recognition and Machine Intelligence.*

## Appendix

Proof of Lemma 2.1:

We first state the fact that $I(A_{lm})$'s are identically distributed exchangeable random variables for a given $l \in [1, ..., L]$ and all $m \in [1, ..., M]$. Then without loss of generality, we can take $l = 1$ and $m_i = i$ for $i = 1, 2, 3, 4$. Then for given $\alpha_1 = 1$ (for $\alpha_1 = -1$, the result will be the same), we have

$$P(1,1) \;=\; 2\Pr(s_1^{(1)} = 2, s_1^{(2)} = 0 \mid \alpha_1 = 1)f_1 + \Pr(s_1^{(1)} = 2, s_1^{(2)} = 2 \mid \alpha_1 = 1)f_2$$

$$+ \quad \Pr(s_1^{(1)} = 0, s_1^{(2)} = 0 \mid \alpha_1 = 1) f_3, \tag{2.57}$$

where

$$\begin{aligned} f_1 &= \Pr(n_-^{(1)} - n_+^{(1)} \geq 2) \Pr(n_-^{(2)} - n_+^{(2)} \geq 1) \\ &+ \frac{1}{2} \Pr(n_-^{(1)} - n_+^{(1)} = 1) \Pr(n_-^{(2)} - n_+^{(2)} \geq 1) \\ &+ \frac{1}{2} \Pr(n_-^{(1)} - n_+^{(1)} \geq 2) \Pr(n_-^{(2)} - n_+^{(2)} = 0) \\ &+ \frac{1}{4} \Pr(n_-^{(1)} - n_+^{(1)} = 1) \Pr(n_-^{(2)} - n_+^{(2)} = 0), \end{aligned} \tag{2.58}$$

$$\begin{aligned} f_2 &= \Pr(n_-^{(1)} - n_+^{(1)} \geq 2) \Pr(n_-^{(2)} - n_+^{(2)} \geq 2) \\ &+ 2 \times \frac{1}{2} \Pr(n_-^{(1)} - n_+^{(1)} = 1) \Pr(n_-^{(2)} - n_+^{(2)} \geq 2) \\ &+ \frac{1}{4} \Pr(n_-^{(1)} - n_+^{(1)} = 1) \Pr(n_-^{(2)} - n_+^{(2)} = 1), \end{aligned} \tag{2.59}$$

$$\begin{aligned} f_3 &= \Pr(n_-^{(1)} - n_+^{(1)} \geq 1) \Pr(n_-^{(2)} - n_+^{(2)} \geq 1) \\ &+ 2 \times \frac{1}{2} \Pr(n_-^{(1)} - n_+^{(1)} = 0) \Pr(n_-^{(2)} - n_+^{(2)} \geq 1) \\ &+ \frac{1}{4} \Pr(n_-^{(1)} - n_+^{(1)} = 0) \Pr(n_-^{(2)} - n_+^{(2)} = 0). \end{aligned} \tag{2.60}$$

Here the same notations are used as in Section 2.4. To find the joint distribution of $s_1^{(1)}$ and $s_1^{(2)}$, it suffices to find the joint distribution of $z_1^{(1)}$ and $z_1^{(2)}$. Now

$$\begin{pmatrix} z_1^{(1)} \\ z_1^{(2)} \end{pmatrix} = \sum_{i=1}^{N} w_{1i} \begin{pmatrix} x_{1i}^{(1)} \\ x_{1i}^{(2)} \end{pmatrix}.$$

Since all the terms in the summation are mutually independent, we see by the multivariant central limit theorem, that for $N$ large

$$\left(\frac{1}{\sqrt{N}} z_1^{(1)}, \frac{1}{\sqrt{N}} z_1^{(2)}\right) \sim N(\mu, \mu, 1, 1, \rho), \tag{2.61}$$

where $N(\mu, \mu, 1, 1, \rho)$ is the joint distribution of two normal random variables with the same mean $\mu$, unit variance and correlation coefficient $\rho$, where

$$\begin{aligned}\mu &= \sqrt{\frac{2N}{\pi M}}, \\ \rho &= -\frac{2}{\pi M}.\end{aligned}$$

$N(0,0,1,1,\rho)$, however, can be expanded as [6]:

$$N(0,0,1,1,\rho) = f(x)f(y) + \sum_{r=1}^{+\infty} \frac{\rho^r}{r!}\tau_r(x)\tau_r(y), \tag{2.62}$$

where $\tau_r(x) = (-\frac{d}{dx})^r f(x)$ and $f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$.

Using such an expansion to evaluate $P(1,1)$, we can obtain

$$P(1,1) = P_{e1}^2 + 2\beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3, \tag{2.63}$$

since

$$\begin{aligned}P_{e1}^2 &= \Pr{}^2(s_1^{(1)} = 2 \mid \alpha_1 = 1)f_2 + \Pr{}^2(s_1^{(1)} = 0 \mid \alpha_1 = 1)f_3 \\ &+ 2\Pr(s_1^{(1)} = 2 \mid \alpha_1 = 1)\Pr(s_1^{(1)} = 0 \mid \alpha_1 = 1)f_1.\end{aligned} \tag{2.64}$$

The term $\beta_1$, $\beta_2$ and $\beta_3$ are evaluated as follows. Let

$$\begin{aligned}I_1 &= \int_{-k\sqrt{\frac{2N}{\pi M}}+\frac{\delta_2}{\sqrt{N}}}^{k\sqrt{\frac{2N}{\pi M}}+\frac{\delta_1}{\sqrt{N}}} \frac{df(x)}{dx} \\ &\approx -k\sqrt{\frac{2}{\pi}}\frac{(\delta_1+\delta_2)}{\sqrt{M}}e^{-\frac{k^2 N}{\pi M}},\end{aligned} \tag{2.65}$$

$$\begin{aligned}I_2 &= \int_{-\infty}^{-k\sqrt{\frac{2N}{\pi M}}+\frac{\delta_1}{\sqrt{N}}} \frac{df(x)}{dx} + \int_{k\sqrt{\frac{2N}{\pi M}}+\frac{\delta_2}{\sqrt{N}}}^{+\infty} \frac{df(x)}{dx} \\ &\approx k\sqrt{\frac{2}{\pi}}\frac{(\delta_1+\delta_2)}{\sqrt{M}}e^{-\frac{k^2 N}{\pi M}},\end{aligned} \tag{2.66}$$

where $\delta_1$ and $\delta_2$ are two small constants in $(-1,1)$ such that $(1-k)\sqrt{\frac{2}{\pi}}\frac{N}{\sqrt{M}} + \frac{\delta_1}{\sqrt{N}}$ and $(1+k)\sqrt{\frac{2}{\pi}}\frac{N}{\sqrt{M}} + \frac{\delta_1}{\sqrt{N}}$ equal to integers respectively. Then

$$\begin{aligned}\beta_1 &= \rho(I_1 I_2) + O(\rho^2)\\ &= O(\rho^2). \end{aligned} \tag{2.67}$$

Similarly, we can obtain that $\beta_2 = O(\rho^2)$ and $\beta_3 = O(\rho^2)$.

Then

$$P(1,1) = P_{e1}^2 + O(\frac{1}{M^2}). \tag{2.68}$$

Using similar derivations and the expansions

$$\begin{aligned} &N(0,0,0,1,1,1,\rho) = f(x_1)f(x_2)f(x_3)\\ &+ \sum_{\substack{m \\ m+n}}^{+\infty} \sum_{\substack{n \\ +o=}}^{+\infty} \sum_{\substack{o \\ r, r=1}}^{+\infty} \frac{\rho^r}{m!n!o!} \tau_{r-o}(x_1)\tau_{r-m}(x_2)\tau_{r-n}(x_3), \end{aligned} \tag{2.69}$$

$$\begin{aligned} &N(0,0,0,0,1,1,1,1,\rho) = f(x_1)f(x_2)f(x_3)f(x_4)\\ &+ \sum_{\substack{k \\ k+l}}^{+\infty} \sum_{\substack{l \\ +m}}^{+\infty} \sum_{\substack{m \\ +n}}^{+\infty} \sum_{\substack{n \\ +o}}^{+\infty} \sum_{\substack{o \\ +p}}^{+\infty} \sum_{\substack{p \\ =r, r=1}}^{+\infty} \frac{\rho^r}{k!l!m!n!o!p!} \tau_{r-n-o-p}(x_1)\tau_{r-l-m-p}(x_2)\tau_{r-k-m-o}(x_3)\tau_{r-k-l-n}(x_4) \end{aligned} \tag{2.70}$$

we can obtain

$$P(1,1,1) = P_{e1}^3 + O(\frac{1}{M^2}), \tag{2.71}$$

$$P(1,1,1,1) = P_{e1}^4 + O(\frac{1}{M^2}). \tag{2.72}$$

where the same notations are used as before for $f(x)$ and $\tau_n(x)$'s.

Q.E.D.

# Chapter 3

# Is The VC Sample Bound Tight? - A Case Study

## 3.1 Introduction

Since the universal sample bound for generalization is a distribution and network independent quantity, it is essentially the worst case bound. Therefore, it is natural to ask the question: whether this bound is tight for distribution dependent cases? That is, for what kind of distributions and networks, the sample complexity for generalization is in the order of the VC-dimension; and when the sample complexity can be smaller? Although these questions have not been answered yet in general, two examples given in this chapter, which consider a particular distribution and a particular network, demonstrate that the sample complexity needed for generalization indeed changes in terms of distributions. Specifically, we show that the sample complexity for generalization is in the order of the VC-dimension in one case, and smaller than the VC-dimension by a log factor in another.

## 3.2 Distribution of the Samples: Binary Clusters

The distribution of the samples we consider can be specified as follows.

Let $\vec{\mu}_1$ and $\vec{\mu}_2$ be two N-dimensional binary vectors, where $\vec{\mu}_1^T = [1, ..., 1]$ and $\vec{\mu}_2 = -\vec{\mu}_1$. Let $\{\vec{X}_1^{(m)}\}$ and $\{\vec{X}_2^{(m)}\}$ be two sets of N-dimensional random vectors, for $m \in [1, M]$. All elements $x_{ji}^{(m)}$ of the random binary vectors are mutually independent, where

$$x_{ji}^{(m)} = \begin{cases} (-1)^j & \text{with probability } p, \\ -(-1)^j & \text{with probability } (1-p), \end{cases}$$

for $0 < p \leq \frac{1}{2}$, $i \in [1, N]$ and $j = 1, 2$. Then the two sets of random vectors form two binary clusters centered at $\vec{\mu}_1$ and $\vec{\mu}_2$. The closer the $p$ to $\frac{1}{2}$, the smaller the separation between the two clusters. When $p = \frac{1}{2}$, two clusters completely overlap with one another.

## 3.3 Construction of the Network

The network considered is a linear network (a single neuron) whose weights are

$$w_i = \frac{1}{2M} \sum_{m=1}^{M} (x_{1i}^{(m)} - x_{2i}^{(m)}), \tag{3.1}$$

where $i \in [1, N]$.

For $0 < p < \frac{1}{2}$, it is easy to check that this network is asymptotically optimal since $w_i \rightarrow w_{oi}$ when $M \rightarrow +\infty$ for all $i \in [1, N]$, where $w_{oi}$'s are the weights of the Baysian classifier

$$\begin{aligned} w_{oi} &= \mu_{1i} - \mu_{2i} \\ &= 2. \end{aligned} \tag{3.2}$$

## 3.4 Definition of Generalization

Let $Z(M) = \sum_{m=1}^{2M} [I(A_{1m}) + I(A_{2m})]$, where $I(A_{jm})$ is the indicator function and $A_{jm}$ is the event that the $m$-th stored sample associated with the $j$-th cluster is classified incorrectly by the network for $m \in [1, M]$ and $j = 1, 2$. Then $\hat{r} = \frac{Z(M)}{2M}$ represents the training error rate. According to our classification rule, the $m$-th stored sample from the $j$-th cluster is classified incorrectly if the presynaptic input $y_j^{(m)}$ of the neuron has the opposite sign as its label for $y_j^{(m)} \neq 0$; and $y_j^{(m)}$ is assigned to be either $+1$ or $-1$ with probability $\frac{1}{2}$ if $y_j^{(m)} = 0$.

For the convenience of analysis, generalization defined here is slightly different from that in the previous chapter. Instead of using the true probability of error of the network, the Bayes error, $P_{be}$, which is the limit of the true probability of error of the network when $M \rightarrow \infty$, is used for comparison with the training error. That is, the probability $\Pr(|\ \hat{r} - P_{be}\ | \geq \epsilon)$ is used to evaluate generalization.

**Definition 3.1** *For a given $\epsilon > 0$ small, generalization happens with probability $1 - \eta$ if the probability $\Pr(|\ \hat{r} - P_{be}\ | \geq \epsilon) = \eta$. When $\eta$ is very small, generalization occurs almost surely. Then the sample complexity for generalization can be identified as a number $M_0$, which is an increasing function of $N$, such that for the given $\epsilon > 0$, $\Pr(|\ \hat{r} - P_{be}\ | \geq \epsilon) \leq \eta$ when $M \geq M_0$, where $\eta \rightarrow 0$ when $N \rightarrow \infty$.*

## 3.5 Evaluation of the Sample Complexity for Generalization

To investigate how the sample complexity for generalization changes with respect to the distribution, we vary the parameter $p$, which represents the amount of separation between the two clusters, and study the following two cases.

### 3.5.1 Case 1

For this case, $p = \frac{1}{2} - \sqrt{\frac{2ln\frac{1}{\epsilon}}{N}}$.

Utilizing a similar derivation to that given in Chapter 2, the probability $\Pr(|\hat{r} - P_{be}| \geq \epsilon)$ can be expressed as

$$\Pr(|\hat{r} - P_{be}| \geq \epsilon) = \Pr(\hat{r} \leq P_{be} - \epsilon) + \Pr(\hat{r} \geq P_{be} + \epsilon). \tag{3.3}$$

For $N$ large, it is easy to show that $P_{be} = \epsilon$ when $p = \frac{1}{2} - \sqrt{\frac{2ln\frac{1}{\epsilon}}{N}}$, using the large deviation theorem [4]. Then

$$\Pr(|\hat{r} - P_{be}| \geq \epsilon) = \Pr(\hat{r} = 0) + \Pr(\hat{r} \geq 2P_{be}), \tag{3.4}$$

where $\Pr(\hat{r} = 0)$ is the probability that all samples are classified correctly.

To find the sample complexity for generalization, two steps need to be taken. First, the capacity must be evaluated from the probability $\Pr(\hat{r} = 0)$. Then the probability $\Pr(\hat{r} \geq 2P_{be})$ must be shown to be always negligible. Therefore, the capacity obtained will then equal the sample complexity for generalization for this network.

#### Evaluation of The Capacity

**Definition 3.2** *The epsilon-capacity for the network considered is defined to be the sharp transition point $C$ of the probability $\Pr(\hat{r} \leq \epsilon')$ such that for $M \leq (1-\gamma)C$, $\Pr(\hat{r} \leq \epsilon') > 1 - \eta$; and $\Pr(\hat{r} \leq \epsilon') < \eta$ for $M \geq (1+\gamma)C$, where $\epsilon'$ is the given small positive number, and $\eta \longrightarrow 0$ when $N \longrightarrow \infty$. Here we assume this sharp transition point exists and is unique, which is usually the case for a single neuron.*

To evaluate the capacity from the probability $\Pr(\hat{r} = 0)$ (a special case for $\epsilon' = 0$), the probability of error of one sample and the joint probability of error of two samples are first calculated and then used to bound the probability $\Pr(\hat{r} = 0)$.

**a) Probability of Error for a Single Sample**

Let $P_{e1}$ be the probability of incorrect classification of one sample stored. Let $y_j^{(m)}$ be the presynaptic input of the neuron when a stored sample $\vec{X}_j^{(m)}$ is fed through, where $j \in [1,2]$ and $m \in [1,M]$. Because $y_j^{(m)}$'s are exchangeable random variables, without loss of generality, we take $j=1$ and $m=1$. Then

$$\begin{aligned} y_1^{(1)} &= \sum_{i=1}^{N} w_i x_{1i}^{(1)} \\ &= \frac{N}{2M} + \sum_{i=1}^{N} \frac{1}{2M} \left( \sum_{m=2}^{M} x_{1i}^{(m)} - \sum_{m=1}^{M} x_{2i}^{(m)} \right) x_{1i}^{(1)}. \end{aligned} \tag{3.5}$$

Since all the terms in the above summation over $i$ are mutually independent, by the large-deviation theorem [4], for $N$ large and $\mid \frac{N}{2M} - N(1-2p)^2 \mid < O(N^{\frac{1}{3}})$

$$\begin{aligned} P_{e1} &= \Pr(y_1^{(1)} \leq 0) \\ &\approx Q(-\hat{\mu}), \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} \hat{\mu} &= \frac{\hat{\mu}_1}{\sigma_1} \\ &= \frac{1}{\sqrt{2}} \sqrt{\frac{N}{2M} + 4ln\frac{1}{P_{be}}}, \end{aligned} \tag{3.7}$$

$$\begin{aligned} \hat{\mu}_1 &= E(y_1^{(1)}) \\ &= \frac{N}{2M} + 4ln\frac{1}{P_{be}} + o(1), \end{aligned} \tag{3.8}$$

$$\begin{aligned} \sigma_1^2 &= E(y_1^{(1)^2}) - E^2(y_1^{(1)}) \\ &= \frac{N}{M} + 8ln\frac{1}{P_{be}} + o(1), \end{aligned} \tag{3.9}$$

and $Q(-x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-x} e^{\frac{-u^2}{2}} du$.

It is easy to check that $P_{e1} \leq P_{be}$ for all $M$, and $P_{e1} \rightarrow P_{be}$ when $M \rightarrow +\infty$ due to the asymptotic optimality of the network.

### b) Joint Probability of Error for Two Samples

Let $P(1,1)$ be the joint probability of incorrect classifications of any two stored samples $\vec{X}_{j_1}^{(m_1)}$ and $\vec{X}_{j_2}^{(m_2)}$ for $j_1$, $j_2 \in [1,2]$ and $m_1$, $m_2 \in [1,M]$ . Again, we can pick $j_1 = 1$, $j_2 = 1$, $m_1 = 1$ and $m_2 = 2$ without loss of generality. Then we have

$$\begin{pmatrix} y_1^{(1)} \\ y_1^{(2)} \end{pmatrix} = \sum_{i=1}^{N} w_i \begin{pmatrix} x_{1i}^{(1)} \\ x_{1i}^{(2)} \end{pmatrix}.$$

Applying the multivariate central limit theorem to the mutually independent two-dimensional random vectors in the summation, we have

$$\begin{aligned} P(1,1) &= \Pr(I(A_{11}) = 1, I(A_{12}) = 1) \\ &\approx \int_{-\infty}^{-\hat{\mu}} \int_{-\infty}^{-\hat{\mu}} f(x,y)dxdy, \end{aligned} \tag{3.10}$$

where $\hat{\mu}$ is given in equation (3.8), $f(x,y)$ is the joint normal distribution $N(0,0,1,1,\rho)$, and $\rho$ is the correlation coefficient between $y_1^{(1)}$ and $y_1^{(2)}$ which satisfies

$$\rho = \frac{\frac{1}{2M} + \frac{4}{N} ln \frac{1}{P_{be}}}{1 + \frac{6M}{N} ln \frac{1}{\epsilon}}. \tag{3.11}$$

For $M \ll \frac{N}{2ln\frac{1}{P_{be}}}$,

$$\rho \approx \frac{1}{2M} + \frac{4ln\frac{1}{P_{be}}}{N}. \tag{3.12}$$

Using the expansion given in [6], we can obtain

$$P(1,1) = Q^2(-\hat{\mu}) + \frac{1}{4\pi M} e^{-\hat{\mu}^2} + o(\frac{1}{M}). \tag{3.13}$$

### c) The Capacity

Although the probability $\Pr(\hat{r} = 0)$ can be shown to have an approximately Poisson distribution when $M \sim O(\frac{N}{lnN})$ using a similar technique to that in Chapter 3, a simpler approach is used here to estimate $\Pr(\hat{r} = 0)$ and to find the capacity. Specifically, a theorem in [3] is given below and used to bound this probability.

**Theorem 3.1** *Let $\vec{X}$ be a k-dimensional random variable and $D_1$, ..., $D_k$ be Borel-measurable subsets of the real line. Then*

$$1 - Q_1 \le \Pr[\cap_{i=1}^{k}\{X_i \in D_i\}] \le 1 - \frac{Q^2{}_1}{Q_1 + 2Q_2}, \tag{3.14}$$

*where* $Q_1 = \sum_{i=1}^{k} \Pr(X_i \notin D_i)$ *and* $Q_2 = \sum_{i=2}^{k}\sum_{j=1}^{i-1} \Pr(X_i \notin D_i, X_j \notin D_j)$.

Applying this theorem to our case, we have

$$Q_1 = 2MP_{e1},$$

$$Q_2 = M(2M-1)P(1,1). \tag{3.15}$$

Then

$$1 - 2MP_{e1} \le \Pr(\hat{r} = 0) \le 1 - \frac{4M^2P_{e1}^2}{2MP_{e1} + 2M(2M-1)P(1,1)}. \tag{3.16}$$

By setting the lower and upper bounds in this inequality to be $1-\eta$ and $\eta$ respectively with $\eta > 0$ arbitrarily small, the capacity can be obtained through Theorem 3.2.

**Theorem 3.2** *For large N, the capacity of the network satisfies*

$$C \approx \frac{N}{4lnP_{be}N}. \tag{3.17}$$

Proof:

a). A lower bound for the capacity $C$.

Let $2M = \frac{N}{alnP_{be}N}$. Then

$$\hat{\mu} = \frac{1}{\sqrt{2}}\sqrt{alnP_{be}N + 4ln\frac{1}{P_{be}}}, \tag{3.18}$$

$$\begin{aligned} 2MP_{e1} &\sim \frac{2M}{\sqrt{2\pi}\hat{\mu}} e^{-\frac{\hat{\mu}^2}{2}} \\ &\sim O(N^{1-\frac{a}{4}}), \end{aligned} \tag{3.19}$$

which goes to 0 when $a > 4$ and $N \to +\infty$. That is, when $2M < \frac{N}{4lnP_{be}N}$, $1-2MP_{e1} \approx 1$. By the definition of the capacity $C$, we have

$$(1-\gamma)\frac{N}{4lnP_{be}N} < C, \tag{3.20}$$

where $\gamma > 0$ arbitrarily small.

b). An upper bound for the capacity $C$.

Using the expansion given by equation (4.66), the upper bound in the inequality (3.16) satisfies

$$1 - \frac{4M^2P_{e1}^2}{2MP_{e1} + 2M(2M-1)P(1,1)} = 1 - \frac{4M^2P_{e1}^2}{4M^2P_{e1}^2 + 2M(P_{e1} - {P_{e1}}^2 + \frac{1}{4\pi}e^{-\hat{\mu}^2}) + o(1)}. \tag{3.21}$$

Let $2M = \frac{N}{blnP_{be}N}$. The similar derivations show that $1 - \frac{4M^2P_{e1}^2}{2MP_{e1}+2M(2M-1)P(1,1)} \sim \eta$ when $b \leq \frac{4}{(1+\gamma)}$, where $\eta \to 0$ for $N \to +\infty$ and $\gamma > 0$ arbitrarily small. By the definition of the capacity $C$, we have

$$C < (1+\gamma)\frac{N}{4lnP_{be}N}. \tag{3.22}$$

Combining equation (3.20) and equation (3.22) and utilizing the assumption for the uniqueness of the sharp transition point, we have $C = \frac{N}{4lnP_{be}N}$.

### 3.5.2 The Sample Complexity for Generalization

As aforementioned, to show that the sample complexity for generalization equals the capacity $\frac{N}{4lnP_{be}N}$, we now only need to show that $\Pr(\hat{r} \geq 2P_{be}) < \eta$, where $\eta \to 0$ when $N \to \infty$. We will show below that this is indeed true if we assume again that $\Pr(\hat{r} \leq \epsilon')$ has at most one sharp transition point for a given $\epsilon' \geq 0$.

**Theorem 3.3** $0 \leq \Pr(\hat{r} \geq 2P_{be}) \leq \eta$, *where* $\eta \to 0$ *when* $N \to +\infty$ *and* $M$ *is an increasing function of* $N$, *assuming that* $\Pr(\hat{r} < \epsilon')$ *has at most one sharp transition point for a given* $0 \leq \epsilon' \leq 1$.

Proof:

Assume that $Pr(\hat{r} \leq 2P_{be})$ does have one sharp transition occurring at $M_0$, where $M_0$ is large. Then for $M = M_0$ fixed,

$$\begin{aligned} P_{e1} &= E\hat{r} \\ &= \int_0^1 x f(x) dr \\ &= \int_0^1 x d \Pr(\hat{r} \leq x) && (3.23) \\ &= \int_0^1 [1 - \Pr(\hat{r} \leq x)] dx && (3.24) \\ &= \int_0^{2P_{be}-\Delta} [1 - \Pr(\hat{r} \leq x)] dx + \int_{2P_{be}-\Delta}^{2P_{be}+\Delta} [1 - \Pr(\hat{r} \leq x)] dx \\ &+ \int_{2P_{be}+\Delta}^{1} [1 - \Pr(\hat{r} \leq x)] dx, && (3.25) \end{aligned}$$

where $f(\hat{r})$ is the density of $\hat{r}$, and $\Delta$ is a small positive quantity. Equation (3.24) is obtained through integration by parts in equation (3.23). Then the three integrals in equation (3.25) can be evaluated one by one using the sharp transition property of the capacity point.

Let $M_0 = M_0(\epsilon')$ where $\epsilon' = 2P_{be}$, which indicates that the epsilon-capacity $M_0$ is a function of the variable $\epsilon'$. We first assert that $M_0(\epsilon')$ is a non-decreasing function of $\epsilon'$. This can be seen easily since $\Pr(\hat{r} \leq 2P_{be}) \leq \Pr(\hat{r} \leq 2P_{be} + \Delta)$ and $\Pr(\hat{r} \leq 2P_{be}) \geq 1 - \eta$, $\Pr(\hat{r} \leq 2P_{be} + \Delta) \geq 1 - \eta$, for $M = M_0$, where $\eta \to 0$ when

$N \rightarrow \infty$. Then the third integral can be obtained

$$\int_{2P_{be}+\Delta}^{1} [1-\Pr(\hat{r} \leq x)]dx = (1-2P_{be}-\Delta)\eta' = \eta'', \tag{3.26}$$

where $\eta'$ and $\eta''$ are negligible when $N$ is large. The second integral is easy to find

$$\int_{2P_{be}-\Delta}^{2P_{be}+\Delta} [1-\Pr(\hat{r} \leq x)]dx = \Delta', \tag{3.27}$$

where $\Delta' < 2\Delta$. To evaluate the first integral, we assume $\frac{dM_0(\epsilon')}{d\epsilon'}$ exists and non-zero at $\epsilon' = 2P_{be}$; then $\frac{dM_0(\epsilon')}{d\epsilon'} > 0$ at $\epsilon' = 2P_{be}$ due to the non-decreasing property we have just shown. Taking the first two terms in the Taylor expansion of $M_0(2P_{be}-\Delta)$ around point $2P_{be}$, we can obtain that the sharp transition point for $\Pr(\hat{r} \leq 2P_{be}-\Delta)$ equals to $M_0(2P_{be}-\Delta) \approx M_0(2P_{be})(1-\gamma')$, where $\gamma' \approx \Delta\frac{dM_0(\epsilon')}{d\epsilon'}$ at $\epsilon' = 2P_{be}$, which is greater than zero according to the assumptions. Then $M_0(2P_{be}) \approx M_0(2P_{be}-\Delta)(1+\gamma')$. By the definition of the epsilon-capacity, $\Pr(\hat{r} \leq 2P_{be}-\Delta) \leq \eta$ for $M = M_0$. Therefore,

$$\int_{0}^{2P_{be}-\Delta} [1-\Pr(\hat{r} \leq x)]dx = (2P_{be}-\Delta)(1-\eta_1), \tag{3.28}$$

where $\eta_1$ is negligible when $N$ is large. Putting equations (3.26) (3.27) and (3.28) into equation (3.25), we have, for $\Delta$ sufficiently small,

$$P_{e1} \approx 1-2P_{be}. \tag{3.29}$$

Because it is easy to verify that $P_{e1} < P_{be}$ for any finite but big $M$, $P_{e1} < P_{be}$ and $P_{e1} \approx (1-2P_{be})$ need to be satisfied simultaneously. $P_{be}$, however, has been assumed to be small. We can then assume that $P_{be} < \frac{1}{4}$. Therefore, both $P_{e1} < \frac{1}{4}$ and $P_{e1} > \frac{1}{2}$ have to hold, which is impossible. Then there must exist no sharp transition point for $Pr(\hat{r} \leq 2P_{be})$ at all. Since $Pr(\hat{r} = 0) \leq Pr(\hat{r} \leq 2P_{be})$, and it has been shown that $Pr(\hat{r} = 0) \approx 1$ when $M \leq C$ and $Pr(\hat{r} = 0) \approx 0$ otherwise, we must have $Pr(\hat{r} \leq 2P_{be}) \approx 1$ for all $M$. That is, $Pr(\hat{r} \geq 2P_{be}) \approx 0$.

Q.E.D

Then the sample complexity for generalization for this network equals to its capacity, $\frac{N}{4lnP_{be}N}$. This quantity is smaller than the VC-dimension of a single threshold element, which is $N$, by a factor of $lnN$.

Monte Carlo simulations for $\Pr(|\hat{r} - P_{be}| \geq P_{be})$ are given in Fig. 3.1 to compare with the theoretical result. For $N = 400$ and $P_{be} = .1$, the estimated sample complexity, which is obtained through the simulated probability $\Pr(|\hat{r} - P_{be}| \geq P_{be})$ over 20 runs, is approximately equal to 27; the capacity estimated from equation (4.2) is about 33. Fig. 3.2 gives the Monte Carlo simulation over 20 runs for $\Pr(\hat{r} \geq 2P_{be})$ using the same parameters. It shows that $\Pr(\hat{r} \geq 2P_{be}) = 0$, which is consistent with the result given by Theorem 3.3.

### 3.5.3 Case 2

For this case, $p = \frac{1}{2}$. It is easily to see that $P_{be} = \frac{1}{2}$. That is, the two clusters completely overlap with one another.

Since any network with asymptotic error rate $\frac{1}{2}$ is now asymptotically optimal, a slightly different construction of the weights of a neuron is used to facilitate the analysis. Specifically, we have

$$w_i = \frac{1}{2M}\sum_{m=1}^{M}(\alpha_1^{(m)}x_{1i}^{(m)} + \alpha_2^{(m)}x_{2i}^{(m)}), \tag{3.30}$$

where $\alpha_j^{(m)}$'s are mutually independent binary random variables taking $+1$ and $-1$ with equal probability $\frac{1}{2}$, $m \in [1, M]$ and $i \in [1, ..., N]$. The $\alpha_j^{(m)}$'s are also mutually independent of the samples.

By the relation between the epsilon-capacity and the VC-dimension derived in Chapter 2, if the epsilon-capacity obtained from the probability $\Pr(\hat{r} \leq \frac{1}{2} - \epsilon')$ is

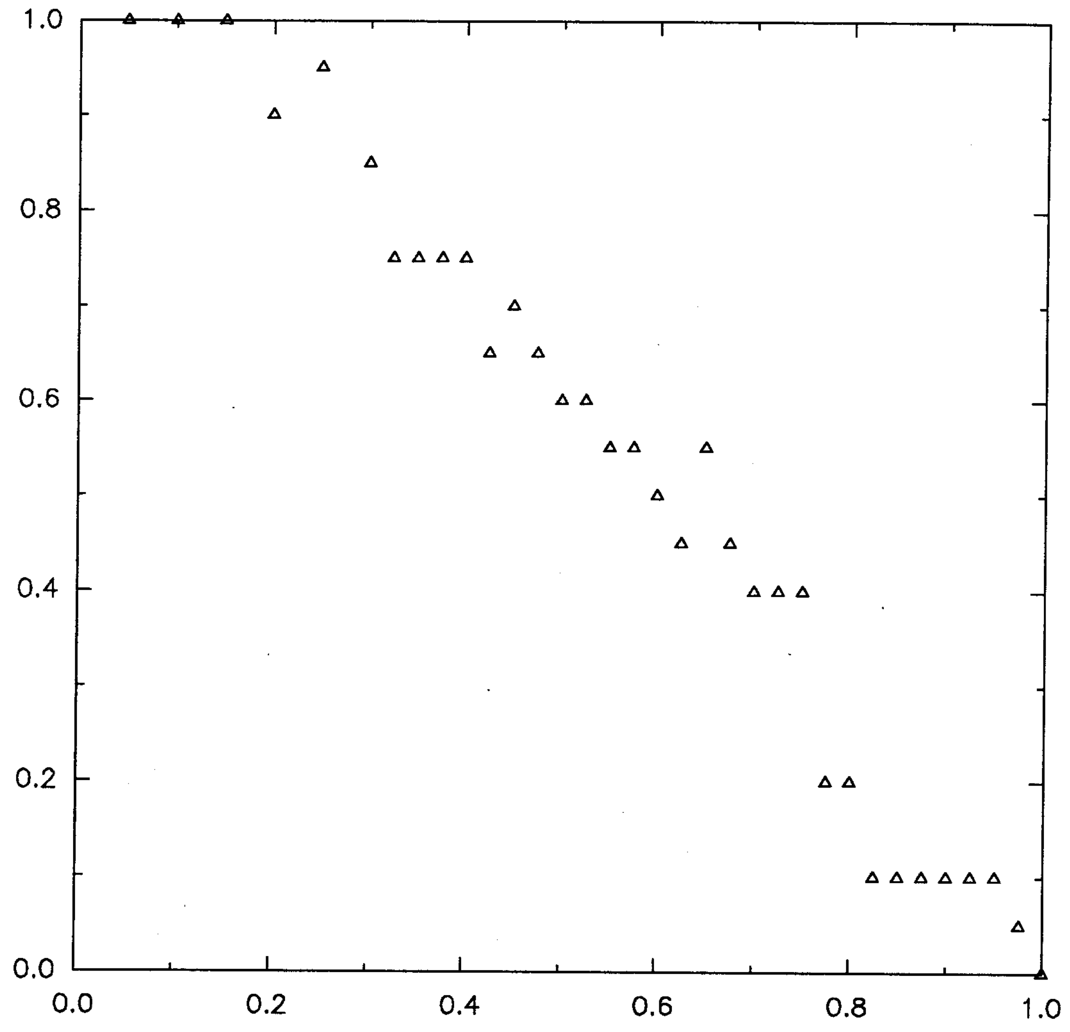


Figure 3.1: Monte Carlo simulation for the probability $\Pr(|\ \hat{r} - P_{be}\ | \geq P_{be})$. The horizontal axis: $\frac{2M}{N}$, the vertical axis: $\Pr(|\ \hat{r} - P_{be}\ | \geq P_{be})$.

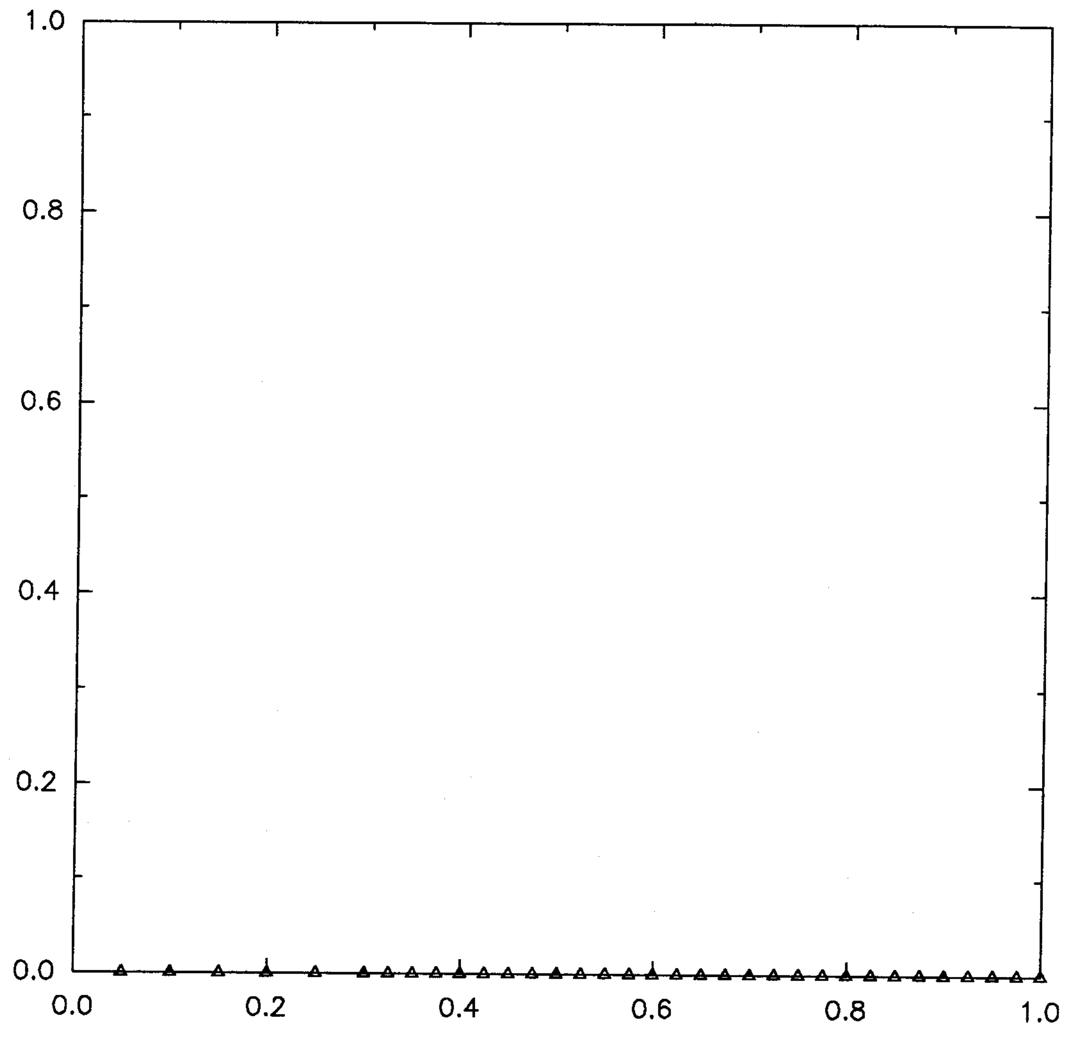


Figure 3.2: Monte Carlo simulation for the probability $\Pr(\hat{r} \geq 2P_{be})$. The horizontal axis: $\frac{2M}{N}$, the vertical axis: $\Pr(\hat{r} \geq 2P_{be})$.

in the order of the VC-dimension, which equals $N$ for a single neuron, the sample complexity for generalization is also in the order of the VC-dimension, where $\epsilon' > 0$ small. In what follows, we show that the epsilon-capacity is indeed $O(N)$.

## Evaluation of The Capacity

Using a similar approach as in case 1, the probability of incorrect classification of one sample stored can be easily obtained as

$$P_{e1} \approx Q(-\sqrt{\frac{N}{2M}}). \tag{3.31}$$

Due to the mutual independence of the random variables $I(A_{jm})$ for $j \in [1,2]$ and $m \in [1, M]$, we can obtain

$$\Pr(\hat{r} \leq \frac{1}{2} - \epsilon') = \sum_{m=0}^{(\frac{1}{2}-\epsilon')2M} \binom{2M}{m} P_{e1}^m (1 - P_{e1})^{(2M-m)}. \tag{3.32}$$

By the law of large numbers, when $M$ is large corresponding to $N$ large, $\mid (\frac{1}{2} - \epsilon' - P_{e1})2M \mid \leq O(\sqrt{P_{e1}(1 - P_{e1})2M})$, so that

$$\begin{aligned} \Pr(\hat{r} \leq \frac{1}{2} - \epsilon') &\approx Q[\frac{\sqrt{2M}(\frac{1}{2} - \epsilon' - P_{e1})}{\sqrt{P_{e1}(1 - P_{e1})}}] - Q[-\frac{\sqrt{2MP_{e1}}}{\sqrt{1 - P_{e1}}}] \\ &\approx Q[\frac{\sqrt{2M}(\frac{1}{2} - \epsilon' - P_{e1})}{\sqrt{P_{e1}(1 - P_{e1})}}]. \end{aligned} \tag{3.33}$$

To find an $M$ such that $Q[\frac{\sqrt{2M}(\frac{1}{2}-\epsilon'-P_{e1})}{\sqrt{P_{e1}(1-P_{e1})}}] \approx 1$, we let $Q(\frac{\sqrt{2M}(\frac{1}{2}-\epsilon'-P_{e1})}{\sqrt{P_{e1}(1-P_{e1})}}) = (1 - \eta)$ and $\frac{1}{2} - \epsilon' - P_{e1} = A(M)$, where $A(M) \geq o(\frac{1}{M^\alpha})$, where $0 \leq \alpha < \frac{1}{2}$ (we take $\alpha = \frac{1}{4}$ here), and $\eta \to 0$ when $N \to +\infty$. Expanding $Q(\frac{\sqrt{2M}(\frac{1}{2}-\epsilon'-P_{e1})}{\sqrt{P_{e1}(1-P_{e1})}})$ at the origin, we can obtain for $\epsilon'$ small

$$\begin{aligned} C_{\frac{1}{2}-\epsilon'} &\sim \frac{N}{2\pi(\epsilon' - A(M))^2} \\ &\sim \frac{N}{2\pi\epsilon'^2}. \end{aligned} \tag{3.34}$$

The Monte Carlo simulations for $\Pr(|\hat{r} - P_{be}| \geq \epsilon')$ over 20 runs is shown in Fig. 3.3, where $P_{be} = \frac{1}{2}$, $\epsilon' = .1$, and $N = 64$. The number of samples needed for generalization from the simulation is about 1019; the epsilon-capacity estimated from equation (3.34) is 1024.

Then the epsilon-capacity is $O(N)$, which is on the order of the VC-dimension $N$ for a single threshold element. By definition, the sample complexity for generalization for this network, which is no smaller than its epsilon-capacity, is therefore also on the order of the VC-dimension.

## 3.6 Conclusion

Case 1 and Case 2 demonstrate that the sample complexity of generalization for the particular network of linear threshold elements and the specific distribution are either on the order of the VC-dimension or smaller by a factor of $lnN$.

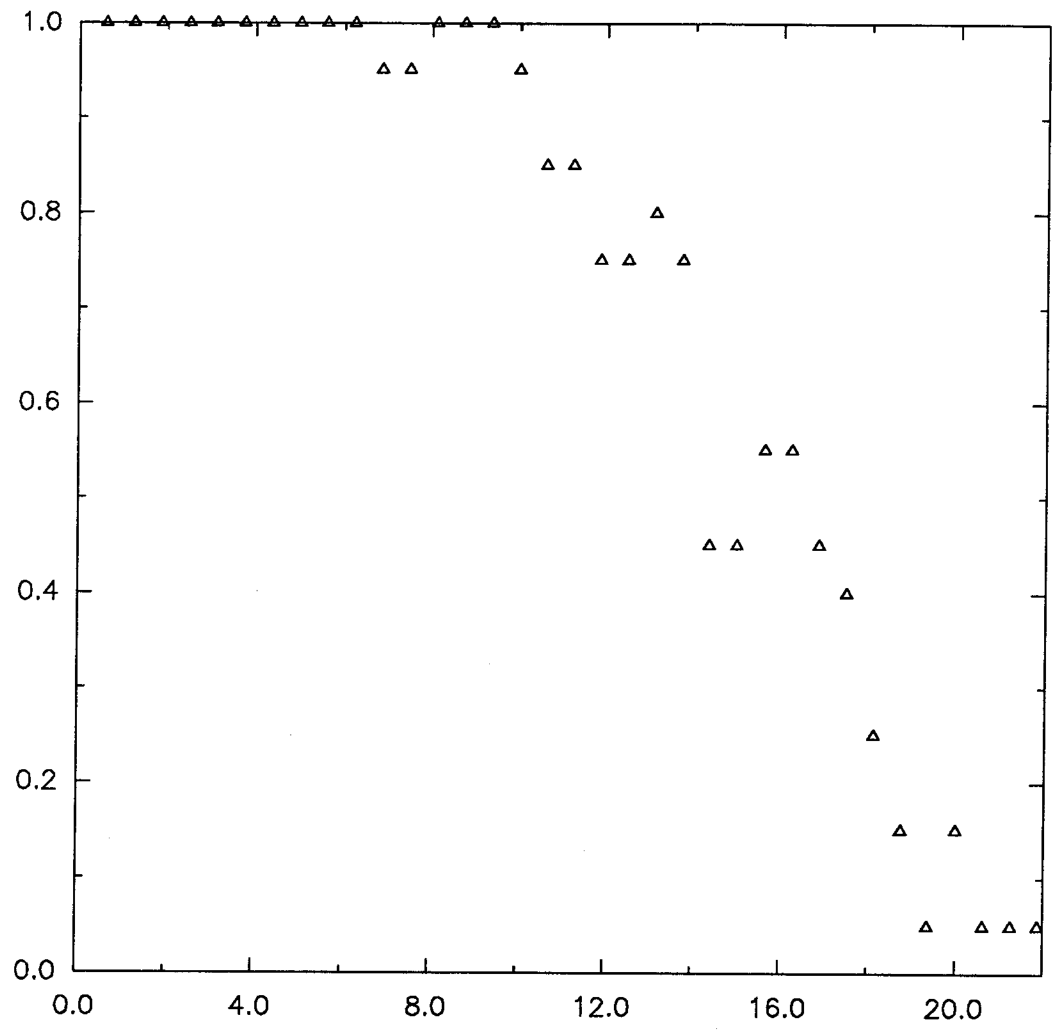


Figure 3.3: Monte Carlo simulation for the probability $\Pr(|\ \hat{r} - P_{be}\ | \geq \epsilon')$. The horizontal axis: $\frac{2M}{N}$, the vertical axis: $\Pr(|\ \hat{r} - P_{be}\ | \geq \epsilon')$.

# Bibliography

[1] W. Feller, *An In'troduction to Probability Theory and Its Applications*, New York: John Wiley and Sons, 1968.

[2] M.G . Kendall, "Proof of Relations Connected with The Tetrachoric Series and Its Generalization,", 196-198.

[3] Y.L . Tong, *Probability Inequalities in Multivariate Distributions*, Academic Press, New York, 1980.

# Chapter 4

# Storage Capacity of Two-Layer Networks with Binary Weights

## 4.1 Introduction

The information capacity is one of the most important quantities for multilayer feedforward networks, since it characterizes the sample complexity that is needed for generalization. For two layer $(N - L - 1)$ feedforward networks with real weights, it has been shown by Baum [3] that the capacity $C$ satisfies the relation $O(W) \leq C \leq O(WlnL)$, where $W$ is the total number of weights, $L$ is the number of hidden units, and $N$ is the input dimension. Although in practical hardware implementations only networks with discrete weights are of prime interest, few theoretical results exist regarding multilayer networks of this kind. The work presented here, which is an extension of our earlier work [7], tends to fill this gap and to provide a theoretical basis for using two layer networks with binary weights.

Upper and lower bounds for the capacity of two-layer $(N - 2L - 1)$ threshold networks, which have binary weights, integer thresholds for the hidden units and zero threshold for the output unit, will be established by the following steps. The statistical capacity of a specifically constructed network is first evaluated using a statistical approach and then used as a lower bound for the capacity $C$ in general. Specifically, the capacity $C$ is shown to be $O(\frac{W}{lnW})$, where $W$ is the total number of weights of the network. An upper bound is then obtained through a simple counting argument, and shown to be $O(W)$. Therefore, we have $O(\frac{W}{lnW}) \leq C \leq O(W)$.

## 4.2 Evaluation of the Lower Bound

### 4.2.1 Construction of the Network

Motivated by the idea which employs the grandmother-cell type of networks used by Baum in [3], our network, given in Fig. 4.1, is constructed by grouping the $2L$ hidden units in $L$ pairs. The two weights between each pair and the output unit are chosen to be $+1$ and $-1$. The hidden units are allowed to have integer thresholds in the range $[-C'N, C'N]$ and the threshold for the output unit is zero, where $C' = \max[1, (1+k)\sigma\sqrt{\frac{2}{\pi}}]$ with $\sigma^2$ being the variance of the input samples and $0 < k < 1$.

Without loss of generality, we assume that among total $LM + L_2M$ samples; $LM$ of them, denoted as a sample set $\{X_{LM}\}$, are assigned to class 1 and have labels 1, while $L_2M$ of them, denoted as $\{X_{L_2M}\}$ assigned to class 2 and have labels 0. Here $L, M$ are large, and $L \geq L_2 \geq 0$.

The weights $w_{li}$'s at the first layer for the $l$-th pair of hidden units ($1 \leq l \leq L$ and $1 \leq i \leq N$) are equal and constructed using $M$ out of $LM$ samples assigned to

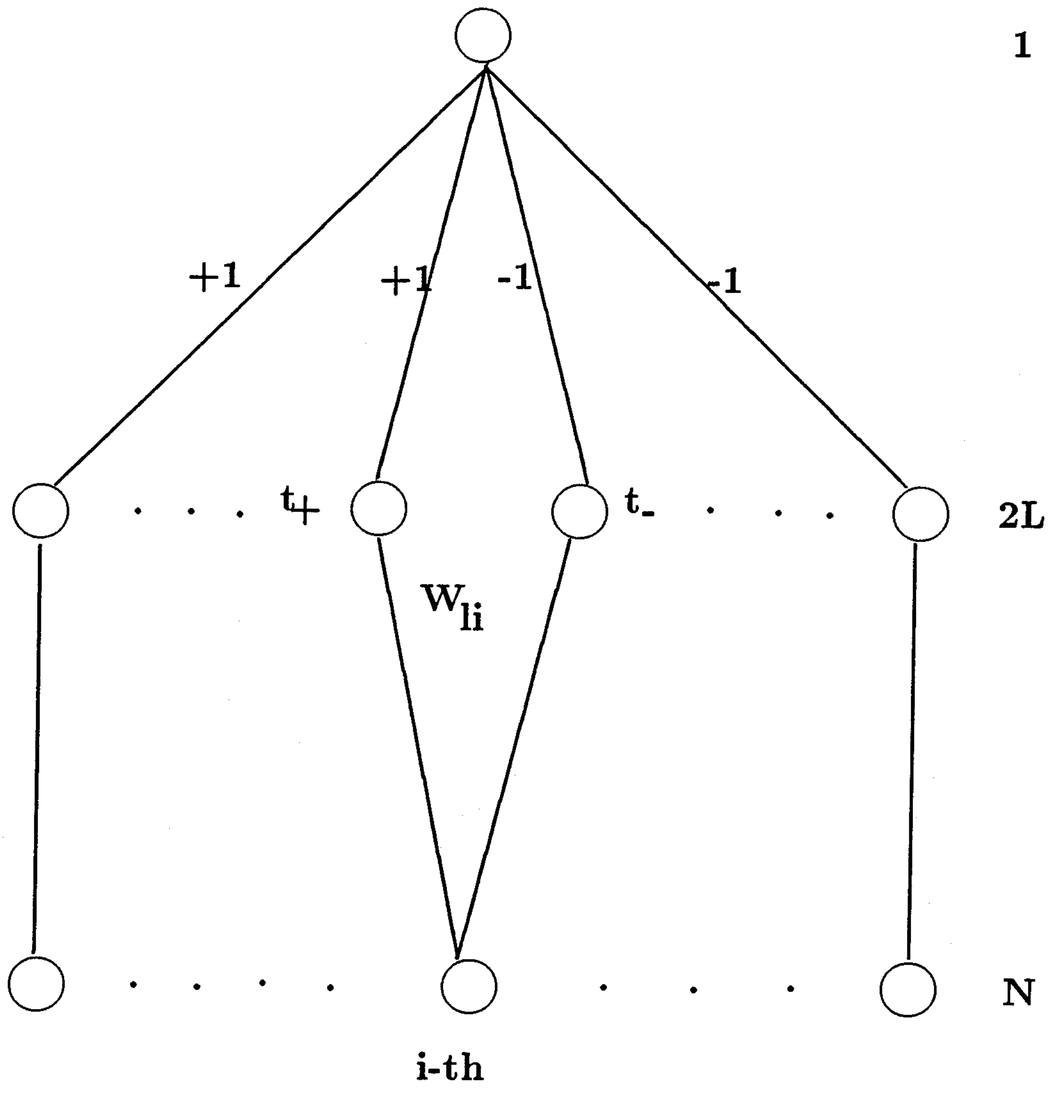


Figure 4.1: Two layer networks with binary weights and integer thresholds

class 1. Specifically,

$$w_{li} = sgn(\sum_{m=1}^{M} x_{li}^{(m)}), \tag{4.1}$$

where $sgn(x) = 1$ if $x > 0$, and $-1$ otherwise; and $1 \leq i \leq N$. $x_{li}^{(m)}$'s are the elements of the $N$-dimensional random vector $\vec{X}_l^{(m)} = (x_{l1}^{(m)}, ..., x_{lN}^{(m)})$, which are drawn independently from a continuous distribution $h(x)$ that has a compact support in $x$ and symmetric about the origin. $Ex = 0$ and $Ex^2 = \sigma^2$ are also satisfied here. Also the two thresholds for the two hidden units in this pair are given as

$$t_{l\pm} = \lfloor (1 \mp k)\sigma\sqrt{\frac{2}{\pi}}\frac{N}{\sqrt{M}} \rfloor, \tag{4.2}$$

where the subscripts $+$ and $-$ correspond to the units with weight $+1$ and $-1$ at the second layer respectively.

Each pair of hidden units constructed can then be viewed as two random parallel hyperplanes. If neither $M$ nor the separation between the two random hyperplanes are too large, these random hyperplanes can separate samples belonging to different classes with high probability by having the samples in $\{X_{LM}\}$ fall in between these parallel hyperplanes, while the samples $\{X_{L_2M}\}$ fall outside. The maximum number of samples whose arbitrary assignments to two classes can be dichotomized by the network with probability almost 1 is defined to be the capacity of the network. Precisely, the capacity of the network can be defined as follows.

**Definition 4.1** *Let $Y$ be the number of incorrectly classified samples out of total $LM + L_2M$ samples by the constructed network. Suppose the probability $\Pr(Y = 0)$, as a function of $M$, has a sharp transition point $C_0$, so that for any $\gamma > 0$ arbitrarily small, when $(L + L_2)M \leq (1 - \gamma)C_0$, $\Pr(Y = 0) \geq 1 - \eta$, and when $(L + L_2)M > (1 + \gamma)C_0$, $\Pr(Y = 0) < \eta$, where $\eta \to 0$ when $N, L \to \infty$. Then we call $C_0$ the capacity of this network.*

In what follows, we will evaluate the probability $\Pr(Y=0)$ first, then find its unique sharp transition point, which is the capacity $C_0$ for this constructed network.

### 4.2.2 Probability of Error for a Single Sample

As the first step to evaluate $C_0$, we compute $P_{e1}$ which is the probability of incorrect classification of a single random sample stored. Let $y_l^{(m)}$ denote the output of the network when the $m$-th sample $\vec{X}_l^{(m)}$ stored in the $l$-th pair is fed through the network, where $1 \le l \le L$ and $1 \le m \le M$. Without loss of generality, we can let $m = l = 1$. Since the labels for the stored samples are all $(+1)$, we say that an error occurs if $y_1^{(1)} = 0$. Then the probability of error for classifying one stored sample can be expressed as $P_{e1} = \Pr(y_1^{(1)} = 0)$.

By the construction of the network, the presynaptic input of any pair to the output unit, which is the difference between the outputs of the two threshold hidden units, can only take two possible values: 2 and 0. Let $s_l^{(1)}$ be the presynaptic input of the $l$-th pair of hidden units to the output unit, and let $S^{(1)} = \sum\limits_{l=2}^{L} s_l^{(1)}$. Then

$$\begin{aligned} P_{e1} &= \Pr(y_1^{(1)} = 0) \\ &= \Pr(s_1^{(1)} = 0, S^{(1)} = 0) \\ &= \Pr(s_1^{(1)} = 0) \prod_{l=2}^{L} \Pr(s_l^{(1)} = 0). \end{aligned} \qquad (4.3)$$

Here the multiplication of the probabilities is due to the mutual independence of $s_l^{(1)}$ $(1 \le l \le L)$ which is easy to verify.

The presynaptic input $z_{11}{}^{(1)}$ to the two units at the first pair is

$$z_{11}^{(1)} = \sum_{i=1}^{N} w_{1i} x_{1i}^{(1)}, \qquad (4.4)$$

where the general notation $z_{lj}{}^{(m)}$ $(1 \le l, j \le L$ and $1 \le m \le M)$ denotes the presynaptic input to the hidden units at the $l$-th pair when the $m$-th sample $\vec{X}_j^{(m)}$

stored at the $j$-th pair is fed through the net. Since different terms in summation (4.4) are independent, by the large-deviation theorem [4], for $N$ large and $| t_{l\pm} - Ez_{11}^{(1)} | < O(N^{\frac{1}{2}})$, we have

$$\begin{aligned} \Pr(s_1^{(1)} = 0) &= \Pr(z_{11}^{(1)} > t_{1-}) + \Pr(z_{11}^{(1)} < t_{1+}) \\ &= 2Q(-k\sqrt{\frac{N'}{M}}), \end{aligned} \tag{4.5}$$

$$\Pr(s_1^{(1)} = 2) = 1 - 2Q(-k\sqrt{\frac{N'}{M}}), \tag{4.6}$$

where $N' = \frac{2}{\pi}N$ and $Q(-x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{-x} e^{\frac{-u^2}{2}} du$. The evaluation of the equation (4.5) is given in Appendix A.

Similarly, we can obtain, for $2 \leq l \leq L$

$$\begin{aligned} \Pr(s_l^{(1)} = 2) &= \Pr(t_{l-} < z_{l1}^{(1)} < t_{l+}) \\ &= Q(-(1-k)\sqrt{\frac{N'}{M}}) - Q(-(1+k)\sqrt{\frac{N'}{M}}), \end{aligned} \tag{4.7}$$

$$\Pr(s_l^{(1)} = 0) = 1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}}). \tag{4.8}$$

Plugging equations (4.5), (4.8) into equation (4.3), we can obtain

$$P_{e1} = 2Q(-k\sqrt{\frac{N'}{M}})[1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{(L-1)}. \tag{4.9}$$

### 4.2.3 Probability of Error for All Samples together

Let

$$Y = Y_1 + Y_2, \tag{4.10}$$

with

$$Y_1 = \sum_{l=1}^{L}\sum_{m=1}^{M} I(y_l^{(m)} = 0), \tag{4.11}$$

and

$$Y_2 = \sum_{j=1}^{L_2 M} I(y_j = 0), \tag{4.12}$$

where $I(A) = 1$ if event A occurs, and $I(A) = 0$ otherwise. $y_j$ is the output of the network when the $j$-th sample in set $\{X_{L_2M}\}$ is fed through the network, where $j \in [1, L_2M]$. Then $Y_1$ and $Y_2$ are random variables representing the number of incorrectly classified samples in $\{X_{LM}\}$ and $\{X_{L_2M}\}$ respectively. Likewise, $Y$ is the total number of incorrectly classified samples. To evaluate the capacity $C_0$ for this network, we need to consider the probability $\Pr(Y = 0)$, which is the probability that all samples are classified correctly. In the theorem below, we will first show that $\Pr(Y = 0) = \Pr(Y_1 = 0)\Pr(Y_2 = 0)$.

**Theorem 4.1** *$Y_1$ and $Y_2$ are independent, and*

$$\Pr(Y = 0) = \Pr(Y_1 = 0)\Pr(Y_2 = 0). \tag{4.13}$$

*Furthermore, the $I(y_j = 0)$'s in equation (4.12) are mutually independent, i.e.*

$$\Pr(Y_2 = 0) = [1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{LL_2M}. \tag{4.14}$$

The proof of the theorem is given in Appendix B.

Then what left is to find $\Pr(Y_1 = 0)$. The difficulty in doing that is that the terms in the summation (4.11) are dependent random variables. By means of a theorem given by Stein, we will show that $\Pr(Y_1 = 0)$ has approximately an Poisson distribution under certain conditions.

**Theorem 4.2** *(Stein [7])*

*Let*

$$Z = \sum_{n=1}^{\hat{N}} I_n, \tag{4.15}$$

*where $I_n$'s are Bernoulli random variables taking values 1 or 0 and $EI_n = P_n$. Let $k = 1, ..., \hat{N}$. Define*

$$Z_k' = \sum_{n=1}^{\hat{N}} I_{nk}', \tag{4.16}$$

*such that the distribution of $Z_k'$ is the same as the conditional distribution of $Z$ given $I_k = 1$. Then*

$$|\Pr(Z = K) - P_\lambda(K)| \le \min(\lambda^{-1}, 1) \sum_{k=1}^{\hat{N}} P_k E|Z - Z_k' + 1|, \tag{4.17}$$

*where $P_\lambda(K) = e^{-\lambda}\frac{\lambda^K}{K!}$, and $\lambda = EZ$.*

Appendix C gives the proof of the theorem.

Roughly speaking, this theorem says that the random variable $Z$ has an approximately Poisson distribution if the distribution of $Z$ is nearly the same as its conditional distribution given $I_k = 1$.

In our problem, in order to show the distribution of $Y_1$, can be approximated by Poisson distribution, we define a random variable $Y_k'$ as follows. Let $I_i = I(y_l^{(m)} = 0)$ for $1 \le i \le LM$, $1 \le l \le L$ and $1 \le m \le M$. Then

$$Y_1 = \sum_{i=1}^{LM} I_i. \tag{4.18}$$

Define

$$Y_k' = 1 + \sum_{i \ne k} I_{ik}', \tag{4.19}$$

where $k \in [1, ..., LM]$, and

$$I_{ik}' = \begin{cases} I_i & \text{if } I_k = 1 \text{ and } i \ne k; \\ 1 & \text{if } I_k = 1 \text{ and } i = k. \end{cases}$$

Then the distribution of $Y_k'$ is the same as the conditional distribution of $Y_1$ given $I_k = 1$. Therefore,

$$|\Pr(Y_1 = 0) - P_{\lambda_1}(0)| \le \min({\lambda_1}^{-1}, 1) \sum_{k=1}^{LM} P_k E|Y_1 - Y_k' + 1|, \tag{4.20}$$

where $P_{\lambda_1}(0) = e^{-\lambda_1}$, and $\lambda_1 = EY = LMP_{e1}$ with $P_{e1}$ given in equation (4.3). Then the theorem below gives the condition for the Poisson hypothesis.

**Theorem 4.3** *When* $M < \min[\frac{3k^2N'}{4(lnL+lnN)}, \frac{k^2N'}{2lnL+lnN}, \frac{(1-k)^2N'}{2lnL}]$, *we have*

$$\Pr(Y_1 = 0) \sim e^{-\lambda_1}, \tag{4.21}$$

*where* $\lambda_1 = LMP_{e1}$ *and* $0 < k < 1$.

The proof is given in Appendix D. Combining equation (4.21) and equation (4.14), we have

$$\Pr(Y = 0) \approx e^{-\lambda_1}[1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{LL_2M} . \tag{4.22}$$

The Poisson hypothesis guarantees that $\Pr(Y_1 = 0)$ is a monotonic function of $M$. $\Pr(Y_2 = 0)$ is monotonic also. Then a unique critical point exists for $\Pr(Y = 0)$, which leads to the results stated in Theorem 4.4.

**Theorem 4.4** *Assuming that* $\frac{(1-k)^2N'}{2lnL} \geq \min[\frac{k^2N'}{2(lnL+lnN)}, \frac{(1-k)^2N'}{2(lnLL_2+lnN)}]$. *Then for any given* $\gamma > 0$ *arbitrarily small,*

*a) When* $L_2 = 0$, $M = (1-\gamma)\frac{k^2N'}{2(lnL+lnN)}$;

*b) When* $L_2 > 0$, $M = \min[(1-\gamma)\frac{k^2N'}{2(lnL+lnN)}, (1-\gamma)\frac{(1-k)^2N'}{2(lnLL_2+lnN)}]$,

$\Pr(Y = 0) \sim 1 - \eta$, *where* $L \geq L_2 \geq 0$, *and* $\eta \sim O(\frac{1}{(LN)^\alpha}) \to 0$ *when* $N, L \to \infty$ *and* $\alpha > 0$, *then we have*

$$\begin{aligned} C_0 &= (L + L_2)M & (4.23)\\ &\sim O(\frac{W}{lnW}), & (4.24)\end{aligned}$$

*where* $W$ *is the total number of weights of the network.*

Appendix E provides the proof for this theorem.

Fig. 4.2 gives both the probability obtained in equation (4.22) and the Monte Carlo simulations which evaluate this probability at different values of $M$ over six runs shown by the triangular markers. The solid curve corresponds to the probability obtained in equation (4.22). In the simulations, the samples are drawn from the uniform distribution in $[-a, a]$, where $a = (\frac{3}{2})^{\frac{1}{3}}$. $N = 2500$, $L = 50$, $k = .5$. It demonstrates that the theoretical result is in a good agreement with the simulations.

### 4.2.4 A Lower Bound for the Capacity of Two Layer $(N-2L-1)$ Networks With Binary Weights

Let $C$ be the capacity of a class of two layer $(N-2L-1)$ networks with binary weights and integer thresholds for the hidden units and zero threshold for the output unit. A lower bound of $C$ is defined to be the number of samples in general position whose arbitrarily assignments to two classes can be dichotomized by at least one network in this class with probability almost 1.

In our construction in Section 4.2.1, a specified continuous distribution from which samples are drawn ensures that these samples are in general position with probability 1. Since the network constructed is able to implement, with probability almost 1, any dichotomy of the samples up to the capacity $C_0$, $C_0$ can be used as a lower bound for the capacity $C$ of the class of networks of the same structure by definition. That is, $C_0 \leq C$.

## 4.3 Evaluation of an Upper Bound

An upper bound for the capacity $C$ is defined to be the number of samples whose arbitrary assignments can be implemented by any network in the class with probability almost zero. This will happen when the total number of possible binary mappings

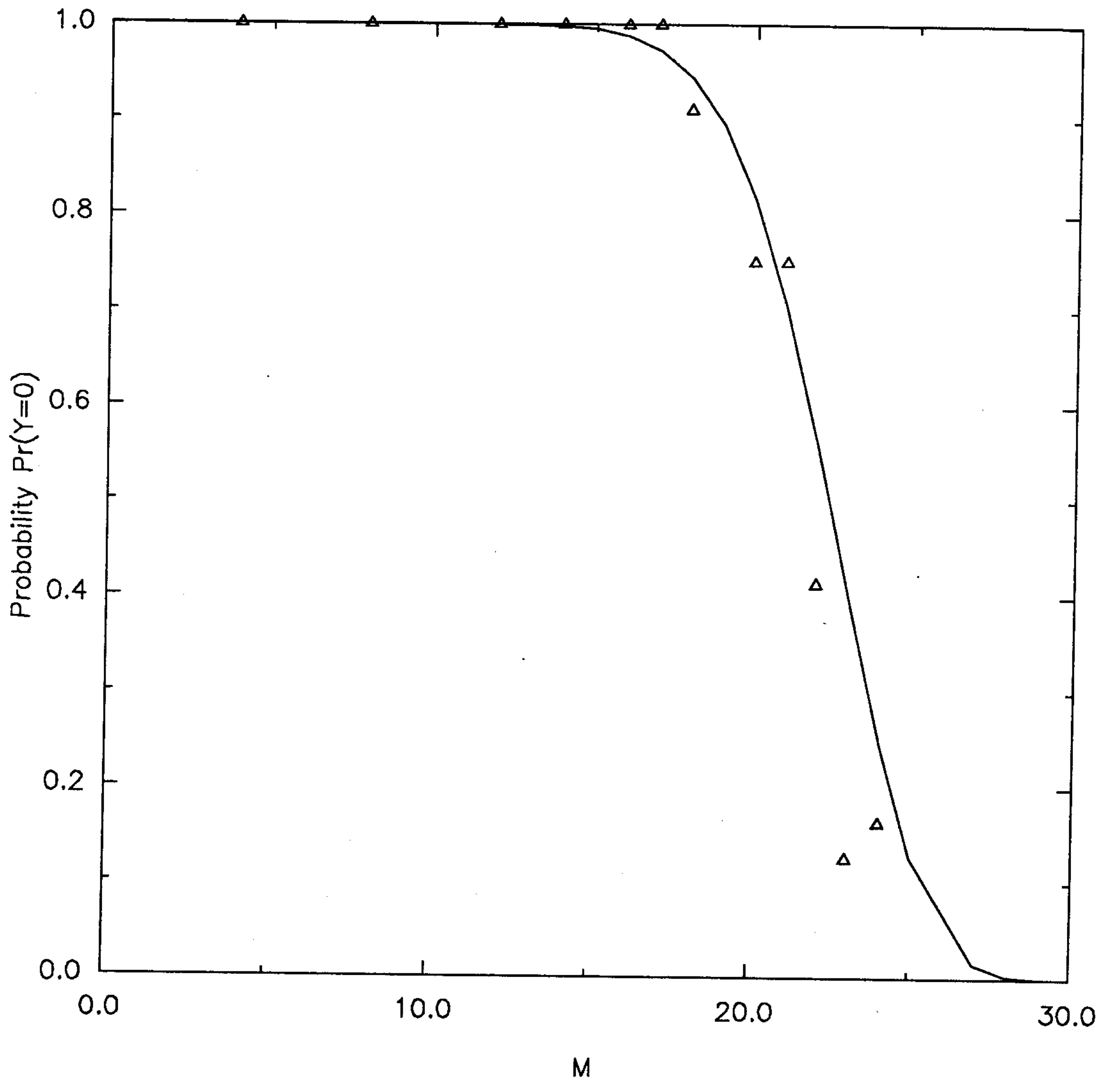


Figure 4.2: The capacity curve.

generated by the networks is smaller than all possible dichotomies of the samples. The total number of binary mappings the networks can possibly generate, however, is no larger than $2^{W+2L\log C'2N}$. Then $(1+\gamma)(W+2L\log 2C'N)$ is an upper bound for the capacity $C$. This is on the order of $W$ when $N$ and $L$ are large, where $\gamma > 0$ arbitrarily small. Therefore $C \leq O(W)$ is obtained.

## 4.4 Conclusion

Combining both lower and upper bounds, we have

$$O(\frac{W}{lnW}) \leq C \leq O(W). \tag{4.25}$$

Compared with the capacity of two layer networks with real weights, the results here show that reducing the accuracy of the weights to just two bits leads to a loss of capacity at most only a factor of $lnW$. This gives strong theoretical support to the notion that multilayer networks with binary interconnections are capable of implementing complex functions. This $lnW$ factor difference between the lower and upper bounds, however, may be due to the limitations of the grandmother-cell type of networks we use. A tighter lower bound could perhaps be obtained if a better construction of a network could be found.

# Bibliography

[1] A. D . Barbour and G. K . Eagleson, "Poisson Approximation for Some Statistics Based on Exchangeable Trials", *Adv. Appl. Prob.*, Vol. 15, 585-600, 1983.

[2] A. D . Barbour and L. Holst, "Some Applications of the Stein-Chen Method for Proving Poisson Convergence", *Adv. Appl. Prob.*, Vol. 21, 74-90, 1989.

[3] E. Baum, " On the Capacity of Multilayer Perceptron", *J. of Complexity*, Vol. 4, 193-215, 1988.

[4] W. Feller, *An Introduction to Probability Theory and Its Applications*, New York: John Wiley and Sons, 1968.

[5] C. Ji and D. Psaltis, "The Capacity of a Two Layer Network with Binary Weights", *IJCNN*, 1991

[6] M.G . Kendall, "Proof of Relations Connected with the Tetrachoric Series and Its Generalization,", 196-198.

[7] C. Stein, "Approximate Computation of Expectations", *Inst. Math. Statis. Lecture Notes, Monograph Series Vol. 7, Hayway, California*, 1988

[8] S. Venkatesh, *Ph.D. dissertation, Caltech*, 1986

## Appendix A: Evaluation of the Equation (4.5)

$$\begin{aligned}
Ez_{11}^{(1)} &= NEw_{11}x_{11}^{(1)} \\
Ew_{11}x_{11}^{(1)} &= \int_{-\infty}^{+\infty} x_{11}^{(1)} E(w_{11} \mid x_{11}^{(1)}) h(x_{11}^{(1)}) dx_{11}^{(1)} \\
&= \int_{x \in D} x E(sgn(x + \sum_{m=2}^{M} x_{11}^{(m)}) \mid x) h(x) dx \\
&\approx \int_{x \in D} x[1 - 2Q(-\frac{x}{\sqrt{(M-1)}\sigma})] h(x) dx \qquad (4.26) \\
&\approx \sqrt{\frac{2}{\pi \sigma^2 M}} \int_{x \in D} x^2 h(x) dx \qquad (4.27) \\
&= \sqrt{\frac{2}{\pi M}} \sigma, \qquad (4.28)
\end{aligned}$$

where $D$ is the compact support of $x$. Equation (4.26) is obtained due to the fact that for $M$ large, $\Pr(\sum_{m=2}^{M} x_{11}^{(m)} > -x)$ and $\Pr(\sum_{m=2}^{M} x_{11}^{(m)} < -x)$ can be approximated by $1 - Q(-\frac{x}{\sqrt{(M-1)}\sigma})$ and $Q(-\frac{x}{\sqrt{(M-1)}\sigma})$ respectively. Moreover, when $M \gg \max \mid x \mid$, which is true since $x$ has compact support, $Q(-\frac{x}{\sqrt{(M-1)}\sigma}) \approx \frac{1}{2} - \frac{x}{\sqrt{2\pi(M-1)}\sigma}$ through the Taylor expansion. Then equation (4.28) is obtained. Therefore, we have that the mean equals to $N\sqrt{\frac{2}{\pi M}}\sigma$. Similarly, we can show that the variance of $z_{11}^{(1)}$ is approximately $\sqrt{N}\sigma$. Then the result in equation (4.5) is obtained.

## Appendix B: Proof of Theorem 4.2 [2]

Proof:

For any bounded $h : Z^+ \rightarrow$ R,

$$\begin{aligned}
E(\lambda h(Z+1) - Zh(Z)) &= \sum_{k=1}^{\hat{N}} [P_k E(h(Z+1)) - E(I_k h(Z))] \\
&= \sum_{k=1}^{\hat{N}} P_k [E(h(Z+1)) - E(h(Z \mid I_k = 1))] \\
&= \sum_{k=1}^{\hat{N}} P_k E(h(Z+1) - h(Z_k^{'})).
\end{aligned}$$

For any integers $i, j \geq 0$ we have

$$\mid h(i) - h(j) \mid \leq \mid i - j \mid \Delta h$$

where

$$\Delta h = \sup_{j \geq 0} \mid h(j+1) - h(j) \mid .$$

Hence

$$\begin{aligned} E(\lambda h(Z+1) - Zh(Z)) &\leq \sum_{k=1}^{\hat{N}} P_k E \mid h(Z+1) - h(Z_k') \mid \\ &\leq \sum_{k=1}^{\hat{N}} P_k E \mid Z - Z_k' + 1 \mid \Delta h. \end{aligned}$$

Let $g$ be defined recursively by

$$g(0) = 0,$$

$$\lambda g(j+1) - jg(j) = I(j \in A) - \Pr(U_\lambda \in A),$$

where $A \subset Z^+$ and $U_\lambda$ is the Poisson random variable with mean $\lambda = EZ$. In Lemma 4 [1] it is proved that

$$\sup_j \mid g(j) \mid \leq \min(1, 1.4\lambda^{-\frac{1}{2}}),$$

$$\Delta g = \sup_{j \geq 0} \mid g(j+1) - g(j) \mid \leq \lambda^{-1}(1 - e^{-\lambda}) \leq \min(1, \lambda^{-1}).$$

Thus

$$\begin{aligned} |\Pr(Z \in A) - \Pr(U_\lambda \in A)| &= |E(\lambda g(Z+1) - Zg(Z))| \\ &\leq \min(1, \lambda^{-1}) \sum_{k=1}^{\hat{N}} P_k E|Z - Z_k' + 1|, \end{aligned}$$

proving the assertion.

## Appendix C: Proof of Theorem 4.1

Proof:

First, we show that $Y_1$ and $Y_2$ are independent.

Consider the presynaptic inputs of the first pair of hidden units to the output unit when the samples $X_l^{(m)} \in \{X_{LM}\}$ and $X_j \in \{X_{L_2M}\}$ are fed through the network. Without loss of the generality, we can choose $l = m = j = 1$. Then we have

$$z_{11}^{(1)} = \sum_{i=1}^{N} w_{1i} x_{1i}^{(1)}, \tag{4.29}$$

and

$$\hat{z}_{11}^{(1)} = \sum_{i=1}^{N} w_{1i} x_{1i}, \tag{4.30}$$

where $x_{1i}$'s are the elements of $X_1$ for $i \in [1, N]$. Since the terms with different subscripts $i$ are independent, which is easy to check, we first show the independence of the two terms with the same subscript $i$ in the above two summations. Let $u = w_{1i}x_{1i}^{(1)}$ and $v = w_{1i}x_{1i}$ for $i \in [1, N]$. Then for any $a, b \in (-\infty, +\infty)$,

$$\begin{aligned}
\Pr(u < a, v < b) &= \Pr(u < a, v < b \mid w_{1i} = 1)\Pr(w_{1i} = 1) \\
&+ \Pr(u < a, v < b \mid w_{1i} = -1)\Pr(w_{1i} = -1), && (4.31) \\
&= \Pr(u < a \mid w_{1i} = 1)\Pr(x_{1i} < b)\Pr(w_{1i} = 1) \\
&+ \Pr(u < a \mid w_{1i} = -1)\Pr(x_{1i} > -b)\Pr(w_{1i} = -1) && (4.32) \\
&= \frac{1}{2}[\Pr(u < a \mid w_{1i} = 1) + \Pr(u < a \mid w_{1i} = -1)] \\
&\times \Pr(x_{1i} < b). && (4.33)
\end{aligned}$$

Here equation (4.32) is obtained from equation (4.31) due to the independence of the samples; while equation (4.33) is derived from equation (4.32) since $\Pr(w_{1i} = 1) = \Pr(w_{1i} = -1) = \frac{1}{2}$, $x_{1i}$ is independent of $w_{1i}$ and $x_{1i}$ is symmetrically distributed around the origin, i.e. $\Pr(x_{1i} < b) = \Pr(x_{1i} > -b)$. On the other hand,

$$\begin{aligned}
\Pr(u < a)\Pr(v < b) &= [\Pr(u < a \mid w_{1i} = 1)\Pr(w_{1i} = 1) \\
&+ \Pr(u > -a \mid w_{1i} = -1)\Pr(w_{1i} = -1)] \\
&\times [\Pr(x_{1i} < b)\Pr(w_{1i} = 1) \\
&+ \Pr(x_{1i} > -b)\Pr(w_{1i} = -1)], \qquad (4.34) \\
&= \frac{1}{2}\Pr(x_{1i} < b)[\Pr(u < a \mid w_{1i} = 1) \\
&+ \Pr(u > -a \mid w_{1i} = -1)]. \qquad (4.35)
\end{aligned}$$

Therefore, $\Pr(u < a, v < b) = \Pr(u < a)\Pr(v < b)$, i.e. $u$ and $v$ are independent.

This approach can be extended to all $N$ variables in summation 4.29 and 4.30 to show the mutual independence of all terms. Then $z_{11}^{(1)}$ and $\hat{z}_{11}^{(1)}$ are independent. Similarly, we can show the independence for the other pairs of hidden units. Therefore, $I(y_l^{(m)})$ and $I(y_j)$ are independent. The mutual independence of $I(y_l^{(m)})$ and $I(y_j)$ for all $l \in [1, L]$, $m \in [1, M]$ and $j \in [1, L_2M]$ can be shown using a similar approach extended to multiple variables. Then $Y_1$ and $Y_2$ can be shown to be independent.

Similarly, we can show that $I(y_j)$'s for $j \in [1, L_2M]$ are also mutually independent. Then

$$\begin{aligned}
\Pr(Y_2 = 0) &= \Pr^{L_2M}(I(y_1 = 0)) \\
&= [1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{LL_2M} . \qquad (4.36)
\end{aligned}$$

Q.E.D.

## Appendix D: Proof of Theorem 4.3

Proof:

Due to the symmetry of $I_i$'s $(1 \leq i \leq LM)$, $E \mid Y_1 - Y_k{}' + 1 \mid = E \mid Y_1 - Y_1{}' + 1 \mid$ for all $k \in [1, ..., LM]$. Then

$$\min(\lambda_1{}^{-1}, 1)\sum_{k=1}^{LM} P_k E|Y_1 - Y_k' + 1| = \min(\lambda_1, 1)E|Y_1 - Y_1' + 1|. \tag{4.37}$$

By Jensen's inequality, $E|Y_1 - Y_1' + 1| \leq \sqrt{E(Y_1 - Y_1' + 1)^2}$. To show that the Poisson hypothesis holds, it suffices to find a condition on $M$ such that

$$\min(\lambda_1, 1)\sqrt{E(Y_1 - Y_1' + 1)^2}$$

is asymptotically small for $N, L$ large, where $\lambda_1 = LMP_{e1}$. Using the basic relations

$$\begin{aligned} EI_1 I_{i1}' &= EI_1 E(I_i \mid I_1) \\ &= P_{e1} \Pr(I_i = 1 \mid I_1 = 1, i \neq 1), \end{aligned} \tag{4.38}$$

$$\begin{aligned} EI_i I_{i1}' &= EI_i E(I_i \mid I_1, I_i) \\ &= P_{e1}, \end{aligned} \tag{4.39}$$

and for $(j \neq i)$

$$\begin{aligned} EI_j I_{i1}' &= EI_j E(I_i \mid I_1, I_j) \\ &= P_{e1} \Pr(I_i = 1 \mid I_1 = 1, I_j = 1), \end{aligned} \tag{4.40}$$

we can then obtain

$$E(Y_1 - Y_1' + 1)^2 = aL^2M^2 + bLM^2 + cLM + dM^2 + eM + f, \tag{4.41}$$

where

$$\begin{aligned} a &= P(x, y) + P(x, y \mid z) - 2P(x)P(x \mid y, z) \qquad (4.42) \\ b &= P(x, x') - P(x, y) + 2(x, y \mid x') + P(x, x' \mid y) \\ &\quad -3P(x, y \mid z) - 2P(x)P(x \mid x', y) - 2P(x)P(y \mid x, x') \end{aligned}$$

$$
\begin{aligned}
& +6P(x)P(x \mid y,z) - 2P(x)P(x \mid y,x') && (4.43)\\
c \;=\; & P(x) - P(x,y) - 2P(x,y \mid x') + P(x \mid y) - P(x,x' \mid y) \\
& -2P(x,y) + 2P(x)P(x \mid x',y) + 2P(x)P(y \mid x,x') \\
& -2P(x) + 2P(x)P(x \mid x',y) && (4.44)\\
d \;=\; & -2P(x,y \mid x') + 2P(x,y \mid z) - 2P(x)P(x \mid x',x'') \\
& +2P(x)P(x \mid x',y) + 2P(x)P(y \mid x,x') \\
& -4P(x)P(x \mid y,z) + 2P(x)P(x \mid x',y) && (4.45)\\
e \;=\; & -3P(x,x' \mid y) + 2P(x,y \mid x') + P(x,x' \mid y) \\
& +6P(x)P(x \mid x',x'') - 2P(x,x') - 2P(x)P(x \mid x',y) \\
& +2P(x,y) - 2P(x)P(y \mid x,x') - 2P(x)P(x \mid x',y) && (4.46)\\
f \;=\; & -P(x \mid y) + 2P(x,x' \mid y) + 2P(x,x') + 2P(x) \\
& -2P(x)P(x \mid x',x''), && (4.47)
\end{aligned}
$$

where $x, x', x'', y$ and $z$ represent the random events as follows.

$$
\begin{aligned}
x &: I(y_{l_1}^{(m_1)} = 0) = 1 \\
x' &: I(y_{l_1}^{(m_2)} = 0) = 1 \\
x'' &: I(y_{l_1}^{(m_3)} = 0) = 1 \\
y &: I(y_{l_2}^{(m)} = 0) = 1 \\
z &: I(y_{l_3}^{(m)} = 0) = 1 \\
P(x) &: P_{e1},
\end{aligned}
$$

for $l_1 \neq l_2 \neq l_3$, $m_1 \neq m_2 \neq m_3$, all $l$'s $\in [1, ..., L]$ and $m$'s $\in [1, ..., M]$. That is, $(x, y, z)$ and $(x, x', x'')$ represent the occurrences of classification errors for three samples stored at three different pairs and the same pair respectively.

To find the joint distributions in the above equations, let us first consider $P(x, x')$.

Due to the symmetry of the random variables, we take $m_1 = l_1 = 1$ and $m_2 = 2$. Then

$$\begin{aligned} P(x,x') &= \Pr(I(y_1^{(1)}=0)=1, I(y_1^{(2)}=0)=1) \\ &= \Pr(s_1^{(1)}=0, s_1^{(2)}=0) \\ &\times [1-Q(-(1-k)\sqrt{\frac{N'}{M}})+Q(-(1+k)\sqrt{\frac{N'}{M}})]^{(2L-2)} . \end{aligned} \quad (4.48)$$

This equation is obtained due to the fact only the presynaptic inputs of the first pair to the output unit are dependent.

Let $z_{lj}^{(m)}$ be the presynaptic input to the threshold units at the $l$-th pair for the stored sample $\vec{X}_j^{(m)}$ being fed through. Then

$$z_{lj}^{(m)} = \sum_{i=1}^{N} w_{li} x_{ji}^{(m)}. \quad (4.49)$$

It is easy to show that $z_{lj}^{(m_1)}$ and $z_{lj}^{(m_2)}$ are mutually independent for all $l \neq j$, where $m_1 \neq m_2$. For $l = j$

$$\begin{pmatrix} z_{ll}^{(m_1)} \\ z_{ll}^{(m_2)} \end{pmatrix} = \sum_{i=1}^{N} w_{li} \begin{pmatrix} x_{li}^{(m_1)} \\ x_{li}^{(m_2)} \end{pmatrix}.$$

Since all the terms in the summation are mutually independent, by the multivariate central limit theorem, for $N$ large

$$(\frac{1}{\sqrt{N}} z_{ll}^{(m_1)}, \frac{1}{\sqrt{N}} z_{ll}^{(m_2)}) \sim N(\mu, \mu, \sigma^2, \sigma^2, \rho), \quad (4.50)$$

where $N(\mu, \mu, \sigma^2, \sigma^2, \rho)$ represents the joint distribution of two gaussian random variables with the same mean $\mu$, variance $\sigma^2$ and correlation coefficient $\rho$, where

$$\begin{aligned} \mu &= \sqrt{\frac{N'}{M}}\sigma, \\ \rho &= -\tfrac{2}{\pi M}, \end{aligned}$$

with $N' = \frac{2N}{\pi}$. It can be shown in a similar fashion that $z_{lj}^{(m_1)}$ and $z_{jl}^{(m_2)}$ are also jointly normal except that the correlation coefficient $\rho$ equals $\frac{2}{\pi M}$.

$N(0,0,1,1,\rho)$, however, can be expanded as [6]:

$$N(0,0,1,1,\rho) = f(x)f(y) + \sum_{r=1}^{+\infty}\frac{\rho^r}{r!}\tau_r(x)\tau_r(y), \tag{4.51}$$

where $\tau_r(x) = (-\frac{d}{dx})^r f(x)$ and $f(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$. Then

$$\begin{aligned}\Pr(s_1^{(1)} = 0, s_1^{(2)} = 0) &= \Pr(z_{11}^{(1)} < t_+, z_{11}^{(2)} < t_+) + 2\Pr(z_{11}^{(1)} < t_+, z_{11}^{(2)} > t_-) \\ &\quad + \Pr(z_{11}^{(1)} > t_-, z_{11}^{(2)} > t_-) \\ &= 4Q^2(-\mu_0) + 2\rho^2 f'^2(\mu_0) + o(\rho^2),\end{aligned} \tag{4.52}$$

where $\mu_0 = k\sqrt{\frac{N'}{M}}$. Then

$$\begin{aligned}P(x,x') &= [4Q^2(-\mu_0) + 2\rho^2 f'^2(\mu_0) + o(\rho^2)] \\ &\times [1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{2L-2}.\end{aligned} \tag{4.53}$$

When $M < \frac{(1-k)^2 N'}{2lnL}$ such that

$$1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}}) \approx 1 - \eta' \tag{4.54}$$

is satisfied where $\eta' \to 0$ when $N, L \to \infty$, we have

$$P(x,x') \approx 4Q^2(-\mu_0) + 2\rho^2 f'^2(\mu_0) + o(\rho^2). \tag{4.55}$$

Similarly the joint distribution of the three gaussian random variables can be expressed as

$$\begin{aligned}N(0,0,0,1,1,1,\rho_{12},\rho_{13},\rho_{23}) &= f(x_1)f(x_2)f(x_3) \\ + \sum_{m}^{+\infty}\sum_{n}^{+\infty}\sum_{\substack{o \\ m+n+o=r,r=1}}^{+\infty} &\frac{\rho_{12}^m\rho_{13}^n\rho_{23}^o}{m!n!o!}\tau_{r-n}(x_1)\tau_{r-o}(x_2)\tau_{r-m}(x_3).\end{aligned} \tag{4.56}$$

Using this expansion and the similar derivations as before, we can obtain

$$P(x,y,z) \approx 8Q^3(-\mu_0) + 6\rho^2 f'^2(\mu_0)Q(-\mu_0) + o(\rho^2). \tag{4.57}$$

For $M$ large ($\rho$ small), we have

$$a \approx \rho^2 3 f'^2(\mu_0). \tag{4.58}$$

The other coefficients in Equation (4.41) can be found similarly. Then for $L$, $N$ large and $M \sim O(\frac{N}{lnN})$,

$$\begin{aligned} aL^2M^2 + bL^2M + cLM + dM^2 + eM &\approx aL^2M^2, & (4.59)\\ f &\approx 4Q(-\mu_0). & (4.60)\end{aligned}$$

Therefore, the bound B for Inequality (4.17) satisfies

$$\begin{aligned} B &= \min(\lambda_1, 1)E|Y_1 - Y_1' + 1| \\ &\leq \min(\lambda_1, 1)\sqrt{E(Y_1 - Y_1' + 1)^2} \\ &\approx \min(\lambda_1, 1)\sqrt{aL^2M^2 + f}. \end{aligned} \tag{4.61}$$

Case 1: $\frac{lnL}{lnN} > \frac{1}{2}$.

For this case, $aL^2M^2 \gg f$ for $M \sim O(\frac{N}{lnN})$. Then, when

$$M < \min[\frac{k^2N}{2lnL + lnN}, \frac{(1-k)^2N'}{2lnL}], \tag{4.62}$$

$\min(\lambda_1, 1)\sqrt{aL^2M^2 + f} \sim \sqrt{a}L^2M \to 0$ as $N \to +\infty$, where the $\frac{(1-k)^2N'}{2lnL}$ term is the condition for the equation (4.54) to be satisfied.

Case 2: $\frac{lnL}{lnN} \leq \frac{1}{2}$.

Then either $aL^2M^2 \ll f$ or $O(aL^2M^2) = O(f)$ for $M \sim O(\frac{N}{lnN})$. Therefore, when

$$M < \min[\frac{3k^2N'}{4(lnL + lnN)}, \frac{(1-k)^2N'}{2lnL}], \tag{4.63}$$

$\min(\lambda_1, 1)\sqrt{aL^2M^2 + f} \sim O(f) \to 0$ as $N \to +\infty$. Combining conditions in equation (4.62) and equation (4.63), we have that when

$$M < \min[\frac{k^2N}{2lnL + lnN}, \frac{3k^2N}{4(lnL + lnN)}, \frac{(1-k)^2N'}{2lnL}], \tag{4.64}$$

the Poisson hypothesis is satisfied.

Q.E.D.

## Appendix E: Proof of Theorem 4.4

Proof:

Let $\Pr(Y =) \approx 1 - \eta$. Then we must have

$$e^{-LMP_{e1}} = 1 - \eta_1, \tag{4.65}$$

and

$$[1 - Q(-(1-k)\sqrt{\frac{N'}{M}}) + Q(-(1+k)\sqrt{\frac{N'}{M}})]^{LL_2M} = 1 - \eta_2, \tag{4.66}$$

both satisfied simultaneously, where $\eta, \eta_1$ and $\eta_2 \to 0$ when $N, L \to \infty$. Equation (4.65) is satisfied if

$$M \leq \frac{k^2N'}{2(lnL + lnN)}. \tag{4.67}$$

Also, equation (4.66) holds for $L_2 > 0$ (for $L_2 = 0$, we have $\eta_2 = 0$) when

$$M \leq \frac{(1-k)^2N'}{2(lnLL_2 + lnN)}. \tag{4.68}$$

If we consider both cases as well as the assumption $M < \frac{(1-k)^2N'}{2lnL}$, we can find that the unique sharp transition point for $\Pr(Y = 0)$ occurs at

$$M = \begin{cases} \min[\frac{k^2N'}{2(lnL+lnN)}, \frac{(1-k)^2N'}{2(lnLL_2+lnN)}], & \text{if } 0 < L_2 \leq L; \\ \frac{k^2N'}{2(lnL+lnN)}, & \text{if } L_2 = 0. \end{cases}$$

That is,

$$\begin{aligned} C_0 &= (L + L_2)M \\ &\sim O(\frac{W}{lnW}), \end{aligned} \tag{4.69}$$

where $W$ is the total number of weights of the network.

Q.E.D.

# Chapter 5

# Network Reduction Algorithm

## 5.1 Introduction

In this chapter, a special kind of generalization is investigated which deals with learning of a smooth analog mapping from a set of discrete samples. This includes the important class of pattern association problems that originate from a natural setting, and hence are subject to continuous constraints, for example, interpolation of a smooth surface from a set of discrete visual inputs [6], and training of a robot arm moving along a specific trajectory in a smooth fashion. For this broad class of problems, the problem of generalization reduces to that of finding a network with a response function that interpolates smoothly between the training samples and which has a sufficiently low mean square error on the test samples. If the set of training samples contains enough information for the problem, the theoretical result given recently by Barron [1] guarantees that the smallest network which has reasonably good performance on the training set is the one which learns the mapping approximately. Furthermore, if some a priori knowledge is available about a problem, it can be used as a constraint which will help the learning [1]. In our learning algorithm, these two ideas are incorporated into a smoothness constraint.

One way of obtaining a network with a smooth response function is to add a measure of smoothness to the standard BEP error function as a perturbation. For example, one might average an absolute measure of the local curvature of the response function over the expected distribution of network inputs. As this requires integrating over a fine mesh embedded in the input space, more computation may be required than is practical. Instead, we will follow a more intuitive path. In particular we will modify the standard BEP learning rule in such a way as to $(i)$ reduce the number of hidden units in the network, and $(i)$ minimize the magnitudes of the network weights. As it tends to overconstrain the network, the first objective parallels that of the structural risk minimization, where small networks generalize better [11]. The second objective is designed to avoid unnecessarily abrupt transitions in the response function. This follows from observing that the gradient of the sigmoidal function $f(\vec{w}^t\vec{x} - \theta)$ with respect to the input vector, $\vec{x}$, is proportional to the weight vector, $\vec{w}$. Effectively, it also reduces the degrees of freedom of a network assuming the number of degrees of freedom is proportional to the magnitude of weights.

This intuitive approach was partially inspired by an algorithm designed to reduce the number of weights in a network during the training phase [9]. Here, one auxiliary term, $\sum_i \sigma(w_i)$, was added to the standard BEP error function, the summation being performed over all of the weights and thresholds in the network. The terms of the summation, $\sigma(w) = w^2/(1 + w^2)$, measure the magnitudes of the weights relative to unity. Thus, this summation is a rough measure of the number of "significant" weights in the network; adding it to the error function biases the algorithm towards architectures that use the least number of significant weights. The combined energy function is then minimized by steepest descent. After a certain training criterion is reached, weights with magnitudes falling below a critical threshold can be removed from the network by clamping their values to zero. Although this algorithm reduces the number of weights, it does not effectively reduce the number of hidden units, as architectures with fewer hidden units, but the same number of weights, are not favored. For this reason, we add two terms to the BEP learning rule. The first is designed to remove as many hidden units as possible, while maintaining an acceptable level of error in the response function over the training data. For this to succeed, the units must be operating near their transition regime. The second term is designed to satisfy this requirement by minimizing the magnitudes of the weights.

These modifications to BEP are detailed in Section 2. Two modified versions of this algorithm are also given in the same section, which include (1) a method that first adds units at the hidden layer to build up a crude architecture, then deletes unnecessary units using the original algorithm, and (2) an incremental learning rule which gradually expands the training set as the network learns until it covers all the samples needed to be learned. In Section 3, we present the results of several numerical simulations that demonstrate the effectiveness of the algorithm derived. In the first set of simulations we show that a network, beginning with a large number of hidden units, can be reduced in size to one having a response function that smoothly interpolates between the training data points. In the second set, we construct training data from a network with an "unknown" number of hidden units, and show that the algorithm can be used to infer the architecture of the unknown network with a high probability. In Section 4, two modified versions of the learning algorithm are applied to a real world problem: training a network to control a two-link manipulator to draw characters. Finally, in Section 5, we conclude this chapter by stating some important issues in generalization for learning analog mappings.

## 5.2 The Learning Algorithm

### 5.2.1 The Network Reduction Algorithm

First, we consider the problem of training a feedforward network having a single input unit, one layer of $N$ hidden sigmoidal units, and a single linear output unit, to smoothly interpolate between the $M$ ordered pairs, $\{(x^\pi, y^\pi) : \pi = 1, \ldots, M\}$, of a given training set. (Here, $y^\pi$ is the desired output value when the network input is set to $x^\pi$.)

We assume that the number of hidden units has been initially estimated to be larger than necessary, and the network architecture, which is the number of units at the hidden layer, does not change during learning. Let $w_{1i}$ and $w_{2i}$ denote the input and output weights of the $i$-th hidden unit, and $\theta_i$ its threshold value. The response function of the network then has the form $g(x; \vec{w}, \vec{\theta}) = \sum_{i=1}^{N} w_{2i} f(w_{1i}x - \theta_i)$, where, for notational convenience, we let $\vec{w} = (w_{11}, \ldots, w_{1N}, w_{21}, \ldots, w_{2N})^t$, and $\vec{\theta} = (\theta_1, \ldots, \theta_N)^t$. The sigmoidal function is usually taken to be a modified hyperbolic tangent, i.e, $f(x) = 1/(1 + e^{-x})$. Under BEP, one attempts to find weight and threshold values that minimize the standard error function,

$$\varepsilon_0(\vec{w}, \vec{\theta}) = \sum_{\pi=1}^{M} \left[ g(x^\pi; \vec{w}, \vec{\theta}) - y^\pi \right]^2, \tag{5.1}$$

by gradient descent [10] [12].

For the architecture described above, we define a hidden unit to be *significant* if it is coupled to both the input and output units with weights of a significantly large magnitude, i.e, greater than one. Thus, the quantity,

$$S_i = \sigma(w_{1i})\sigma(w_{2i}), \tag{5.2}$$

can be viewed as a measure of the significance of the $i$-th hidden unit, where, as before, $\sigma(w) = w^2/(1+w^2)$. Following the first objective of the previous section, we desire to favor those architectures that require the fewest number of significant hidden units.

If the given training set does not fully constrain the given network architecture, then there is, in general, a degenerate set of solutions over which $\varepsilon_0(\vec{w}, \vec{\theta})$ is acceptably small. Following our first objective, we add a term proportional to

$$\varepsilon_1(\vec{w}) = \sum_{i=1}^{N} \sum_{j=1}^{i-1} S_i S_j \tag{5.3}$$

to the standard error function. This biases the algorithm toward those solutions that require the fewest number of significant hidden units. From its definition, $\varepsilon_1$ achieves a minimum value of zero if no more than one hidden unit has a non-zero significance, and approaches its upper bound of $N(N-1)/2$ as the magnitudes of all weights increase without bound. After applying the gradient descent algorithm, we obtain the learning rule

$$\begin{aligned} w_{ji}^{n+1} &= w_{ji}^{n} - \eta \frac{\partial \varepsilon_0}{\partial w_{ji}}(\vec{w}^n, \vec{\theta}^n) - \lambda \frac{\partial \varepsilon_1}{\partial w_{ji}}(\vec{w}^n), && (5.4) \\ \theta_i^{n+1} &= \theta_i^{n} - \eta \frac{\partial \varepsilon_0}{\partial \theta_i}(\vec{w}^n, \vec{\theta}^n), && (5.5) \end{aligned}$$

where $\eta$ and $\lambda$ are learning rate parameters, and $j \in [1,2]$. Note that the last term in equation (5.4) couples the dynamics of the weights so that, for example, increasing the significance of the $k$-th hidden unit increases the decay rate of every weight associated with the other hidden units. Also note that this term is proportional to $\sigma'(w_{ji}) = 2w_{ji}/(1+w_{ji}^2)^2$, which becomes insignificant for large enough $\mid w_{ji} \mid$. This will help stabilize the dominant weights, but will also, in part, necessitate the second objective stated in the previous section.

Because of possible conflicts between the two gradients in Equation (5.4), spurious equilibria may exist. It is thus helpful to include the auxiliary term only after the network has learned the training set to a sufficient degree. Consequently, we let $\lambda = \lambda(\varepsilon_0) = \lambda_0 \exp(-\beta\varepsilon_0)$, where $\beta^{-1}$ defines a characteristic standard error: the value of $\varepsilon_0$ below which the auxiliary term comes into play. Note that the resulting learning rule no longer follows the direction of steepest descent of the combined error function; however, the desired objective is ultimately obtained.

We attain our second objective of reducing the weight magnitudes by subtracting an amount proportional to $\tanh(w_i)$ from the right-hand side of Equation (5.4). Although other choices are possible, this one has shown to be effective in practice. Unlike the weight reduction scheme [9] discussed in the introduction, our method preferentially reduces the larger weights in the network. We also apply this term to the threshold's update rule, because, in our examples of interest, we are interested in the region of input space around the origin.

We thus obtain the network reduction algorithm

$$\begin{aligned} w_{ji}^{n+1} &= w_i^n - \eta \frac{\partial \varepsilon_0}{\partial w_{ji}}(\vec{w}^n, \vec{\theta}^n) - \lambda \frac{\partial \varepsilon_1}{\partial w_{ji}}(\vec{w}^n) - \mu \tanh(w_{ji}^n), && (5.6) \\ \theta_i^{n+1} &= \theta_i^n - \eta \frac{\partial \varepsilon_0}{\partial \theta_i}(\vec{w}^n, \vec{\theta}^n) - \mu \tanh(\theta_i^n). && (5.7) \end{aligned}$$

As before, we gate the influence of the new term on how well the network is learning the training set. In this case, it appears helpful to reduce this term gradually with time. In particular, we let $\mu = \mu_0 \mid \varepsilon_0(\vec{w}^n, \vec{\theta}^n) - \varepsilon_0(\vec{w}^{(n-1)}, \vec{\theta}^{(n-1)}) \mid$.

Once an acceptable level of performance is reached, any weight with magnitude below a certain level is removed from the network. When a hidden unit is connected to the rest of the network with only "removed" weights, then the unit is discarded. Thus, as is desired, the number of hidden units is reduced.

Finally, we mention that this algorithm can be extended to other architectures.

For example, for a network having $K$ inputs and $L$ outputs, one may let

$$S_i = \sum_{k=1}^{K}\sum_{l=1}^{L}\sigma(w_{1ki})\sigma(w_{1il}), \tag{5.8}$$

where, $w_{1ki}$ is the value of the weight connecting the $k$-th input to the $i$-th hidden unit, and $w_{2il}$, is that of the weight connecting the $i$-th hidden unit with the $l$-th output.

### 5.2.2 Modification 1 (Algo-a1) of the Network Reduction Algorithm: Adding Units

One of the disadvantages of the network reduction algorithm is that the tuning of the weights takes a long time, especially when the initial network is much larger than necessary. Therefore, it is natural to first find a crude architecture by gradually adding neurons at the second layer, then using the network reduction algorithm to delete unnecessary units. Specifically, the modified algorithm which incorporates the adding rule can be described as follows.

a). An initial network, which has for instance one unit at the hidden layer, is chosen, and trained until a local minimum is reached.

b). A new hidden unit is added and trained while the old one is frozen.

c) The old unit is released and trained simultaneously with the new one until a new local minimum is reached.

This adding process continues until the mean square error on the training set reaches a desired value. Then the network is tuned to a possibly smaller size using the network reduction algorithm. In Section 4 this modified version of the algorithm is applied to training a network to control a robot arm to draw characters, and demonstrate that it is more efficient in finding networks that have better generalization than the networks obtained using a fixed architecture.

## Modification 2 (Algo-i2) of the Network Reduction Algorithm: Incremental Learning

One of the difficulties in learning is that training gets longer and longer when a problem becomes more and more complex. For example, when a network is trained to control a robot arm traveling along a single trajectory, training may still be manageable. However, if the training needs to be done in the whole input space, the training task may be too complicated and time-consuming to tackle. One way to deal with this problem is to partition the original task into manageable sub-tasks. If learning one sub-task helps to learn the others, the overall task can be gradually accomplished by training sub-networks one at a time for the sub-tasks and frequently reviewing what has been learned. Based on this idea, a learning scheme called incremental learning is derived as follows.

Let $\{X_1, ..., X_k\}$ be a training set which contains $k$ subset s $X_1$, ..., and $X_k$.

*a*) The algorithm Algo-a1 is used to learn the first subset $X_1$ until a desired performance is achieved and the first subnetwork $N_1$ is obtained.

*b*) $N_1$ is frozen and the step *a*) is repeated on the second sub-training set $X_2$ until the second sub-network $N_2$ is obtained.

*c*) $N_1$ and $N_2$ are trained simultaneously using the original network reduction algorithm on $X_1$ and $X_2$ to prevent the network from forgetting $X_1$ and enhance the learning the union of the two sub-training sets.

The procedure is repeated until the whole training set $X$ is learned and the size of the overall network is small.

Whether this method facilitates the learning of a complicated task usually depends on the nature of the problem and the way of partitioning it. In Section 4, we will show that it is feasible to use incremental learning to train a network to draw characters anywhere horizontally in the space.

## 5.3 Simulation Results for the Network Reduction Algorithm

### 5.3.1 Learning a Smooth Function

In the first simulation, the network reduction algorithm is used to reduce the number of hidden units and results in a smooth response function. The training set of the first run consists of 9 equally spaced data points taken from the graph of the function. $\phi(x) = e^{-(x-1)^2} + e^{-(x+1)^2}$ over the domain $[-2, 2]$. We begin with the network described in Section 2.1, with $N = 20$; the 40 weights and 20 thresholds are randomly initialized from a uniform distribution over the interval $[-25, 25]$. With learning parameters set to $\eta \approx 5 \times 10^{-3}$, $\beta = 0.1$, $\lambda_0 = 6.5 \times 10^{-3}$, and $\mu_0 = 5 \times 10^{-4}$, the network is trained by applying the learning rules in equation (5.6) and (5.7). After the value of $\varepsilon_0$ falls below the value 0.05, any weight with a magnitude less than 0.1 is set to zero. The resulting network uses only 5 of the 20 available hidden units. At this point, the nominal increase in $\varepsilon_0$ resulting from eliminating the weaker weights is corrected by training the reduced network with the standard BEP algorithm for a few additional iterations.

In Figure 5.1, the response function of the network obtained by this procedure is compared against one obtained by BEP (i.e, $\lambda_0 = \mu_0 = 0$) with the same initial conditions. Note that the response function obtained by the network reduction algorithm smoothly interpolates between the 9 training points and possesses the same number of local extrema as $\phi(x)$. This cannot be said for the response function generated by BEP, which oscillates wildly with minimum and maximum values $-17.5$ and $10.9$ over the input domain, $[-3,3]$.

A more quantitative comparison can be made by averaging the root-mean-square (RMS) deviation between each response function and $\phi(x)$ over the interval $[-2,2]$, i.e,

$$\epsilon_{RMS} = \frac{1}{2}\{\int_{-2}^{2}[g(x;\vec{w},\vec{\theta}) - \phi(x)]^2, dx\}^{1/2}.$$

(Note that $\epsilon_{RMS}$ is a random variable, as the particular determination of the response function, $g$, depends on the initial values of the weights and thresholds, which are set at random.) For this instance of the network reduction algorithm, we obtain $\epsilon_{RMS} = 1.15 \times 10^{-2}$, while for BEP, $\epsilon_{RMS} = 1.71$. In the figure, the solid curve indicates the output response of the network with 1 input, 20 hidden units, and 1 output, trained by the network reduction algorithm on the 9 training points indicated by circular markers. The broken curve indicates a response function obtained by training the same network with the same data using BEP. The dotted curve indicates the graph of $\phi(x)$ from which the training points were selected

If the algorithm does yield a network that generalizes well, then the size of the network should not depend critically on the number of training samples used. This necessarily assumes that the data in each set faithfully represents the significant features of the training problem. Therefore, in a second run, the network is trained with the same initial state, but with a training set containing 17 equally spaced samples taken from the graph of the same function. Again the algorithm reduces the network to 5 significant hidden units, with $\epsilon_{RMS} = 8.81 \times 10^{-3}$. Training the network under BEP with this data yields a network with $\epsilon_{RMS} = 4.73 \times 10^{-2}$. The response functions of these two networks are displayed in Figure 5.2. At the left end of the

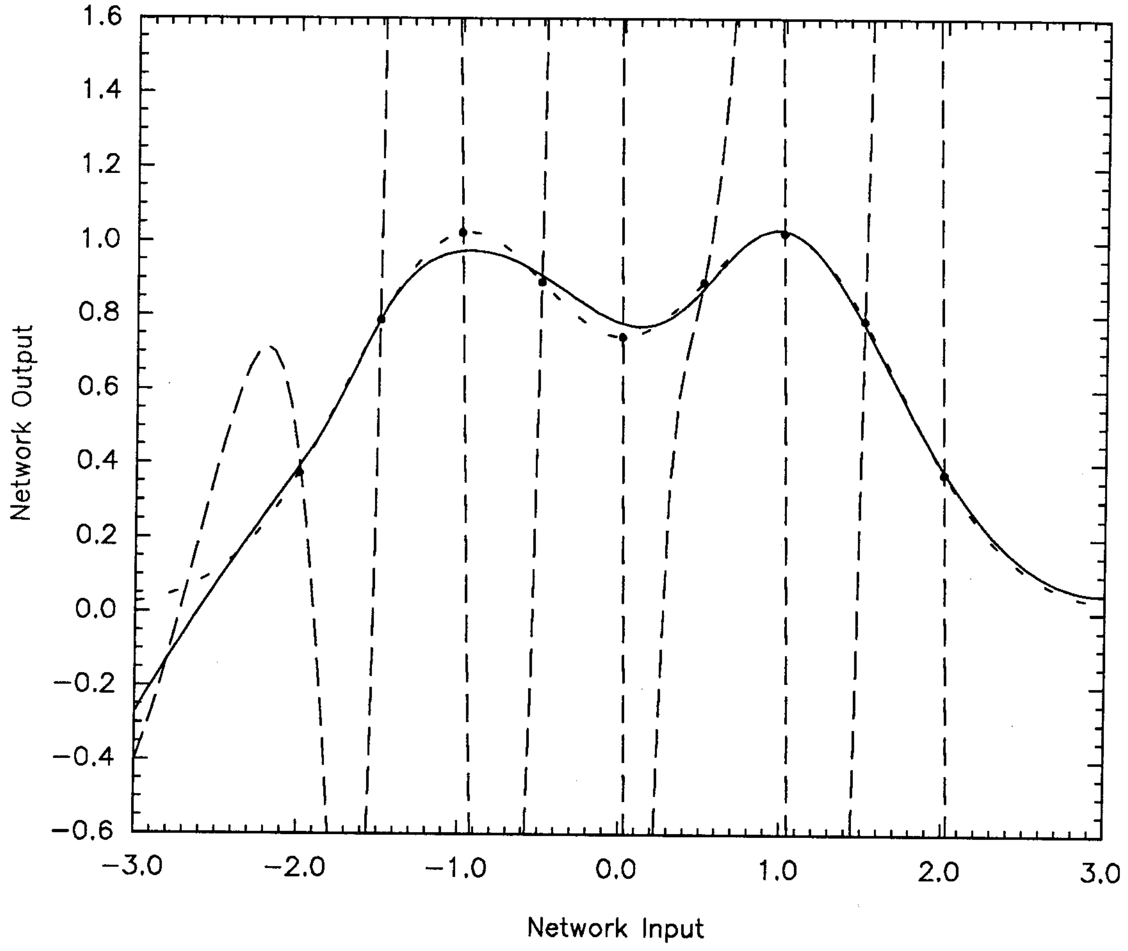


Figure 5.1: Learning a smooth function with 9 samples

| Hidden Units ($n$) | Frequency ($\nu_n$) | E($\epsilon_{RMS} \mid n$) |
|---|---|---|
| 2 | 22 | 0.0275 |
| 3 | 9 | 0.0263 |
| 4 | 6 | 0.0405 |
| 5 | 6 | 0.0401 |
| 6 | 3 | 0.0656 |
| 7 | 4 | 0.0700 |

Table 5.1: Simulation results for the "inverse network" problem. Out of a total of 50 runs, the center column shows the number of times a network with $n$ significant hidden units was obtained. The right column equals the average of $\epsilon_{RMS}$ — computed over the interval $[-2, 2]$ — over the $\nu_n$ runs ending with $n$ significant hidden units.

displayed interval, the network response obtained from BEP (the broken curve) drops off scale to $-5$. At the right end, it quickly climbs to 20, and then saturates.

### 5.3.2 Learning a Target Network

Next, we explore an "inverse network" problem: to what extent can one use this algorithm to infer the architecture of a feedforward neural network from only a finite sample of its response function? A network containing one input, a layer of two hidden units, and a single linear output is chosen, and a training set of 17 sample points from its response function is generated. Then, an ensemble of 50 new networks, each containing 10 hidden units, is trained on the data set with the network reduction algorithm. The results of these 50 simulations are summarized in Table 5.1.

Note that twenty-two times out of fifty the algorithm finds a network of minimal size. It is encouraging that the response functions with the least average RMS error,

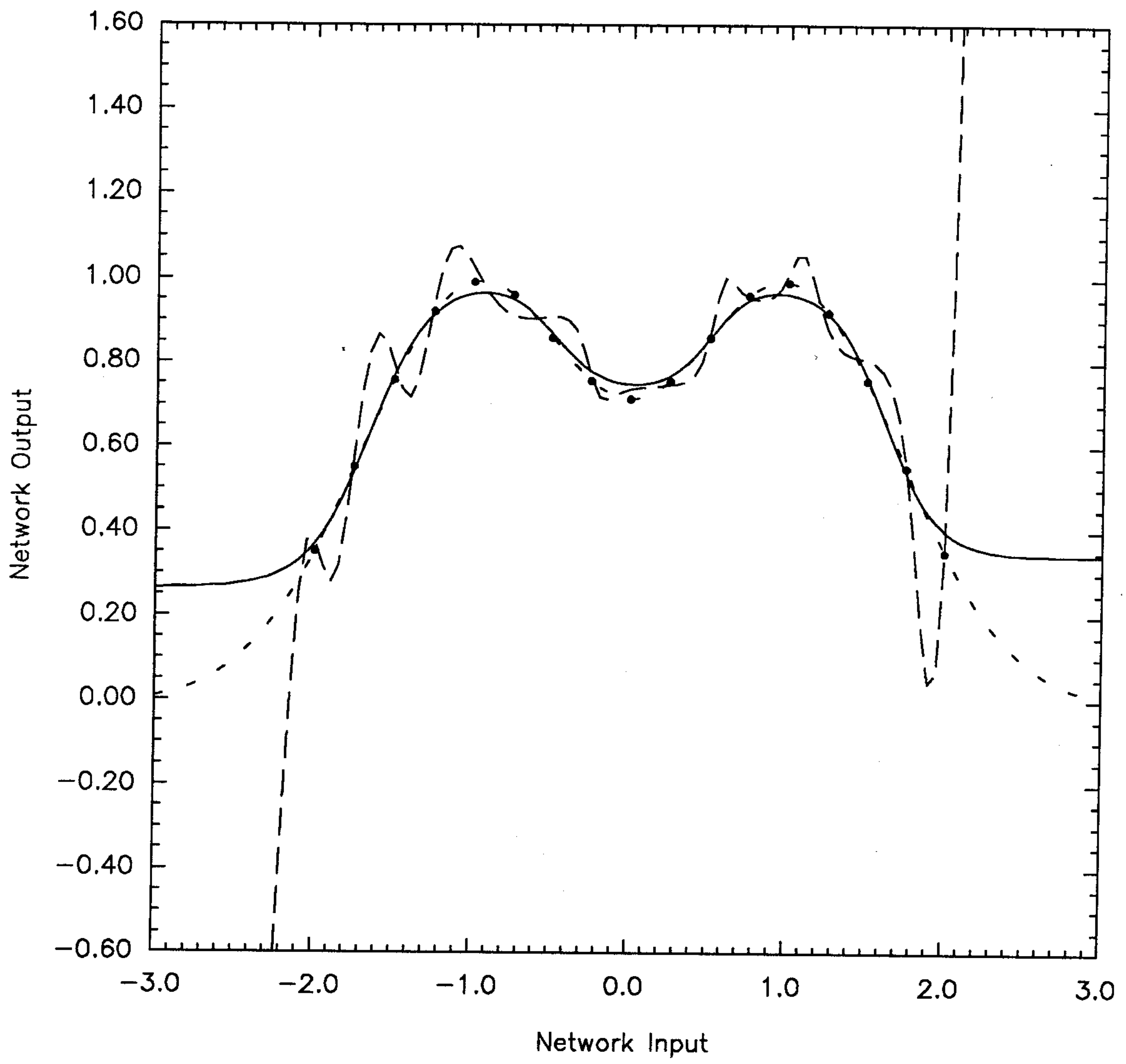


Figure 5.2: Learning a smooth function with 17 samples

measured over the interval $[-2, 2]$, come from networks having two or three hidden neurons. In Figure 5.3, we show how the response functions of the network regress towards that of the "concealed" network as the number of hidden units used decreases. Response functions obtained for the "inverse network" problem are displayed as solid curves, that of the "concealed" network, as a dotted curve, and the training points as circular markers. Graphs (a) – (c) reflect the median outcomes — the networks resulting in the median values of $\epsilon_{RMS}$ — for the sets of trials resulting in $n = 2$, 4, and 7 significant hidden units, respectively. Graph (d) reflects the median outcome of ten trials using BEP, all of which resulted in 10 significant hidden units. Values of $\epsilon_{RMS}$ were computed over the input interval $[-2, 2]$; for the response functions displayed in graphs (a) – (d), $\epsilon_{RMS}$ equals $2.87 \times 10^{-2}$, $3.35 \times 10^{-2}$, $4.49 \times 10^{-2}$, and 0.291 respectively. For comparison, ten networks trained by BEP alone yielded an average RMS error of 0.461 without any apparent reduction in the number of significant hidden units.

### 5.3.3 Discussions

In the above we have shown that by adding suitably chosen terms to the BEP learning rule, desirable global properties in the network's response function can be obtained. In particular, the BEP algorithm was tailored to prefer networks having smoother response functions. From the simulations, it is apparent that this behavior is attained at the cost of a slower convergence rate. In the first simulation, where $\phi(x)$ was approximated using a nine-point training set, $50,000$ iterations were required by the network reduction algorithm, but only 300 were required by BEP. This discrepancy was however reduced when the networks were trained from the 17 point set: the network reduction algorithm needed $650,000$ iterations, and BEP 150,000. Approximately one-half of the fifty "inverse network" simulations required more than $800,000$

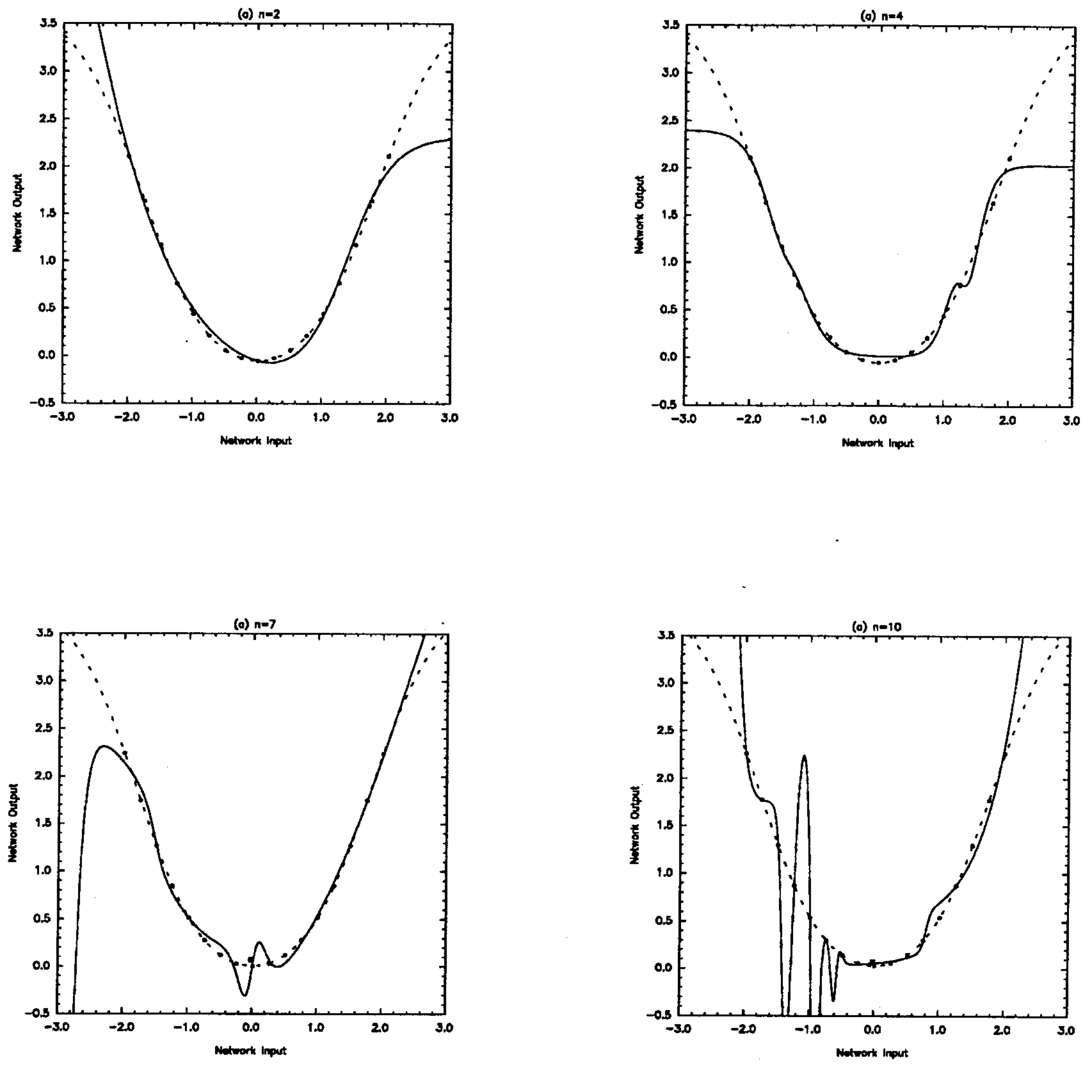


Figure 5.3: Learning the target network.

iterations, which is larger than the 400,000 iterations typically needed to train a network using BEP with ten hidden units, with the same 17 samples drawn from the "concealed" network. It should be noted that in the above examples we choose initial weights with fairly large magnitudes over the interval $[-25, 25]$ to ensure that the number of degrees of freedom of the networks approximately equals the number of weights. When the weights are initially small, that is, the effective degrees of freedom of a network are smaller than the total number of weights, the convergence rate can be greatly accelerated and the resulting network may generalize well to a certain degree. However, in the examples given in Section 4, we will show that small networks, which are capable of learning the training set, usually have better performance in generalization than large networks in learning analog mappings.

## 5.4 Application of the Algorithm in Control

In this section two modified versions of the original network reduction algorithm are applied to train a network to control a two-link manipulator drawing characters. Through this application, properties of small and larger networks are further investigated to gain a better understanding of generalization for analog mappings. The incremental learning rule is also applied which shows that it does facilitate the training task for the problem we consider.

Most of the simulations described in this chapter are done using the feedforward learning mechanism. Details on the feedback learning scheme, which is more practical in real situations, can be found in [3].

### 5.4.1 Feedforward Learning Mechanism

The feedforward learning mechanism is shown in Figure 5.4. For this case, the

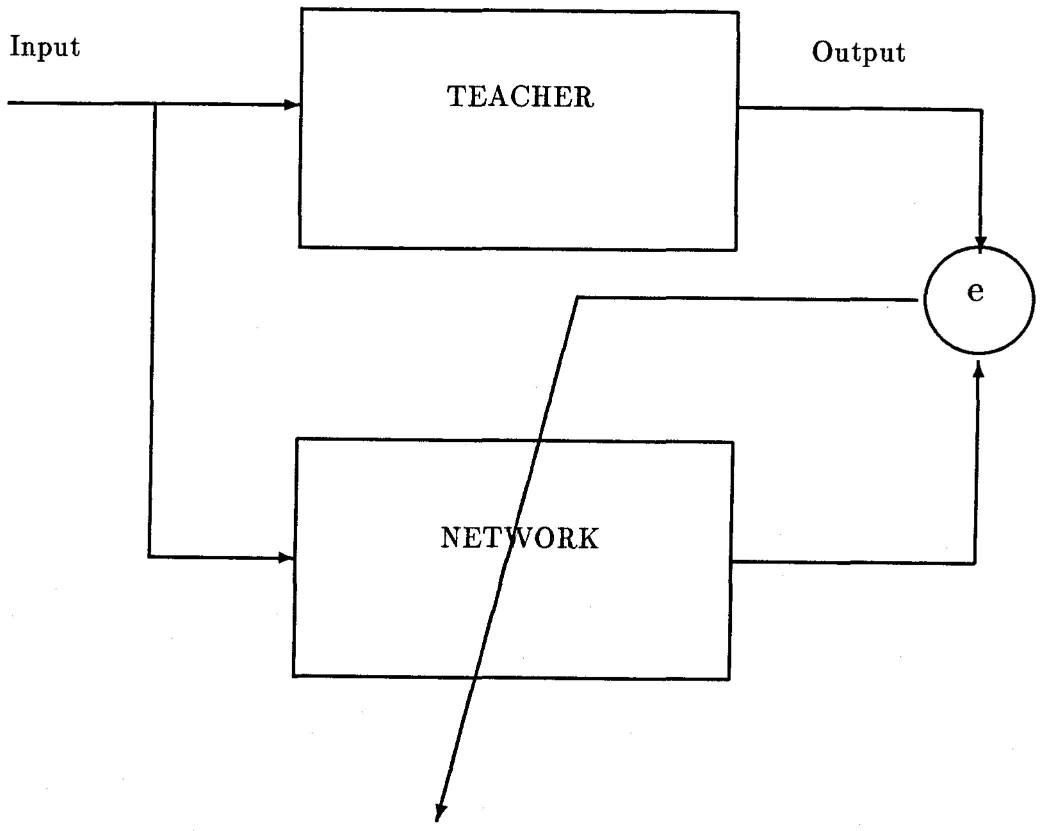


Figure 5.4: Feedforward Training Mechanism.

training is identified as the off-line training compared with the feedback learning system that does training on-line. In order to do off-line training, it is assumed that some desired input-output pairs of a "plant" are known and form a stationary training set. Training is accomplished when the mean square error on the training set is sufficiently small. Although this type of feedforward learning system is not feasible in real situations due to the lack of desired outputs of the unknown plant, it can be used when some a priori knowledge is available and used as a crude off-line training set before more accurate training- feedback learning takes place. Besides, it is easier to investigate generalizing capability of networks using such a simple system, since the ability of the resulting network in generalization can be identified as to how well it learns the analog mapping characterized by the discrete training samples.

**a) A Two-Link Manipulator**

In this application, the plant is a two link manipulator shown in Figure 5.5. The Newton-Euler equation [5] that describes its dynamics is given in the following expressions:

$$\underline{\tau} = M(\underline{\theta})\ddot{\underline{\theta}} + \underline{v}(\underline{\theta}, \dot{\underline{\theta}}) + \underline{g}(\underline{\theta}),$$

$$M(\underline{\theta}) = \begin{pmatrix} 2m_2l_1l_2\cos(\theta_2) + m_1l_1^2 + m_2l_1^2 + m_2l_2^2 & m_2l_1l_2\cos(\theta_2) + m_2l_2^2 \\ m_2l_1l_2\cos(\theta_2) + m_2l_2^2 & m_2l_2^2 \end{pmatrix},$$

$$\underline{v}(\underline{\theta}, \dot{\underline{\theta}}) = \begin{pmatrix} -m_2l_1l_2\sin(\theta_2)\dot{\theta}_2^{\,2} - 2m_2l_1l_2\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 \\ m_2l_1l_2\sin(\theta_2)\dot{\theta}_1^{\,2} \end{pmatrix},$$

$$\underline{g}(\underline{\theta}) = \begin{pmatrix} m_2l_2g\cos(\theta_1 + \theta_2) + m_1l_1g\cos(\theta_1) + m_2l_1g\cos(\theta_1) \\ m_2l_2g\cos(\theta_1 + \theta_2) \end{pmatrix},$$

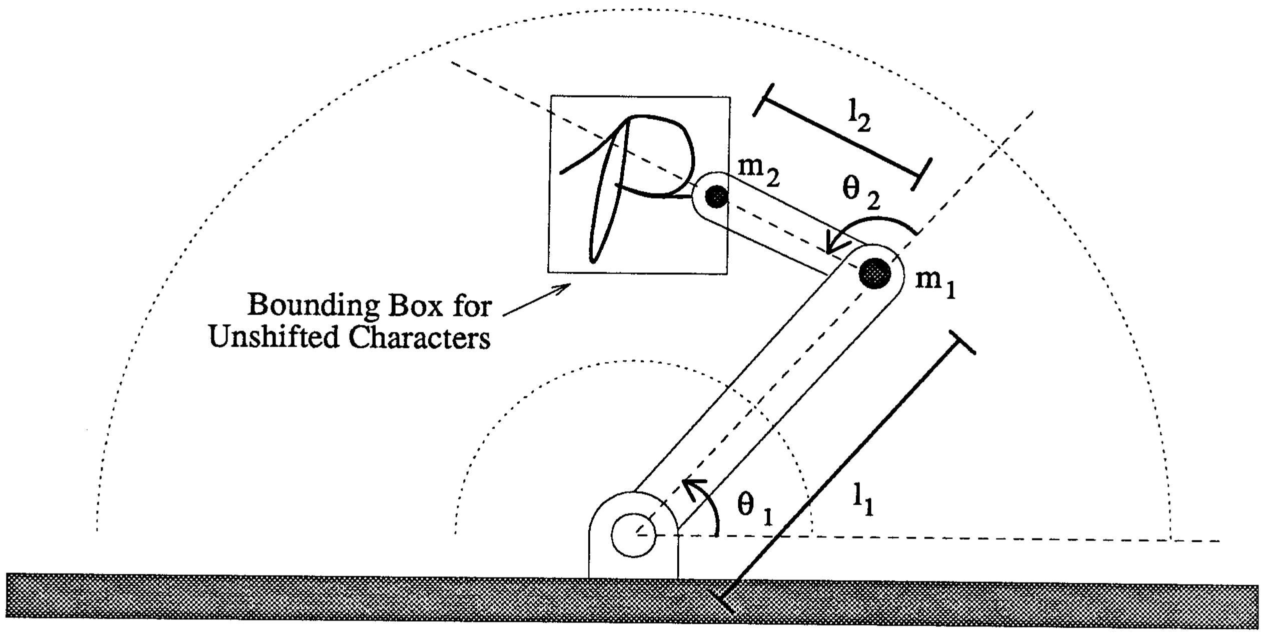


Figure 5.5: Two-Link Manipulator

where $M$ is the mass matrix, $\underline{v}$ is the vector of centrifugal and Coriolis terms, and $\underline{g}$ is the vector of gravitational terms; $\underline{\theta}^t = (\theta_1, \theta_2)$, and $\underline{\tau}^t = (\tau_1, \tau_2)$. The equations are used to simulate the plant in our computer experiments with the following parameters: $m_1 = 2$kg, $l_1 = 2$m, $m_2 = 1$ kg, $l_2 = 1$m, and $g = 10$mm$/s^2$.

When a network is trained to approximate an inverse plant, a set of $M$ desired inputs $\underline{\theta}$, $\underline{\dot{\theta}}$, $\underline{\ddot{\theta}}$ and outputs $\underline{\tau}$ is used, which are obtained using the Newton-Euler equation with time step $\Delta_t = 0.1$. Specifically, in our simulations, those training samples are associated with a particular trajectory for drawing a character.

### b) Feedback Control Mechanism

Once the off-line training is done, the resulting network is connected in the feedback control system in Figure 5.6 to test whether desired characters can indeed be

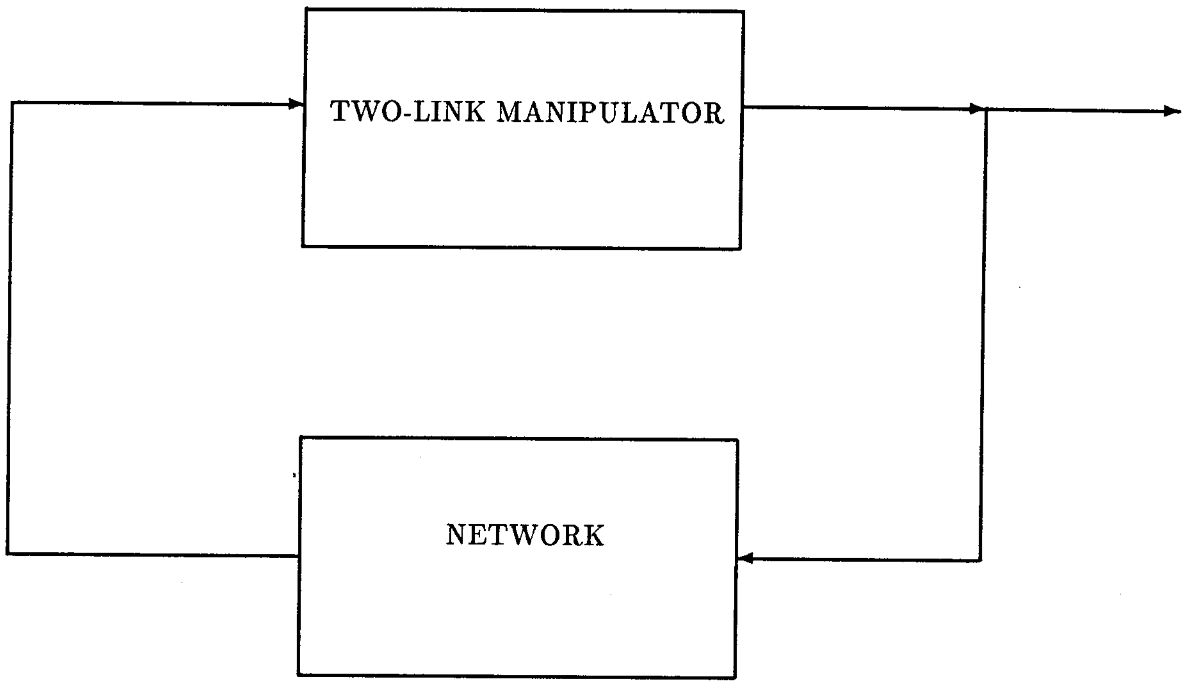


Figure 5.6: Feedback control mechanism for testing.

drawn by the robot arm. The feedback loop is necessary for correcting numerical errors.

### 5.4.2 Drawing Charaters Within the Window

We first train a network for drawing a letter "$P$" inside the square window shown in Figure 5.5. The training set contains a 100 input-output pairs which are the given $\underline{\theta}$, $\dot{\underline{\theta}}$, $\ddot{\underline{\theta}}$ and the desired $\underline{\tau}$. The details for obtaining the training samples from a character drawn on the screen are described in [2].

The goal for this experiment is two-fold: 1) to find out how each individual resulting network generalizes when drawing other letters on which the network has not been trained, as well as translated versions of the trained $P$; 2) to study how the performance of networks in generalization varies in terms of their sizes.

The algorithm Algo-a1 is first used to find appropriate number of hidden units for this problem. Networks with 3 hidden units are obtained 8 times in 10 simulations using different random initial conditions. Moreover, networks of fixed structures with different hidden units were also trained 10 times using BEP, each starting with different random initial weights. All these networks are then tested on drawing different letters. Figure 5.7 gives the letters drawn by one of the resulting networks trained with the algorithm Algo-a1 and training letter $P$, where the dotted and the solid characters are drawn by the teacher and by the network respectively. The character $P$ is the trained pattern, and the rest of characters are untrained.

Table 5.2 summarizes the performance of all these networks. The second column in the table indicates sizes of networks as well as the training method. There are 10 networks for each kind trained with BEP, and 8 networks obtained using the Algo-a1. Number of Failures indicates the number of networks in each class which fail to draw the character. MSE is the mean square error between a desired trajectory and an actual trajectory averaged over those networks which succeed in drawing the character. "*" indicates that it is impossible to compute the MSE due to too many failures.

The results given in this table indicates that the small networks obtained using our algorithm and the networks with only a few hidden units trained with BEP can all generalize well in terms of drawing not only the trained letter $P$ but also other letters which they have never seen before. The large networks (the networks with 20 hidden units), although they have learned the discrete training samples, they fail to generalize in terms of drawing the whole letter $P$ as well as the other letters.

To explore further the maximum generalization achievable by the networks using our algorithm and how the size of a network affects its performance in generalization,

| Character Drawn | Hidden Units (Training Method) | Number of Failures | MSE |
|---|---|---|---|
| P(trained) | 3 (Algo-a1) | 0 | .005 |
| | 2 (BEP) | 0 | .004 |
| | 6 (BEP) | 2 | .011 |
| | 10 (BEP) | 0 | .046 |
| | 20 (BEP) | 9 | * |
| M | 3 (Algo-a1) | 0 | .005 |
| | 2 (BEP) | 0 | .003 |
| | 6 (BEP) | 2 | .010 |
| | 10 (BEP) | 0 | .009 |
| | 20 (BEP) | 7 | * |
| Circle | 3 (Algo-a1) | 0 | .004 |
| | 2 (BEP) | 0 | .002 |
| | 6 (BEP) | 0 | .010 |
| | 10 (BEP) | 1 | .011 |
| | 20 (BEP) | 7 | * |
| Line | 3 (Algo-a1) | 0 | .004 |
| | 2 (BEP) | 0 | .003 |
| | 6 (BEP) | 2 | .012 |
| | 10 (BEP) | 1 | .013 |
| | 20 (BEP) | 6 | * |

Table 5.2: Simulation and test results for drawing characters, trained and untrained.

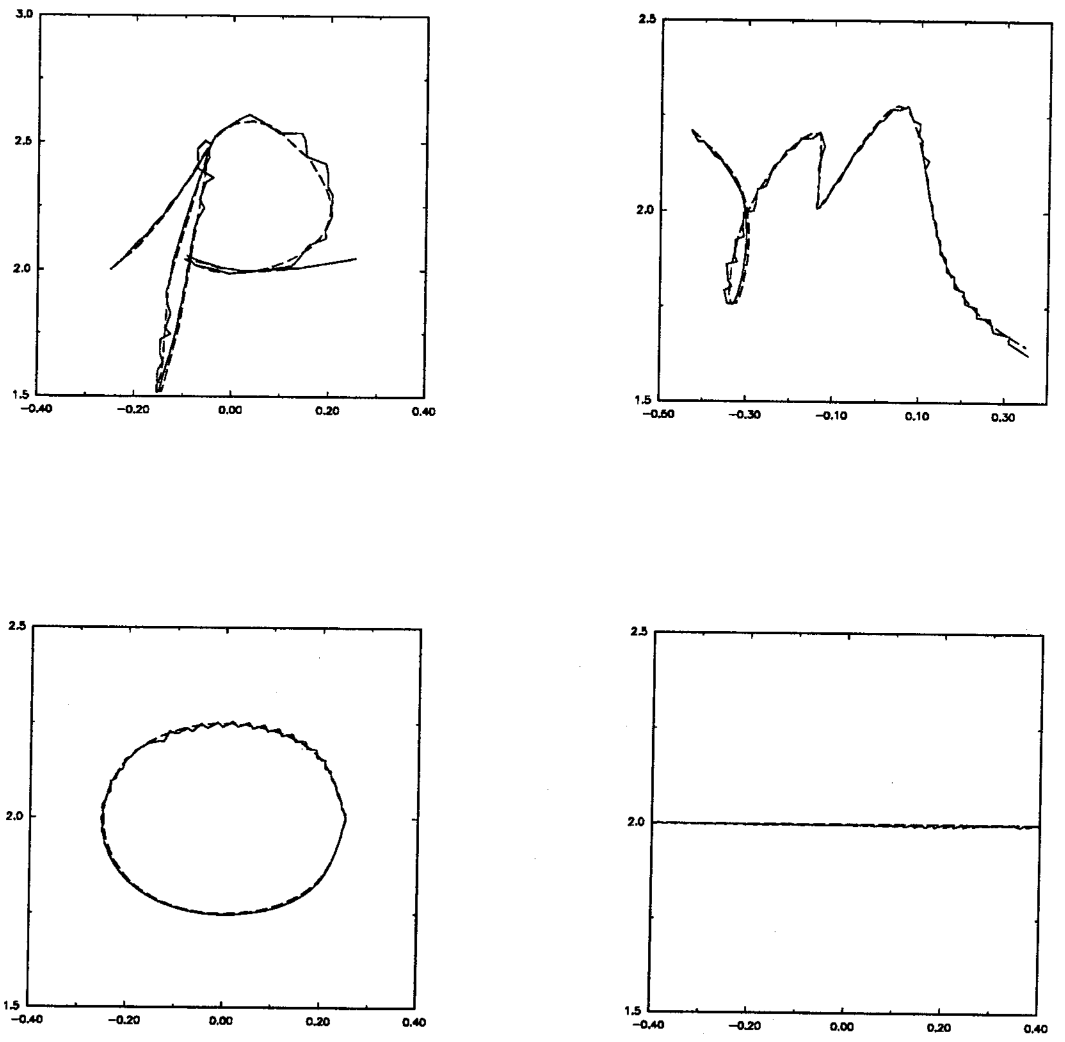


Figure 5.7: Characters drawn by a network trained with the algorithm Algo-a1.

the networks are tested on shifted, rotated and scaled versions of the trained $P$. Results are given in Table 5.3, where dx and dy are the shifts in x and y respectively; $\theta$ is the rotation angle, and "scale" is the scaling factor. The rest of the notation is the same as in Table 5.2. These results demonstrate that small networks generalize much better than large networks in drawing the translated versions of the trained $P$.

Since it is still hard to discriminate, through the results in Table 5.3, between the performance of the networks obtained through our algorithm and small networks (with 2 hidden units, for instance), trained with BEP, more rotated versions were tested on these two types of networks, and the results shown in Table 5.4.

These results suggest that although both types of networks are pretty small, the networks obtained through our algorithm have better tolerance to the rotation of the trained pattern than the networks trained with a fixed architecture using BEP.

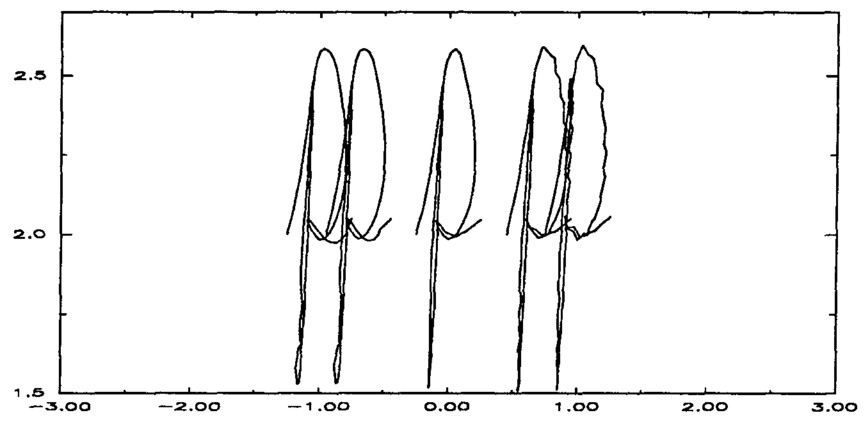
### 5.4.3 Drawing Characters Outside the Window

Although training a network to draw a letter within the window is tractable, training a network to draw the same letter with shifts anywhere in a much larger window can be a formidable task. However, the results in Table 5.3 indicates that training a network with a letter at one position may help to learn its shifted version since the network does generalize to some degree to the shifted letters . Therefore, the incremental learning technique (Algo-i2) is used to train a network drawing the letter $P$ with any horizontal shift in the region.

Specifically, our overall training set consists of 5 subsets of the training samples with 100 each drawing according to the trajectories corresponding to 5 horizontally shifted $P$'s. The first subnetwork is obtained by the algorithm Algo-a1 on the first subset corresponding to the $P$ without any shift. Then the other 4 letters are learned

| dx | dy | $\theta$ (degree) | Scale | Hidden Units (Training Method) | Number of Failure | MSE |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | .9 | 3 (Algo-a1) | 1 | .009 |
| | | | | 2 (BEP) | 0 | .006 |
| | | | | 3 (BEP) | 4 | .009 |
| | | | | 6 (BEP) | 7 | .011 |
| | | | | 10 (BEP) | 5 | .046 |
| | | | | 20 (BEP) | 10 | * |
| 0 | 0 | 10.0 | 1.0 | 3( Algo-a1) | 0 | 0.007 |
| | | | | 2 (BEP) | 3 | 0.050 |
| | | | | 3 (BEP) | 3 | 0.164 |
| | | | | 6 (BEP) | 5 | 0.155 |
| | | | | 10 (BEP) | 5 | 0.215 |
| -.3 | -.1 | 0 | 1.0 | 3 (Algo-a1) | 1 | .007 |
| | | | | 2 (BEP) | 0 | .008 |
| | | | | 3 (BEP) | 1 | .021 |
| | | | | 6 (BEP) | 2 | .068 |
| | | | | 10 (BEP) | 3 | .073 |
| | | | | 20 (BEP) | 10 | * |

Table 5.3: Test results for the shifted, rotated and scaled $P$.



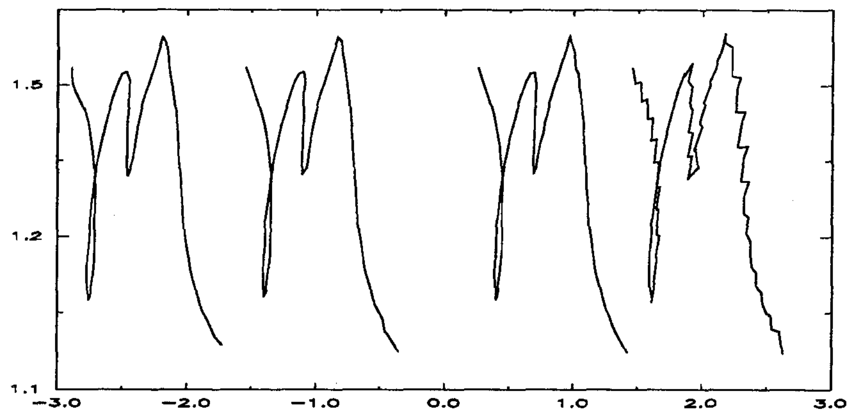


Figure 5.8: Shifted $P$'s (training letters) drawn by the teacher. Shifted $M$'s drawn by the resulting network trained on the shifted $P$'s

| $\theta$ (degree) | Hidden Units (Training Method) | Number of Failures | MSE |
|---|---|---|---|
| 5.0 | 3 (Algo-a1) | 0 | .006 |
| | 2 (BEP) | 0 | .006 |
| 8.0 | 3 (Algo-a1) | 0 | .00608 |
| | 2 (BEP) | 1 | .031 |
| 10 | 3 (Algo-a1) | 0 | .009 |
| | 2 (BEP) | 3 | .050 |
| 12 | 3 (Algo-a1) | 2 | .061 |
| | 2 (BEP) | 3 | .075 |
| 15 | 3 (Algo-a1) | 4 | .064 |
| | 2 (BEP) | 10 | * |

Table 5.4: Simulation results for the rotated $P$

using the incremental learning rule. The resulting network, which ends up with 8 hidden units, was tested on drawing 5 letter $M$'s with different horizontal shifts from the trained $P$. Figure 5.8 gives the shifted $P$'s trained and the shifted $M$'s tested drawn by the network.

The algorithm Algo-a1 was also used on the entire training set with 500 samples, but the training was unable to finish within tolerable time. However, more simulations are needed to make careful comparisons between incremental learning and conventional learning as to learning time, performance of generalization and complexity of networks.

## 5.5 Conclusion

In this chapter, we have developed a learning algorithm which implements the smoothness constraint by minimizing the degrees of freedom of a network as well as the magnitude of weights. We have applied this algorithm as well as modified versions to learning analog mappings.

We have found that for learning analog mappings, if a training set contains sufficient information for the problem, a as small as possible network obtained from our algorithm usually has a superior performance in generalization over large networks as well as over other small networks with a similar number of hidden units but trained with BEP.

One of the disadvantages of this algorithm, however, is that the parameters in the algorithm for the penalty terms affect the size of a resulting network and have to be chosen empirically. This is also an intrinsic disadvantage of any similar optimization approach using regularizers. Besides, the training time is somewhat long, especially when an initial network is started out much larger than necessary. Adding hidden units speeds up the training significantly. But since the units can only be added at the second layer, a more general architecture, such as a three-layer architecture and local connectivity patterns, are unable to be explored in this same way in this same way.

# Bibliography

[1] Y. Abu-Mostafa, "Learning From Hints in Neural Networks," *J. Complex.* Vol. 6, 192-198, 1990.

[2] A. Yamamura, *Ph.D. Thesis, Caltech*, 1991.

[3] A.A . Yamamura, C. Ji and D. Psaltis, " An Algorithm for Training A Two-Link Manipulator to Draw Characters," submitted to *Neural Computation.*

[4] A. Barron, " Approximation and Estimation Bounds for Artificial Neural Networks," *Proc. of The 4th Workshop on Computational Learning Theory*, 243-249, 1991.

[5] J.J . Craig, *Introdction to Robotics: Mechanics & Control*, Addison-Wesley, 1986.

[6] D. Marr, *Vision*, W.H . Freeman, San Fransisco: 1981.

[7] T. Poggio and C. Koch, "Ill-posed problems in early vision: from computational theory to analogue networks," *Proc. R. Soc. Lond. B* **226**, 303-323, 1985.

[8] D.E . Rumelhart, presentation given at Hewelett-Packard, Seminar on Neural Networks, 1988.

[9] D.E . Rumelhart, personal communication, 1989.

[10] D. E . Rumelhart, G. E . Hinton, and R. J . Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, ed., M. I. T. Press, (Cambridge, MA: 1986), 318-364.

[11] V.N . Vapnik and A.Y . Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Dokl. Acad. Nauk SSSR,* **181**, 4 (1968), 264-280, English translation in *Theory of Probability and its Applications* **XVI**, 1971,

[12] P. Werbos, *Ph. D. Thesis, Harvard University*, 1974.

# Chapter 6

# Network Addition-Deletion Algorithm

## 6.1 Introduction

In this chapter, a learning algorithm is developed that allows us to find a suitable network structure in a general fashion in terms of number of layers, number of units in each layer and possible local connectivity patterns. The learning algorithm is then applied to investigate how the size of a network affects its performance in generalization for binary mappings only.

There are two questions often encountered in training of feedforward multilayer neural networks for generalization.

1) How many degrees of freedom does a network need to absorb the information in the training set?

2) How can one find a network architecture (in terms of connectivity patterns,

neurons in each layer and number of layers) which is appropriate for the problem when no information other than the training set is available for choosing the structure?

For binary mappings, the VC-inequality[10] provides a guideline for the first question. That is, generalization is achieved (the true error rate is bounded by twice the training error rate $\epsilon$) when the relationship $M \sim O(\frac{d}{\epsilon})$ is satisfied, where $M$ is the number of training samples and $d$ is the VC-dimension which is directly related to the degrees of freedom in the network. Roughly speaking, for a fully trained feedforward multilayer neural network, therefore $W$, which is the number of independently modifiable weights, can be used as a crude measure for the VC-dimension. Then $M \sim O(\frac{W}{\epsilon})$ provides a measure for generalization. According to this theory, small networks ($W$ small) generalize better than large networks.

Several learning algorithms have been developed to address the second question. Roughly they can be divided into two categories: structure minimization through weight deletion and structure growth by adding resources. Most of the algorithms that use weight deletions are done through solving a regularization problem. That is, a network with a fixed structure is used and an extra term is added as a regularizer to the original energy function in a setting of supervised learning. Such a term usually measures the complexity of the network [7] [2] and constrains the network to learn the training samples using as few degrees of freedom as possible. Although these methods have been shown to be capable of finding networks which generalize well, they require either a very slow tuning of weights from a fixed network architecture or a lot of prior knowledge in the network design. On the other hand, learning algorithms for network structure growth [5] [6] [14] add neurons and layers during the learning phase, and this does not require a fixed structure to be prespecified. However, since these algorithms usually control additions of resources by trying to learn the training set as well as possible, overfitting may occur instead of generalization, in fact, the opposite of generalization.

The learning algorithm we present in this chapter combines both structure minimization and growth and tries to circumvent their individual drawbacks. Specifically, the algorithm has three salient features.

First, it tries to find a network as small as possible to learn the training samples.

Second, an architecture of such a small network is searched for through two learning phases: an adding phase and a deleting phase, while the network is learning the given input-output associations using gradient descent. The adding phase builds up a crude structure by filling the connections and neurons at a large "virtual" multilayer network with a maximum $L$ layers possible. The deleting phase refines the structure by trimming off unnecessary connections. The adding and deleting are done based on probability rules and sensitivity measures such that only those connections which are most effective in reducing the error are preserved. This method provides an effective way to search for network structures in a general framework without specifying a fixed structure a priori.

Third, two criteria are used for two different versions of the algorithm to control the switching between the two learning phases and the termination of learning. The first criterioncriterion, drawn from the VC-inequality, is $M \sim O(\frac{W}{\epsilon})$, where $M$ is the total number of training samples, $W$ the number of weights of the network and $\epsilon$ the training error rate. This criterion can estimate generalization performance of the network without a test set. The second criterion is based on the generalization error evaluated on line through a validation set. That is, the error on the validation set is evaluated during the training process to provide an estimator for the true generalization error. Since such an estimator is distribution-dependent, it is more accurate than the VC criterion which gives a worst case bound. The chapter is organized as follows. Section 2 gives the definitions for the sensitivity measures used in addition and deletion. Section 3 explains two versions of the algorithm and presents simulation results to show their effectiveness. In Section 4, the performance of large and small networks on generalization is further investigated through an example. The conclusion will summarize the results and state some related issues.

## 6.2 Learning Algorithm

Let us choose $L$ to be the maximum number of layers for a virtual feedforward multilayer networks with $n(l)$ units at layer $l$ , where $0 \leq l \leq L$. Here $n(0)$ and $n(L)$ are fixed and the remaining $n(l)$ are adaptable. The transfer function of all the neurons is the sigmoid function $f(x) = tanh(x)$.

Let the energy function used for training be the usual quadratic function, i.e.,

$$E = \sum_i || \vec{y_i} - \vec{t_i} ||^2,$$

where the $\vec{y_i}$'s and $\vec{t_i}$'s denote the $i$th actual and desired outputs of the network respectively. The modification of the weights is done using gradient descent, e.g.,

$$\Delta w(i,j,l) \propto -\frac{\partial E}{\partial w(i,j,l)},$$

where $w(i,j,l)$ denotes the connection from unit $i$ at the $(l-1)$-th layer to unit $j$ at the $l$-th layer for $1 \leq i \leq n(l-1)$, $1 \leq j \leq n(l)$, and $1 \leq l \leq L$.

The training starts by modifying the weights of an initial network, for instance $n(0)$-1-1,...-$n(L)$ network, which has 1 unit in each hidden layers and the fixed $n(0)$ and $n(L)$ units in the input and output layers respectively. When a local minimum is reached, candidate units at layer $l$ $(1 \leq l \leq L-1)$ are evaluated for possible addition.

### 6.2.1 Adding Phase

To add resources, we define the sensitivity measures and the probabilistic rules as follows.

Let $n(l)+1$ be a candidate unit at layer $l$ where $1 \leq l \leq L-1$.

**Definition 6.1** *The sensitivity $S_{li}$ of connection $w(i,n(l)+1,l)$ of the candidate unit at layer $l$ is:*

$$S_{li} = [\frac{\partial E}{\partial w(i,n(l)+1,l)}]^2, \tag{6.1}$$

*for* $1 \leq i \leq n(l-1)$.

**Definition 6.2** *The sensitivity $S_l$ of candidate unit $n(l)+1$ at layer $l$ is*

$$S_l = \sum_{i=1}^{n(l-1)} S_{li}. \tag{6.2}$$

**Definition 6.3** *The probability that the candidate unit at layer $l$ gets chosen is*

$$\Pr \textit{(a unit at layer } l \textit{ is chosen)} = \frac{S_l}{\sum_{m=1}^{L-1} S_m}. \tag{6.3}$$

**Definition 6.4** *The probability that the connection $w(i,n(l)+1,l)$ gets chosen given the candidate unit $n(l)+1$ is selected is*

$$\Pr(w(i,n(l)+1,l) \textit{ is chosen given the l-th layer selected}) = \frac{S_{li}}{S_l}, \tag{6.4}$$

*for* $1 \leq i \leq n(l-1)$.

One complete adding phase is done as follows. Each incoming connection of a candidate unit is initially set to be zero while its outgoing connections are picked to be small random numbers. Then the sensitivity of each incoming weight is evaluated and the sensitivity of each candidate unit is obtained by summing up the sensitivities over all its incoming weights through equation (6.1). The probability rule defined in equation (6.3) is used to select a unit. Specifically, if a number generated randomly from a uniform distribution on (0,1) exceeds the probability calculated from equation (6.3), the candidate unit $n(l)+1$ is selected. Once a unit is chosen, we apply the same procedure to choose connections for this unit using the probability defined in equation (6.4). Then the weights associated with the newly added unit are adapted while the old ones are kept frozen. After that, all the weights are trained simultaneously until a new local minimum is reached.

### 6.2.2 Deleting Phase

The goal of the deletion is to remove those connections that cause the least error increase. Two schemes have been investigated. In the first approach, the deleting phase is considered to be an inverse process to the adding phase, and an insensitivity measure for each weight is used as a criterion for its possible deletion. Specifically, the average insensitivity $IS(i,j,l)$ of weight $w(i,j,l)$ is defined as follows for $1 \leq i \leq n(l-1)$, $1 \leq j \leq n(l)$ and $1 \leq l \leq L$:

**Definition 6.5** *The insensitivity of the weight $w(i,j,l)$ is*

$$IS(i,j,l) = \frac{1}{\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\partial E}{\partial w(i,j,l)}\right)_t^2}, \tag{6.5}$$

*where $t$ represents the $t$-th iteration for $w(i,j,l)$ and $T$ is the total number of times the connection $w(i,j,l)$ has been modified since it was generated.*

Weights with big insensitivities get deleted. This approach, motivated by a method used in [8], is easy to use since it does not cost significantly extra computation to evaluate the insensitivity. However, since the average insensitivity does not always indicate the importance of a weight, a wrong deletion may occur. Therefore, another deleting method based on exhaustive search has also been examined. Specifically, this scheme is defined as follows.

**Definition 6.6**

$$\textit{Set } w(i,j,l) = 0 \textit{ if } E(w(i,j,l) = 0) = \min_{w(k,m,n)\neq 0} E(w(k,m,n) = 0), \tag{6.6}$$

*where* $1 \leq i \leq n(l-1)$, $1 \leq j \leq n(l)$ *and* $1 \leq l \leq L$. *Here* $E(w(k,m,n) = 0)$ *is the error obtained when the weight* $w(k,m,n)$ *is set to zero.*

Using this method, each weight is set to zero consecutively and the corresponding error is computed and stored. The weight which causes the smallest error increase gets deleted. This approach always decides correctly which weight should be removed. The computational complexity is polynomial in $W$ ($\sim O(W^3)$) since computing the corresponding errors for all nonzero weights costs about $MW^2$ multiplications and $M \sim O(\frac{W}{\epsilon})$. However, this can still be quite time consuming in practice if $M$ is large.

Two versions of the addition-deletion algorithm are given below, based on two different criteria, to control the addition and deletion phases and the termination of learning.

### 6.2.3 The First Version of the Algorithm: Al-vc

#### a) The Criterion : $M \sim O(\frac{W}{\epsilon})$

A network learns a problem if the training error rate is as low as possible and the performance between the training and true error rates is consistent. The consistency can be accomplished if the number of training samples loaded, $M$, sufficiently exceeds the VC-dimension. Specifically, if any network in a class of networks with VC-dimension $d$ can classify a fraction $1-\epsilon$ of the training samples, then with good confidence (when the number of samples is large), its true probability of error will be no bigger than $2\epsilon$ if the relationship $M \sim C\frac{d}{\epsilon}$ is satisfied, where $C$ is a constant. For our case, we choose $W$ as an estimator of $d$. Then the algorithm based on this criterion can be described as follows:

**b) The Algorithm: Al-vc**

1) Pick an initial error rate $\epsilon_0$, and a constant $C$. Ideally we would like to pick $\epsilon_0$ equal to the Bayes error. However, since we do not know what the Bayes error is, $\epsilon_0$ is chosen according to the performance we only hope to achieve.

2) Add resources to the initial network during learning until the network achieves the error rate $\epsilon_0$. If now $M \geq C\frac{W}{\epsilon_0}$ is satisfied, then stop.

3) Otherwise turn to the deleting phase and compute the error rate on the training samples whenever deletions of connections are about to occur. Stop when the relation $M \sim C\frac{W}{\epsilon}$ is satisfied. This new error rate $\epsilon$ at the stopping point will be bigger than $\epsilon_0$. If $\epsilon$ is sufficiently low, stop.

4) If $\epsilon$ is unacceptably high, which may be due to over-deletion of connections, repeat steps 2) and 3). If repeated growth and the deleting phases do not reduce $\epsilon$, then we accept $\epsilon$ as our estimate for the Bayes error.

### 6.2.4 The Second Version of the Algorithm: Al-vl

**a) The Criterion: validation error $\epsilon_{vl}$**

Although the algorithm Al-vc is based on a simple criterion which can estimate the generalization on line without a test set, it does have disadvantages. Specifically, there is no systematic way to choose the constant $C$ for each individual problem, not to say that the sample complexity needed for generalization can be much smaller than the VC-dimension for a specific distribution, as was shown in Chapter 3. Therefore, to develop a more systematic method to estimate the generalization error, we incorporate a validation set in training. Specifically, the algorithm Al-vl is given as follows.

**b) The Algorithm: Al-vl**

In this algorithm, the training which refers to modifications of the weights is done using a training set, while a validation error is computed using a validation set.

1) Train an initial network and compute the validation error $\epsilon_{vl}$ at each iteration until a local minimum is reached, where two conditions are satisfied: a) The error can hardly change anymore. That is, $\sum_{i,j,l} = (\frac{\partial E}{\partial w(i,j,l)})^2$ is small that a given small quantity $\delta$; b) The validation error does not change anymore. That is, the change of the validation error is smaller than a given small number $\delta'$. Denote $\epsilon_{vl}(k)$ as the validation error when local minimums have been reached the $k$ time. For the initial network to reach a local minimum the first time, $k = 1$.

2) Go to the adding phase until another local minimum is reached. Then $\epsilon_{vl}(k+1)$ is obtained.

If the new validation error $\epsilon_{vl}(k+1)$ at the $k+1$-th local minimum after the addition is smaller than the validation error $\epsilon_{vl}(k)$ at the previous (the $k$-th) local minimum before the addition occurs, i.e. $\epsilon_{vl}(k+1) < \epsilon_{vl}(k)$, goto 2) until $\epsilon_{vl}(k+1) \approx \epsilon_{vl}(k)$ for the first time. Then go to 2) again to add one more time. If this cause the validation error to decrease again, go to 2); otherwise go to 3), since the validation error indeed can not decrease anymore.

3) Go to the deleting phase until $\epsilon_{vl}$ goes up at a local minimum, and record $\epsilon_{vl}$ as $\epsilon_{vl}(k')$. Then go to the add the resource one more time until a new local minimum is reached and $\epsilon_{vl}(k'+1)$ is obtained. If $\epsilon_{vl}(k'+1) < \epsilon_{vl}(k')$, got to 2); otherwise stop.

At the same time always recored the optimal weights which correspond to the

| | Two-layer | Three-layer |
|---|---|---|
| Target network (with random weights) | 15-7-1-1 | 7-4-3-1 |
| Desired training error rate | 10% | 5% |
| Number of training samples | 1100 | 850 |
| Number of test samples | 1100 | 850 |
| Success in finding number of layers | 7/10 | 10/10 |
| Average number of units obtained at layer 1(*) | 4.4 | 3.5 |
| Standard deviation for (*) | 1.2 | 0.5 |
| Average number of units obtained at layer 2 (**) | 1.3 | 2.9 |
| Standard deviation for (**) | 0.2 | 0.8 |
| Range of the test error rate | 10.9% $\sim$ 16.7% | 4.0% $\sim$ 5.9% |

Table 6.1: Simulation results for learning target networks.

smallest $\epsilon_{vl}$ so that one can always go back and use the optimal set of weights.

## 6.3 Simulations

### 6.3.1 Learning Target Networks

The algorithm Al-vc is used for this problem with the constant $C$ chosen as 1.

In this problem, two random target networks ($n(0)$=15, $n(1)$=7, $n(2)$=1, $n(3)$=1 and $n(0)$=7, $n(1)$=4, $n(2)$=3, $n(3)$=1) are used to generate labels for random vectors whose elements are drawn independently with uniform distribution in 15 and 7 dimensional hyperspheres respectively. $L$, the maximum number of layers possible, was chosen to be 3. Ten runs using different random initial conditions were carried out for both cases with initial 15(7)-1-1-1 networks, respectively. After completion of learning, the performance of the resulting network was tested with new samples from the two target networks.

The expected generalization here is twofold. First, the algorithm should be able to result in networks which have similar structures to those of target networks, especially to distinguish the 2-layer (15-7-1) from the 3-layer (7-4-3-1) structure. Second, consistency should be achieved, i.e. the test error rate should be bounded by twice the training error rate.

The results given in Table 6.1 demonstrate that the algorithm only fails three out of twenty runs in finding the correct number of layers. Since a small error is allowed on the training sets, the resulting networks usually have a fewer number of units than the target networks. The test error, as expected, is well bounded by twice the training error.

### 6.3.2 Handwritten Digit Recognition

Although the algorithm Al-vc works well with $C = 1$ for the previous problem, it is not clear at all what an appropriate value of $C$ should be chosen for networks with multiple outputs. Nor is it clear for classification problems using majority rules (winner-take-all). For instance, as shown later in this section, for the handwritten digit recognition, if $C$ is still chosen to be 1, the resulting network with 35 connections will have about a 12% error rate on the test set, which indicates that network is too small to approximate the mapping. Due to lack of a general way to choose $C$, a more general version of the algorithm, the algorithm Al-vl, is used in this section.

In the previous work, LeCun et al.[10] incorporated network designs into a 5 layer network and obtained good results in handwritten digit recognition. Some other work [13] [11] suggested that the under-trained large networks with more number of weights than necessary can also generalize surprisingly well. For our work, we

| Nets | E(train) | E(test)/E(reject) | E(test)/ E(reject) |
|---|---|---|---|
| $101-2-0-3^{*1}$ | 2.22 | 7.27/0 | 4.77/7.00 |
| $101-5-0-3^{*2}$ | 1.20 | 7.50/0 | 5.00/8.41 |
| $101-2-2-3^{*3}$ | 2.00 | 8.86/0 | 5.22/17.0 |
| $101-10-2-3^{*}$ | 2.00 | 7.24/0 | 5.00/5.91 |
| $101-20-0-3^{*}$ | 0.89 | 7.72/0 | 5.00/10.7 |
| $101-40-0-3^{*}$ | 1.05 | 7.72/0 | 5.00/8.64 |

Table 6.2: Simulation results for the handwritten digit recognition with and withoyt rejection.

expect to achieve two things beyond this: 1) obtain an appropriate network structure for this problem without using any a priori knowledge; 2) compare performance of small networks with that of large networks in order to understand the generalizing properties of large networks.

The training set we use contains 450 $10 \times 10$ binary $(1,0)$ images of handwritten digits: 3's, 6's and 8's obtained from post-office zipcode data. The test set has 440 similar samples. The simulations are usually done twice using different initial conditions.

Table 6.2 gives results obtained from one set of simulations. In the table, "$*1$", "$*2$" and "$*3$" indicate the networks obtained using the algorithm Al-vl with $W = 180, 240$ and $82$ respectively, where $W$ is the total number of weights of the network. The "$*$" here refers to the networks obtained using BEP along with the validation set. E(train), E(test) and E(reject) are the training, testing and rejection error rates (%) respectively.

These results show that the small (two-layer) networks obtained through our algorithm Al-vl have comparable test errors to those of the large networks when the rejection rate is zero. When rejections are allowed with the same test error $\sim 5\%$, the small (two-layer) networks usually have smaller rejection rates than the larger networks except for the $(101 - 10 - 2 - 3)$ network. If the network is too small (too few connections), the $(101 - 2 - 2 - 3)$ net for instance, its performance on the test samples may deteriorate due to possibly insufficient resources.

To test how these networks generalize to noisy images, the resulting networks are tested on two types of noisy patterns. First, the $1 - 0$ type of noisy pattern is generated, for which the 1's in each image are changed to 0 with probability .05. Second, patterns with additive white noise are produced by adding a number generated from the uniform distribution in $[0, .5]$ to each bit. These networks are tested on the noisy digits generated with and without tolerance for rejections. The results shown in Table 6.3 demonstrate that small networks with two layers seem to be robust to the $(1 - 0)$ noise in the inputs, while large networks are more robust to the additive white noise in the input patterns. In this table, E$(1 - 0)$, E$(1 - 0$ reject$)$, E(w-n) and E(w-n-reject) represent the test error and rejection rates on digits with the $1 - 0$ and white noise respectively. The rest of the notation follows that in Table 6.2.

From the above results, the advantage of using a 3-layer structure is not so obvious. This may be due to the simplicity of the problem. Yet one important feature exhibited by these results is that networks larger than necessary in terms of number of weights can generalize surprisingly well. The reason from our conjecture is that for a relatively simple problem where samples belonging to different classes are well separated, an under-trained large network can also generalize well. This may well be the case for the digit recognition problem, since in a 100 dimensional space, they may be well-separated indeed. This conjecture may be further verified by applying

| Nets | E(1 − 0)/E(1 − 0 reject) | E(w-n)/E(w-n-reject) |
|---|---|---|
| $101-2-0-3^{*1}$ | 8.18/0, 5.00/13.4 | 16.6/0, 8.18/51.6 |
| $101-5-0-3^{*2}$ | 8.41/0, 5.23/12.3 | 18.6/0, 5.00/44.8 |
| $101-2-2-3^{*3}$ | 11.1/0, 5.23/26.8 | 20.7/0, 10.0/50.0 |
| $101-10-2-3^{*}$ | 9.55/0, 5.00/9.32 | 13.0/0, 5.23/30.0 |
| $101-20-0-3^{*}$ | 11.4/0, 5.23/19.5 | 12.7/0, 5.00/25.7 |
| $101-40-0-3^{*}$ | 10.0/0, 5.23/13.9 | 8.41/0, 5.00/9.32 |

Table 6.3: Test results on the noisy handwritten digits.

| Nets | $W$ | $C$ | E(train) | E(test)/E(reject) |
|---|---|---|---|---|
| 101-2-2-3 | 35 | 1 | 5.80 | 12.3/0 |
| 101-3-3-3 | 72 | 2 | 5.00 | 8.86/0 |

Table 6.4: Simulation results for the hand written digit recognition using the algorithm Al-vc.

Fisher discriminant [4] in multi-dimensions to the inputs. We shall dwell on this here.

To show that over-constraining a network may end up with poor generalization, results obtained using the first version of the algorithm Al-vc are given in Table 6.4, where $W$ represents the total number of weights of the network, and $C$ the constant in the criterion. .

## 6.4 A Problem for Which Large Networks Can Not Generalize

### 6.4.1 Learning a "Critical" Target Perceptron

To investigate why large networks can generalize so well for classification problems (binary mappings) but not for analog mappings, as shown in Chapter 4, a simple example is chosen in which networks of different sizes are trained to learn a target network which is a single-neuron $(15-1)$ network with random weights. The training samples are generated independently from a uniform distribution in $[-3, 3]$. Two networks $(15-1$ and $15-20-1)$ are trained with BEP using 600 training samples, and then tested on another 1000 randomly drawn samples from the same distribution. Their training and test error rates are all below 2.5% and 4.2% respectively. However, when both networks are tested on 1000 samples which are very close to the original decision boundary, inputs whose outputs are in the range $[-.1, .1]$ for the sigmoidal neuron of the random target network, they both make about 50% errors on the test samples! The reason for this is there are only 0.2% samples in the previously used 1600 samples which are close to the boundary. That is, the randomly drawn samples from two classes are very well separated. So the resulting small network is actually tilted away from the original decision boundary while the large one is wiggling around it.

Therefore, to investigate a case that an under-trained large network can not generalize well, we use the training and validation sets of the same sizes as before, but the samples are only the ones close to the boundary (within the $[-.1, .1]$ range) defined by the target perceptron. Similar training for networks with different sizes is carried out; and the results are given in Table 6.5, where "$*1$" and "$*$" represent the networks obtained using the algorithm Al-vl with 15 connections and BEP along with the validation set, respectively. E(train) and E(test) are the training and test error rates. The networks that are trained all start with very small random initial weights in the range $[-a, a]$, where $a < 10^{-5}$. Consistent with the perceptron convergence theorem [4], the addition-deletion algorithm with the initial $15-1$ network results in a single neuron which generalizes well. The $15-2-1$ network trained with BEP also

| Nets | E(train) (%) | E(test) (%) |
|---|---|---|
| $15 - 1^{*1}$ | 8.70 | 9.80 |
| $15 - 2 - 1^{*}$ | 6.89 | 9.40 |
| $15 - 10 - 1^{*}$ | 5.50 | 53.0 |

Table 6.5: Simulation results for learning the critical target perceptron.

converges to a network with equivalently the same number of weights as the target perceptron (some of the weights in the resulting networks are almost zero), although the loading of the training samples takes quite long. The $15-10-1$ network, however, learns the training samples but completely fails to learning the mapping.

## 6.5 Conclusion

The algorithm presented in this chapter finds small networks (which according to Barron [1] requires a small number of training samples) that give good generalization. Alternatives are to hand pick the network (LeCun [10]) and still have good generalization if the choices are good, or else to use large under-trained networks. The latter still work well because of the Rummelhart argument [15] if the problem is "simple". However, if the mapping is complex and requires multiple layers with specific connectivity patterns, small networks obtained through our algorithm give better generalization.

We also need to comment here on the relative time it takes to converge the different algorithms and how we can combine the three methods (1. clever design, 2. undertrained, larger than necessary networks, 3. data-driven architecture design, such as in our approach). The under-trained large networks usually learn faster than the small ones, especially for "simple" problems which are commonly encountered in classification problems. Moreover, it is preferable to use a two-layer structure instead of multilayers if it is efficient for implementing the mapping, since learning slows down significantly when the number of layers is bigger than 2.

In terms of choice of an algorithm, it is certainly best to incorporate clever designs into a network if some a priori knowledge is available, then to use a data-driven approach to find the complete network architecture. However, if no a priori knowledge is available, the network addition-deletion algorithm can always be used to find a good structure. It seems to be reasonably safe to use under-trained large networks in a lot of classification problems, if the generalization requirement is not that strict.

Finally, we point out that incorporating a validation set does not guarantee a completely reliable estimate for the true generalization error. Again, according to the VC theory [3], the difference between a validation error and the true probability of error is also roughly bounded by $O(\frac{d}{M_t})$, where $d$ is the VC-dimension of the class of networks and $M_t$ is the number of test samples. That is, sufficient validation samples are needed to obtain a reliable estimate. A large validation set, however, will increase the training time. From our experience, if the samples are independently drawn and not very noisy, it seems to be sufficient to use a small fraction of training samples as a validation set.

# Bibliography

[1] A. Barron, " Approximation and Estimation Bounds for Artificial Neural Networks," *Proc. of The 4th Workshop on Computational Learning Theory*, 243-249, 1991.

[2] J.S . Denker, Y. LeCun and S.A . Solla, "Optimal Brain Damage," *Advances in Neural Information Processing Systems*, Vol. 2, 598-605, 1989.

[3] L. Devroye, "Automatic Pattern Recognition: A Study of Probability of Error," *IEEE Trans. on Pattern Recognition and Machine Intelligence*, Vol. 10, No. 4, 1162-1179, July 1988.

[4] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

[5] A.E . Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," *Advances in Neural Information Processing Systems*, 2, 524-532, 1989.

[6] M. Frean, "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks," *Neural Comp.* 2, 198-209, 1990.

[7] C. Ji, R. Snapp and D. Psaltis, "Generalizing smoothness constraints from discrete samples," *Neural Comp.* 2, 190-199, 1990.

[8] E.D . Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," *IEEE Trans. on Neural Networks*, vol. 1. no. 2, June 1990.

[9] Y. LeCun, J.S . Denker and S.A . Solla, "Optimal Brain Damage," *Advances in Neural Information Processing Systems*, 2, 598-605, 1989.

[10] Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comp.* 4, 541-551, 1990.

[11] Y. Lee, "Handwritten Digit Recognition Using K Nearest Neighbour. Radial-Basis Function, and Backpropagation Neural Netowrks," Neural Computation, vol. 3, no. 3, 440-449, Fall 1991.

[12] R. Lippmann, "Pattern Classification Using Neural Networks," *IEEE Communications Magazine*, vol. 27, no. 11, 43-64, November 1989.

[13] G.L . Martin and J.A . Pittman, "Recognizing Hand-Printed Letters and Digits using Back Propagation Learning," Neural Computation, vol. 3, no. 2, 258-267, Summer 1991.

[14] J. Nadel, "Study of a Growth Algorithm for Neural Networks," *Int. J. Neural Syst.*, 1, 55-59, 1989.

[15] D.E . Rummelhart, personal communications.

[16] V.N . Vapnic and A. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability and its Applications*, vol. 16, 264-280, 1971.

[17] A. Weigend, D.E . Rumelhart and B.A . Huberman, "Generalization by Weight Elimination with Application to Forecasting," *Neural Information Processing Systems*, vol 3, 875-882, 1990.

# Chapter 7

# Conclusion

In this thesis, the generalization capability of feedforward multilayer neural networks is investigated analytically and algorithmically. The theoretical results obtained for generalization can be summarized as follows.

First, a general relationship is developed between the statistical capacity and the universal sample bound for generalization. It is shown that the universal sample bound for generalization is lower-bounded by the lower epsilon-capacity of a particular network for one specific distribution of the samples. This relation associates the VC-dimension with the Baysian classifier and the optimal classifier in a class of networks with the same structure. It is not clear yet, however, whether an upper bound in the order of the upper capacity is also plausible. Moreover, it would be interesting to investigate whether the Cover type of capacity is also in the order of the VC-dimension, in order to obtain a complete unifying view of the capacity and the sample bound for generalization.

As one of its applications, this general relation is used as a constructive approach to find a lower bound for the VC-dimension of two layer networks with binary weights.

It could also be used in a similar fashion in the future to find a lower bound for the VC-dimension of any networks of interest, such as three-layer networks or two-layer networks with discrete weights which take $K$ values.

Second, as another application of the general relation, the sample complexity needed for generalization for a distribution-dependent case is studied through investigating a single neuron under a specific sample distribution. It is shown that the sample complexity actually needed for generalization can be much smaller than the VC-dimension. In general, it would be useful to obtain a sample bound for generalization for a class of commonly used distributions.

Third, as an independent chapter, bounds for the capacity of two layer networks with binary weights are obtained through a statistical approach. The approach can be extended to evaluate the capacity of other types of networks.

One of the theoretical aspects of generalization which has not been studied yet is how under-trained large networks generalize. This type of network has demonstrated many interesting features through our simulations. A valid measure for their effective degrees of freedom would be a useful first step toward understanding their generalizing capability. Some other properties of the under-trained large networks such as robustness to noisy inputs could also be interesting to study.

Guided by our theoretical results, two learning algorithms are developed and shown to be capable of finding a suitable structure of a network during learning without any a priori knowledge. These algorithms make it possible to explore how the size and structure for a network affects its ability to generalize. Specifically, most of our simulations are focused on comparisons of small with large networks in generalization.

The first learning algorithm, the network reduction algorithm, is applied to obtain networks capable of learning analog mappings. Particularly, through a real application, training a network to control a two-link manipulator to draw letters, we find that learning analog mappings is "difficult". Moreover, small networks obtained through this algorithm generalize better than under-trained large networks. That is, for learning analog mappings, it is preferable to use a network with fewer degrees of freedom possible, which is the number of modifiable weights.

The second learning algorithm, the network addition-deletion algorithm, provides a general method for finding a small network with an appropriate structure. It is used in this thesis to investigate the effect of network size on generalization for binary mappings, though it is general enough to be used for analog mappings as well. Through various simulations on classification problems, we find that most of these problems are "simple", i.e., the samples belonging to different classes are reasonably well separated. For problems such as handwritten digit recognition, the small networks obtained through our algorithm perform slightly better than under-trained large ones. However, for complex binary mappings such as learning the "critical" perceptron, it is important to use small networks which can learn training samples in order to achieve good generalization. More simulations are needed to investigate further the generalizing capability of under-trained large networks.

For the problems we have encountered, the 3-layer structure seems to be immaterial, since most of these problems can be solved using a simple two-layer structure. It could be useful to find types of problems for which more layers are preferable.

In terms of learning time, for relatively simple problems such as handwritten digit recognition, searching for a small network using our algorithms is more time consuming than using an under-trained large network. Training usually slows down significantly when the number of layers increases. And it seems to be easier for a three-layer network to get stuck at a local minimum than a two-layer network. More simulations are needed, however, to carefully compare the learning time. For real applications, it may be more efficient to incorporate prior knowledge into preprocessing or network designs and leave the very part that can not be dealt with this way to the network addition-deletion algorithm.