Statistical Mechanics of Learning and Optimization

物理学

井上純一

Thesis

Statistical Mechanics of Learning and Optimization

Jun-ichi Inoue

January 1998

Department of Physics, Tokyo Institute of Technology
Meguro-ku, Oh-okayama 152, Japan

Preface

Recently statistical physicists have extended their research fields to complex systems which have not been regarded as subjects of traditional physics. Statistical mechanics has originally been developed as a useful tool to investigate the various physical properties of condensed matters in which about 10^{23} elements interact each other in the complicated fashion. In real life, there are a lot of similar situations, including social science such as economics, in which lots of simple elements interact each other and their collective behavior is sometimes beyond our expectation.

Among such complicated systems, the brain has been one of the most attractive subject. In the real brain, there exist about 10^{12} neurons and each of them is connected with the other neurons of order 10^5 by synaptic efficacy. For this highly complicated non-linear system, many researchers in the statistical mechanics community have partially succeeded in explaining some aspect of brain functions. They assume that the synaptic connections are randomly distributed and a single neuron is a binary threshold unit which takes $+1$ or -1 in analogy with random spin systems like spin glasses.

However, if success of statistical mechanics consists only in explaining some aspect of brain functions, or giving a way for understanding of macroscopic psychological phenomena from a microscopic point of view, we are obliged to think that researchers who have used statistical mechanics only changed their subjects and succeeded in giving an explanation for experiments of the new subjects. I think the success of statistical mechanics consists not only in giving an explanation of physiological experiments but in providing the engineers or information scientists, who need practical applications, with some mathematical concepts or ideas to construct a new type of computer machine based on the way of information processing in the real brain. Thus, statistical mechanics has two aspects of contribution to the brain science;

- Explanation for physiological phenomena.
- Providing mathematical foundation for constructing new type of information processing or optimization.

For a remarkable example of the former contribution, statistical mechanics has succeeded in showing a limitation of the learning ability of some specific models of the real brain (the so-called neural network model), which is one of the fundamental brain functions. As the statistical mechanical analysis for the machine learning has some advantages in comparison with the other approaches based on traditional mathematical statistics, we can obtain a deep understanding of the learning mechanism in a large network model (the size N goes to infinity) which has a particular structure.

Learning is regarded as a process of finding a suitable parameter of the machine so as to make the machine adapt to the environment. Then, the success in learning is measured by the cost function which represents the degree of mis-adaptation to the environment. In the so-called supervised learning context, the environment to which the machine should be adapted is referred to as the input-output relations of the teacher machine. If the teacher and the student machines have the same structure, the learning corresponds to the producing a clone of the teacher machine. However, such a case is somewhat ideal in comparison with the real world. We should investigate more realistic cases in order to extract some knowledges of the learning in the real brain from the mathematical model system. Therefore, we take up a realistic learning situation in the sense that the structures of teacher and student machines are different. For this case, we can not train the student machine so that the cost function becomes zero in principle.

If we would like to make an artificial learning machine in this context, it is necessary for us to choose the learning algorithm which minimizes the cost function as quickly as possible. This is also a typical

problem of optimization. In general, the cost function has a lot of minima, like the energy landscape of spin glasses at low temperatures, and it is very difficult to apply the conventional method like a gradient decent to obtain the global minimum. In optimization for such a multi-valley cost function, one should prevent the system from being trapped in one of the local minima. Some mechanism of escaping from the local minimum is needed and thermal fluctuation is suitable for this purpose. The simulated annealing method is based on the thermal fluctuations and is regarded as one of the most effective heuristic methods for the problem of combinational optimization.

These two keywords *learning* and *optimization* have attracted much attention of the researchers who work in the cross-disciplinary fields.

In this thesis, the following problems concerning these two keywords are investigated in detail from a statistical mechanical point of view;

- How does the neural network model in realistic situations learn from the environment ?
- What is the most effective strategy for the learning machine ?
- What is the optimal performance for a specific neural network model ?
- How do thermal fluctuations act effectively on problems of optimization ?
- What is the limitation for the optimization method based on thermal fluctuations ?

Most of the results we obtain in this thesis are rigorous or exact within our model system.

Acknowledgements

This work leading up to this thesis was done during my years as a graduate student in the condensed matter theory group at Tokyo Institute of Technology. These five years have been very stimulating and instructive, and the liberal atmosphere in the group has given rise to several interesting ideas on topics related to my research.

First of all, I would like to thank my supervisor, Professor Hidetoshi Nishimori not only for his support and encouragement during this work but also allowing me to work freely in the hearty atmosphere. Special thanks also goes to Dr Yoshiyuki Kabashima for collaborations and introducing me to a lot of interesting topics in machine learning. I also want to take this opportunity to thank Professor Shun-ichi Amari for allowing me to join his laboratory for Information Synthesis in the Institute of Physical and Chemical Research (RIKEN) and useful discussions. I would like to thank Drs Noboru Murata, Siegfried Bös and H H Yang who are members of Professor Amari's group in RIKEN, for interesting discussions and useful advice. I want to thank explicitly Professors Ido Kantor and Desire Bolle, Drs Michael Wong and A C C Coolen for stimulating discussions when they were visiting our group. A special thanks should go to Professor C Van den Broeck for interesting discussions and useful comments on this work when I participated in the international conference in Hong Kong "Theoretical Aspects of Neural Computation (TANC-97)". I would like to thank Drs Masato Okada, Domenico Carlucci, Naoki Kawashima, Toshiyuki Tanaka, Tsuyoshi Chawanya and Professor Constantino Tsallis for useful comments.

This work was partially supported by Junior Research Associate program of RIKEN.

This was a great fun to do. Thank you to everyone.

List of papers submitted for the requirement of this degree

1. **Retrieval phase diagrams of non-monotonic Hopfield networks**
   Jun-ichi Inoue
   J. Phys. A: Math. Gen. **29** (1996) 4815

2. **Statistical mechanics of the multi-constraint continuous knapsack problem**
   Jun-ichi Inoue
   J. Phys. A: Math. Gen. **30** (1997) 1047

3. **On-line learning of non-monotonic rules by simple perceptron**
   Jun-ichi Inoue, Hidetoshi Nishimori and Yoshiyuki Kabashima
   J. Phys. A: Math. Gen. **30** (1997) 3795

4. **On-Line AdaTron Learning of Unlearnable Rules**
   Jun-ichi Inoue and Hidetoshi Nishimori
   Phys. Rev. E **55** (1997) 4544

5. **A simple perceptron that learns non-monotonic rules**
   Jun-ichi Inoue, Hidetoshi Nishimori and Yoshiyuki Kabashima
   Proceedings of TANC-97 Hong Kong International Workshop on Theoretical Aspects of Neural Computation: A multidisciplinary Perspective, pp.257-266

6. **Generalization ability of a simple perceptron with non-monotonic transfer function**
   Jun-ichi Inoue, Hidetoshi Nishimori and Yoshiyuki Kabashima
   Phys. Rev. E **58** (1998) 849

7. **Learning of non-monotonic rules by simple perceptrons**
   Yoshiyuki Kabashima and Jun-ichi Inoue
   *J. Phys. A: Math. Gen.***31** (1998) 123

8. **Learning curves for non-monotonic rules — Statistical mechanical analysis**
   Jun-ichi Inoue, Hidetoshi Nishimori and Yoshiyuki Kabashima
   Proceedings of Annual Conference of Japanese Neural Networks Society in Kanazawa 1997 (JNNS '97 in Kanazawa) pp.166-167

9. **Convergence of simulated annealing by the generalized transition probability**
   Hidetoshi Nishimori and Jun-ichi Inoue
   J. Phys. A: Math. Gen. **31** (1998) 5561

Contents

Chapter 1

Introduction

1.1 What is learning ?

Let us consider a black box which produces an output $y \in \Re^m$ after receiving an input $x \in \Re^n$ according to some rule. It is convenient for us to suppose that the black box produces its output y according to some conditional probability $p(y|x;\theta)$ which means the probability of the output y on condition that an input x is received. A parameter $\theta \in \Re^k$ is a kind of control parameter of the conditional probability. The integers m, n and k means the dimensions of outputs, inputs and parameters, respectively. Although we deal with the case of $m = n = k = 1$ in this subsection, it is easy for us to extend our discussions to the case of higher dimensions. We can regard x as an element which is extracted from some probability distribution $p(x)$ and the detail of $p(x)$ specifies the environment in which the black box is placed.

Then, we call a pair of the conditional probability $p(y|x;\theta)$ and the distribution of inputs $p(x)$, namely,

$$\mathcal{S} = (p(x), p(y|x;\theta)), \tag{1.1}$$

system. If the shape of the conditional probability $p(y|x;\theta)$ is completely determined by the parameter θ, the parameter θ specifies the black box perfectly and we call this parameter *machine.* After the machine θ receives an input x form the distribution $p(x)$, the machine θ calculates the conditional probability $p(y|x;\theta)$ for the input x and produces the output y according to this probability $p(y|x;\theta)$. In contrast, let us suppose that input x and output y are fixed to some specific values y^0 and x^0. Then it is obvious that the conditional probability $p(y|x;\theta)$ means the distribution of the machine θ which is able to produce the set of an input-output relation (x^0, y^0).

If one knows $p(y|x;\theta)$ and $p(x)$, the statistical properties of the system are completely determined as

$$P(x, y;\theta) = P(y|x;\theta)P(x). \tag{1.2}$$

If the machine is determined by some function $f(x;\theta)$ instead of a conditional probability, we can easily extend our notation using the delta function as

$$\mathcal{S} = (p(x), \delta(y - f(x;\theta))). \tag{1.3}$$

Figure 1.1: The black box. Input x vs. output y relations of the box is determined by a conditional probability $p(y|x;\theta)$.

x	5.27	5.08	6.25	7.21	8.02	8.71	8.42
y	12	18	24	30	36	42	48

Table 1.1: The two-dimensional data.

For each case, we should bear in mind that it is possible for us to refer only to pairs of the inputs and outputs and we have no direct knowledge of the conditional probability $p(y|x;\theta)$, especially, the parameter θ. Then, an interesting question arises.

Is it possible for us to identify the mechanism of the black box, namely, the conditional probability $p(y|x;\theta)$ from only a set of input-output relations ? Then, if we remember that the conditional probability is specified by the parameter θ, we can put the above question in another way; Can we specify the parameter θ after referring to a set of input-output relations ?

Such a kind of question is a very general and important problem in the real world. For example, we sometime should fit experimental data of natural science, economics, *etc.* to a line or curve and derive a law from the data. In general, such a line or curve is specified by several parameters and we should estimate the suitable fitting parameters. In order to make our concept clear, we discuss the following simplest model.

Let us suppose that we received the N experimental data $(x_i, y_i), i = 1, \cdots, N$. For this case, we should regard that $x_i (i = 1, \cdots, N)$ is input and $y_i (i = 1, \cdots, N)$ is output and the dimensions of the input and output are $m = 1$ and $n = 1$, respectively. In general, we do not know what mechanism or law exists behind the data, however, let us assume that these data are generated by the next simple relation;

$$y = \theta_1 x + \theta_2 \tag{1.4}$$

where $\theta = (\theta_1, \theta_2)$ means the parameter and the dimension of this parameter is $k = 2$. Therefore, this system can be written formally as

$$\mathcal{S} = (p(x), \delta(y - \theta_1 x - \theta_2)) . \tag{1.5}$$

Then, our problem is how we determine the parameter θ from a finite set of input-output relations $(x_i, y_i), i = 1, \cdots, N$. This problem is called the *linear regression problem.*

For this one-dimensional linear regression problem, we usually use the *least square method* as the strategy to obtain the parameter. We first consider the following quantity from the observable, namely, N input-output relations;

$$T(\theta) = \sum_{i=1}^{N} [y_i - (\theta_1 x_i + \theta_2)]^2 \tag{1.6}$$

According to the least square method, we determine the suitable parameter θ^* so that $T(\theta)$ takes a minimum value at the suitable parameter. Therefore, we should perform

$$\left. \frac{\partial T(\theta)}{\partial \theta} \right|_{\theta^*} = 0 \tag{1.7}$$

Then we obtain the suitable parameter as

$$\theta_1^* = \frac{N \sum_{i=1}^N x_i y_i - (\sum_{i=1}^N x_i)(\sum_{i=1}^N y_i)}{N \sum_{i=1}^N x_i^2 - (\sum_{i=1}^N x_i)^2} \tag{1.8}$$

$$\theta_2^* = \frac{(\sum_{i=1}^N x_i^2)(\sum_{i=1}^N y_i) - (\sum_{i=1}^N x_i)(\sum_{i=1}^N x_i y_i)}{N \sum_{i=1}^N x_i^2 - (\sum_{i=1}^N x_i)^2}, \tag{1.9}$$

and we can estimate that the data are produced by the relation $y = \theta_1^* x + \theta_2^*$. As a simple examination, we apply the least square method to the data which are listed in Table 1.1. After simple calculations, we obtain $y = 10x - 40$. In Figure 1.2, we plotted the observed data listed in Table 1.1 and the line which was observed by the least square method.

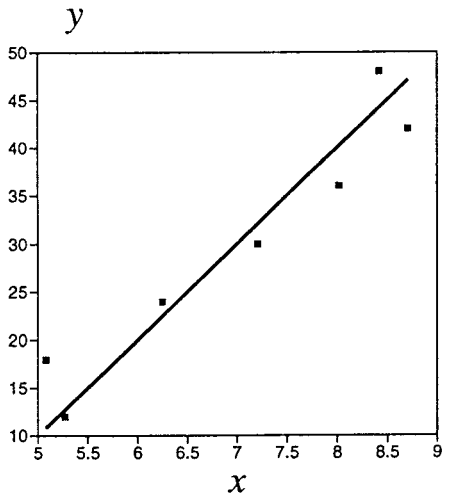

Figure 1.2: The one-dimensional linear regression problem. The dots are observable data. The line $y = 10x - 40$ is the result of the least square method.

Let us therefore define the machine learning as follows;

Learning

Finding the optimal value of parameter θ of the system from a finite set of input-output relations.

For the above one-dimensional linear regression problem, the optimal value of the parameter corresponds to θ^* and the strategy to obtain the optimal value, namely, the least square method, is referred to as the *learning algorithm.*

In the next section, we introduce neural network models as a typical example of learning systems.

1.2 Neural network model

Neural networks were proposed not only as a model of the real brain functions but also as an alternative to von Neumann type computers to perform the task of processing input data into output data. McCullock and Pitts [1] introduced a simplest model for a single neuron as a binary threshold unit. Nowadays this model is called the *perceptron.* The unit (neuron) computes the weighted sum of its inputs (from other units) and produces $+1$ or -1 according to whether or not this sum is greater than an arbitrary specified threshold. One of the mathematical representations of the above statement is

$$y = f\left[\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N} - \theta\right] \tag{1.10}$$

where $\boldsymbol{x}$ is an input on the N-dimensional sphere with radius 1. On the other hand, the parameter $\boldsymbol{J}$ is called *synaptic weight vector* on the N-dimensional sphere with radius $\sqrt{N}$ and θ is some threshold [1]. Therefore, for this neural network model, the dimension of input $\boldsymbol{x}$ is $m = N$ and the dimension of the parameter $\boldsymbol{J}$ is also $k = N$ and the dimension of output is $n = 1$. If we regard the above unit as the system with some conditional probability as we discussed in the previous section, (1.10) is represented as

$$\mathcal{S} = \left(p(\boldsymbol{x}), \delta\left(y - f\left[\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N} - \theta\right]\right)\right) \tag{1.11}$$

where $p(\boldsymbol{x})$ is a uniform distribution on the N-dimensional sphere $|\boldsymbol{x}| = 1$.

[1] There are several candidates for the scaling of vector J. One should choose the scaling so that the argument of the function f becomes $\sim \mathcal{O}(1)$.

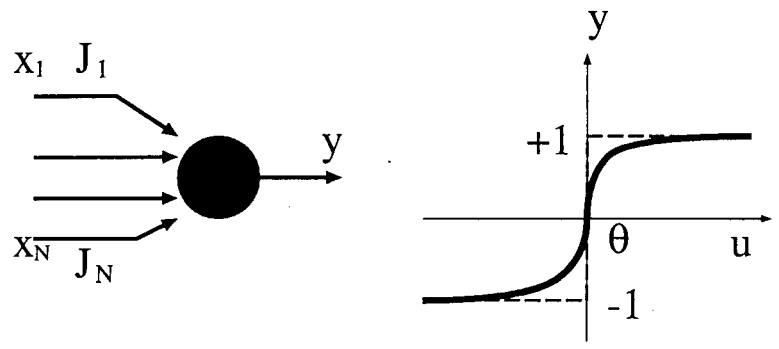

Figure 1.3: A single neuron model as a graded-response (solid line) and non-linear (broken line) unit.

The above mathematical model is based on the next two features of a single neuron in the real brain;

- A single neuron is connected with the other neurons of order $\sim 10^5$.
- A single neuron transfers the information as electric pulses and fires if the synaptic potential exceeds some threshold.

The *graded-response* function f in the definition (1.10) is called the *transfer function* or the *activation function*. The non-linear version of the transfer function also has been considered, for a typical example, as

$$y = \text{sign}\left[\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N} - \theta\right]. \tag{1.12}$$

We illustrate the graded-response and non-linear unit of a single neuron in Figure 1.3. The graded-response transfer function has the advantage of easy handling of mathematics. In contrast, the non-linear transfer function is robust with respect to some input noises.

This input noise robustness of the non-linear transfer function is understood as follows. Let us first consider that some input noise η is added to the input vector $\boldsymbol{x}$ as $\boldsymbol{x}\rightarrow\boldsymbol{x}+\eta$. Then, the internal potential changes as $\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N}-\theta\rightarrow\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N}-\theta+\boldsymbol{J}\cdot\eta/\sqrt{N}$. For the graded-response transfer function, the output of the unit changes due to the input noise by

$$f\left(\boldsymbol{J}\cdot\boldsymbol{x}\sqrt{N} - \theta + \boldsymbol{J}\cdot\eta/\sqrt{N}\right) - f\left(\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N} - \theta\right). \tag{1.13}$$

This takes a finite value as long as $\boldsymbol{J}\cdot\eta/\sqrt{N}$ is non-zero.

On the other hand, for the non-linear transfer function, the change of the output is given as

$$\text{sign}\left(\boldsymbol{J}\cdot\boldsymbol{x}\sqrt{N} - \theta + \boldsymbol{J}\cdot\eta/\sqrt{N}\right) - \text{sign}\left(\boldsymbol{J}\cdot\boldsymbol{x}/\sqrt{N} - \theta\right). \tag{1.14}$$

We should notice that this difference becomes zero as long as the signature function does not change. As the result, the unit with the non-linear transfer function is hard to be disturbed by the input noise.

In addition, as it is easy to argue by analogy between the unit with ± 1 output and Ising spin systems, the unit with the non-linear transfer function is familiar to statistical physicists.

One of the remarkable abilities of the above neuronal units is the *learning ability* (or the *adaptation ability*). In order to deal with this learning ability, we regard the input $\boldsymbol{x}$ and the corresponding output y as the *question* and the *answer* respectively. Then, a perceptron can realize a set of the desired question-answers by modifying its own synaptic weight $\boldsymbol{J}$ adaptively [1].

If these pairs of questions and correct answers $\{y, \boldsymbol{x}\}$ (what we call *examples* or the *training set*) are generated by a different neural network, this fashion of learning is called the *supervised learning* and the network which gives examples is called the *teacher* or *supervisor*. The network which should adapt the

examples is referred to as the *student* or *learner*. The examples are sometimes regarded as *teacher signals*. The student should modify his own synaptic weight $\boldsymbol{J} = (J_1, \cdots, J_N)$ in order to reproduce the teacher signals correctly.

For a single layer perceptron without threshold, the structure of the teacher is perfectly specified by its synaptic weight vector and transfer function. Therefore, if the teacher and student have the same structure, the student should find the correct weight (teacher weight) $\boldsymbol{B} = (B_1, \cdots, B_N)$.

Let us remember that the learning problem we discussed in the previous section, namely, the one-dimensional linear regression problem, is different from this supervised learning problem in the sense that there does not exist the target vector for the one-dimensional regression problem. For the one-dimensional linear regression problem, the signal or data (x_i, y_i) $i = 1, \cdots, N$ is presented explicitly, however, we cannot tell whether the modification of the fitting parameter θ is correct or not in principle. In contradiction to the supervised learning, such a learning without teacher signal, is called the *unsupervised learning*. In this thesis, we deal with the problems of the supervised learning.

1.3 Several learning algorithms

In order to achieve the good performance of learning from examples, a lot of learning algorithms have been proposed and improved. In this section, we introduce three typical learning algorithms.

1.3.1 Perceptron learning algorithm

In the early stage of the effort to construct the effective learning algorithm, Rosenblatt [2] proposed the so called *perceptron learning algorithm.*

The perceptron learning algorithm states that for a single neuronal unit like (1.12), one can find a solution $\boldsymbol{B}$ from a training set $\xi^t \equiv \{(y^1, \boldsymbol{x}^1), (y^2, \boldsymbol{x}^2), \cdots, (y^t, \boldsymbol{x}^t)\}$ within a finite training time.

The perceptron learning algorithm is described as follows.

Perceptron Learning Algorithm

START Choose an arbitrary weight $\boldsymbol{J}$.

TEST For an arbitrary example $(y^k, \boldsymbol{x}^k)$ in ξ^t, test whether the unit satisfies the input-output relation or not. If the unit is correct, go to **TEST**. If the unit is wrong, go to next **ADD**.

ADD Modify the weight of the unit as

$$\boldsymbol{J} \to \boldsymbol{J} + y^k \boldsymbol{x}^k,$$

and back to **TEST**.

For this algorithm, there is the following convergence theorem.

Perceptron Convergence Theorem

The perceptron learning algorithm can find the solution $\boldsymbol{B}$ at least within δ^{-2} times modifications, where, $\delta = \min_{(y,\boldsymbol{x})\in\xi^t}\{y \cdot v\} > 0$ with $v \equiv \boldsymbol{B}\cdot\boldsymbol{x}$ $(|\boldsymbol{B}| = 1)$.

Proof.

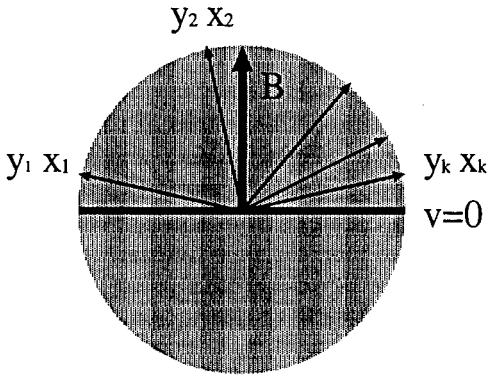

Figure 1.4: δ is the minimum projection of the input vector to teacher's weight.

At first, we rewrite the student weight $\boldsymbol{J}$ and the input vector $\boldsymbol{x}$ as $\boldsymbol{J} = (J_1, \cdots, J_N, \theta)$ and $\boldsymbol{x} = (x_1, \cdots x_N, -1)$ respectively. Then, we can neglect the threshold θ in the sign function.

i) For arbitrary $\boldsymbol{J}$,

$$G(\boldsymbol{J}) \equiv \frac{\boldsymbol{B}\cdot\boldsymbol{J}}{|\boldsymbol{J}|} \leq 1 \tag{1.15}$$

is satisfied because $|\boldsymbol{B}| = 1$.

ii) Let us regard $\boldsymbol{J}_i$ as a weight vector after the i-th modification.
Then,

$$\begin{aligned} \boldsymbol{B}\cdot\boldsymbol{J}_{i+1} &= \boldsymbol{B}\cdot(\boldsymbol{J}_i + y\cdot\boldsymbol{x}) \\ &= \boldsymbol{B}\cdot\boldsymbol{J}_i + y(\boldsymbol{B}\cdot\boldsymbol{x}) \\ &\geq \boldsymbol{B}\cdot\boldsymbol{J}_i + \delta. \end{aligned}$$

Therefore, using this relation n-times, we obtain

$$\boldsymbol{B}\cdot\boldsymbol{J}_n \geq n\delta \tag{1.16}$$

where we have used the initial condition $\boldsymbol{J}_0 = 0$ [2] . Next we calculate the square of $\boldsymbol{J}_{i+1}$ and find

$$\begin{aligned} |\boldsymbol{J}_{i+1}|^2 &= |\boldsymbol{J}_i + y\boldsymbol{x}|^2 \\ &= |\boldsymbol{J}_i|^2 + 2y(\boldsymbol{J}_i\cdot\boldsymbol{x}) + 1 \\ &< |\boldsymbol{J}_i|^2 + 1 \end{aligned} \tag{1.17}$$

where we have used $y(\boldsymbol{J}_i\cdot\boldsymbol{x}) < 0$. Therefore we obtain

$$|\boldsymbol{J}_n| < \sqrt{n}. \tag{1.18}$$

From (1.16) and (1.18), the function $G(\boldsymbol{J}_n)$ should satisfy

$$G(\boldsymbol{J}_n) = \frac{\boldsymbol{B}\cdot\boldsymbol{J}_n}{|\boldsymbol{J}_n|} \geq \sqrt{n}\delta. \tag{1.19}$$

[2] In this argument, we assume that the signature function sign(x) takes +1 if the argument of the function x is identically zero.

x_1	x_2	y	condition
-1	-1	-1	$-J_1 - J_2 - \theta < 0$
-1	$+1$	-1	$-J_1 + J_2 - \theta < 0$
$+1$	-1	-1	$J_1 - J_2 - \theta < 0$
$+1$	$+1$	$+1$	$J_1 + J_2 - \theta > 0$

Table 1.2: The truth table of the AND logic

x_1	x_2	y	condition
-1	-1	-1	$-J_1 - J_2 - \theta < 0$
-1	$+1$	$+1$	$-J_1 + J_2 - \theta > 0$
$+1$	-1	$+1$	$J_1 - J_2 - \theta > 0$
$+1$	$+1$	$+1$	$J_1 + J_2 - \theta > 0$

Table 1.3: The truth table of the OR logic.

iii) If we suppose that the number of modifications n is larger than δ^{-2}, namely, $n > \delta^{-2}$, (1.15) and (1.19) contradict each other. ■

At the end of this subsection, we would like to mention the value δ. We first should notice that as y_k is the output of the teacher, $y_k \boldsymbol{x}$ is restricted to the upper half area of the N-dimensional sphere in which the teacher's weight $\boldsymbol{B}$ exists (see Figure 1.4). Then, $\delta_k \equiv \boldsymbol{B} \cdot (y_k \boldsymbol{x}_k) = y_k (\boldsymbol{B} \cdot \boldsymbol{x}_k)$ represents a projection of an input $\boldsymbol{x}_k$ to the teacher's weight vector $\boldsymbol{B}$ and $\delta = \min_k \{\delta_k\}$ is the minimum projection. Therefore, if this value δ is small, the region in which the target $\boldsymbol{B}$ should exist is large. On the other hand, if δ is large, the region in which the target should exist is small. As the result, the inverse of δ or δ^{-2} is proportional to the times of modifications which we need to find the target in the restricted region [3] .

1.3.2 Error-back propagation learning algorithm

The perceptron learning algorithm is most general in the sense that it is guaranteed to converge to the solution of the given problem under the condition that *such a solution exists.* In order to explain that there exists a problem which a simple perceptron cannot solve, we would like to examine the several simple logical operations, namely, *AND*, *OR* and *XOR(exclusive or)*. For the simplicity, we treat the case of two-dimensional perceptron. For this two-dimensional perceptron, the input-output relation is given by

$$y = \text{sign}\,(J_1 x1 + J_2 x_2 - \theta)\,. \tag{1.20}$$

We first examine whether the above two-dimensional perceptron (1.20) can realize the following AND logic. In Table 1.2, the *condition* means the required condition for us in order to satisfy the input-output relation (x_1, x_2, y). It is easy for us to choose a suitable solution (J_1, J_2, θ) to satisfy the AND logic, namely, the four conditions in Table 1.2. We should choose $(J_1, J_2, \theta) = (1, 1, 0.5)$. Therefore, the two-dimensional perceptron can realize the AND logic and the perceptron learning algorithm we discussed in the previous section guarantees the convergence to the solution within a finite number of iterations. We next examine the OR logic using the same argument as the case of the AND logic. We listed the truth table in Table 1.3 of the OR logic (x_1, x_2, y) and the corresponding required conditions. We can easily find a candidate of the solutions $(J_1, J_2, \theta) = (1, 1, -0.5)$. Therefore, we conclude that the two-dimensional perceptron can be trained by the perceptron learning algorithm so that the perceptron finds the solution of the OR logic within a finite time step. From the above results with respect to the AND logic and the OR logic, it seems that the perceptron can realize every input-output relation. However, it is worth while to examine the following XOR logic which is listed in Table 1.4. We can see that it is impossible

[3] We sometimes refer to this region as *version space.*

x_1	x_2	y	condition
-1	-1	-1	$-J_1 - J_2 - \theta < 0$
-1	$+1$	$+1$	$-J_1 + J_2 - \theta > 0$
$+1$	-1	$+1$	$J_1 - J_2 - \theta > 0$
$+1$	$+1$	-1	$J_1 + J_2 - \theta < 0$

Table 1.4: The truth table of the XOR logic.

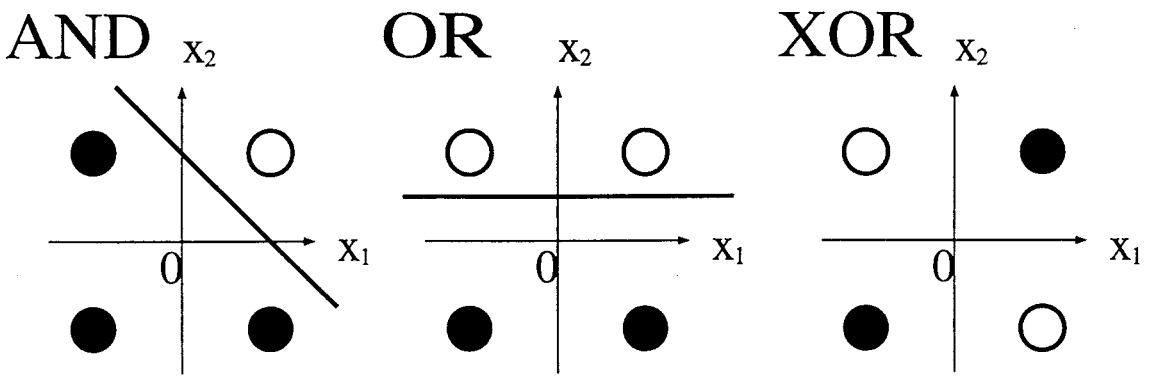

Figure 1.5: The logic of the AND, OR and XOR. For the AND and the OR, we can divide the white circle $(+1)$ and brack circle (-1) by a line. On the other hand, there does not exist such a line for the XOR logic. These circles are located at $(1,1)$, $(1,-1)$, $(-1,1)$ and $(-1,-1)$.

to choose the parameter set (J_1, J_2, θ) so as to satisfy the four conditions in Table 1.4.

In order to understand the above case intuitively, we illustrate the situation in Figure 1.5. From this figure, we see that for the realizable tasks for the perceptron, namely, the AND logic and the OR logic, there exists the line $J_1 x_1 + J_2 x_2 - \theta = 0$ which divides the white circles $(y = +1)$ from the black circles $(y = -1)$. On the other hand, there are no such lines for the XOR logic.

From this result, we are obliged to say that there exists a task which can not be solved by the perceptron. In other words, the power of the perceptron is limited.

However, such types of difficulties were overcome by constructing a different type of network which has intermediate units. Let us consider the neural network which has the following architecture. The network has two intermediate units, each output of which is represented as

$$y_1 = \text{sign}\left(J_1^{(1)} x_1 + J_2^{(1)} x_2 - \theta_1\right) \tag{1.21}$$

$$y_2 = \text{sign}\left(J_1^{(2)} x_1 + J_2^{(2)} x_2 - \theta_2\right). \tag{1.22}$$

The final output of the network is given by the following relation;

$$y = \text{sign}\left(J_1 y_1 + J_2 y_2 - \theta\right) \tag{1.23}$$

We illustrate the structure of the network in Figure 1.6.

The key point of our treatment to solve the XOR logic using this type of the network is translating the difficult XOR logic (x_1, x_2, y) to a much easier logic (y_1, y_2, y) by controlling the parameters of the intermediate units $(J_1^{(1)}, J_2^{(1)}, \theta_1)$, $(J_1^{(2)}, J_2^{(2)}, \theta_2)$ and the parameter of the output unit (J_1, J_2, θ). Namely, we should find the solution for the following Table 1.5. We illustrate the relation between the logic (x_1, x_2, y) and the logic (y_1, y_2, y) in Figure 1.7. From this figure, we see that there exists the line which divides the white circles and the black circles for the case of the logic (y_1, y_2, y). Thus, if we

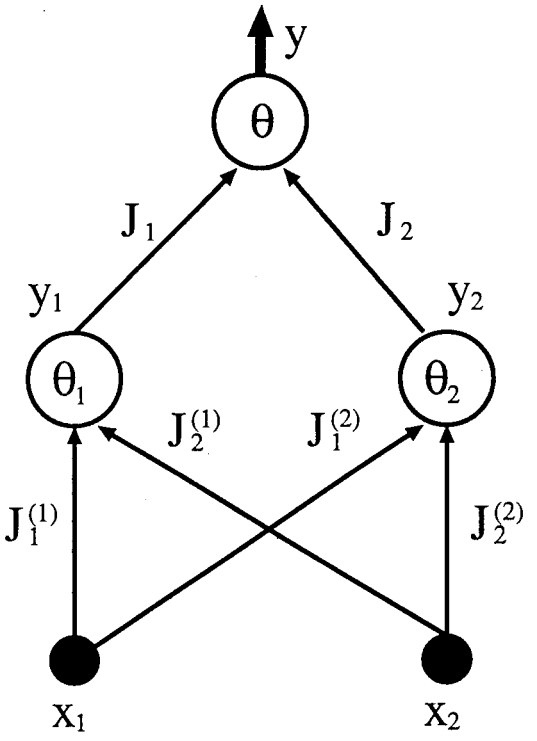

Figure 1.6: The structure of the network which can solve the XOR logic.

x_1	x_2	y	y_1	y_2
−1	−1	−1	−1	−1
−1	+1	+1	+1	−1
+1	−1	+1	+1	−1
+1	+1	−1	+1	+1

Table 1.5: The truth table of the XOR logic (x_1, x_2, y) and the logic which can be solved by a simple perceptron (y_1, y_2, y).

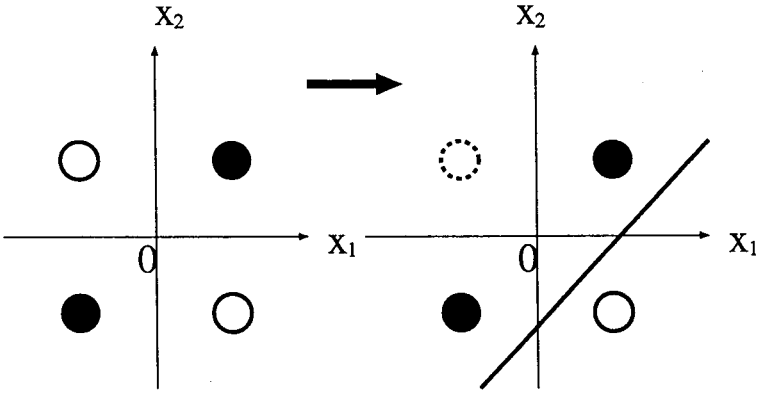

Figure 1.7: The relation between the solvable logic (right) and the unsolvable logic (left).

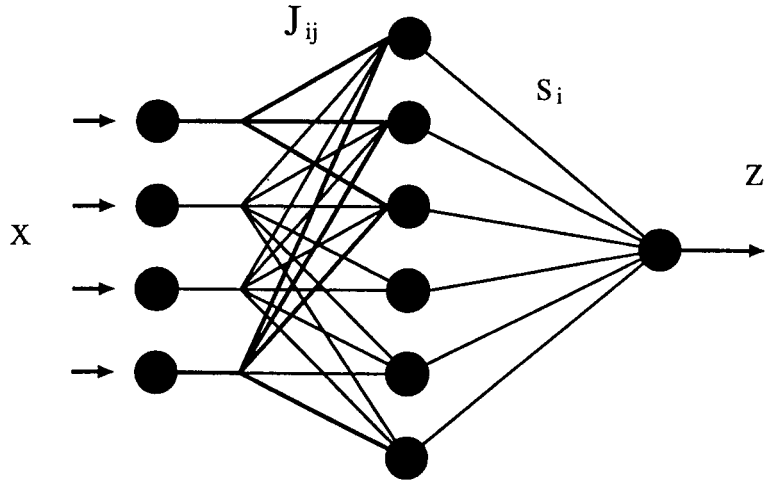

Figure 1.8: A kind of multi-layer perceptrons.

find the set of the parameter $(J_1^{(1)}, J_2^{(1)}, J_1^{(2)}, J_2^{(2)}, J_1, J_2, \theta_1, \theta_2, \theta)$ which satisfies the logic in Table 1.5 with respect to (x_1, x_2, y, y_1, y_2), we can construct the neural network which can solve the XOR logic. After some consideration, we can obtain one of the solutions as $(J_1^{(1)}, J_2^{(1)}, J_1^{(2)}, J_2^{(2)}, J_1, J_2, \theta_1, \theta_2, \theta) = (1, 1, 1, 1, -1, 1, 2)$.

This type of networks is referred to as *multilayer perceptrons*. For the multi-layer perceptrons, the *error back-propagation algorithm* was proposed by Amari [3]. Here we show how the algorithm works. We first consider the network which is illustrated in Figure 1.8. The output z of the machine is given as

$$z = f\left[\sum_{i=1}^{M} s_i f\left(\sum_{j=1}^{N} J_{ij} x_j\right)\right] \tag{1.24}$$

where M and N are the numbers of the intermediate units and input units, respectively. $\boldsymbol{s} = (s_1 \cdots s_N)$ is the weight vector between the output unit and intermediate units, and $\boldsymbol{J} = \{J_{ij}; i = 1, \cdots, M, j = 1, \cdots, N\}$ is the weight vector between inputs and intermediate units. For simplicity, let us represent a set of the weight vectors $\boldsymbol{s}$ and $\boldsymbol{J}$ as $\boldsymbol{\theta} = (\boldsymbol{s}, \boldsymbol{J})$. Then, the input-output relations of the machine are formally represented as

$$z = z(\boldsymbol{x}; \boldsymbol{\theta}). \tag{1.25}$$

The teacher machine is specified by the weight $\boldsymbol{\theta}_{\mathrm{T}}$ and its output is $z(\boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{T}})$. Then, the *loss function* (or *cost function*) is defined as

$$l(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{2}\left[z(\boldsymbol{x}; \boldsymbol{\theta}) - z(\boldsymbol{x}; \boldsymbol{\theta}_{\mathrm{T}})\right]^2. \tag{1.26}$$

If the inputs $\boldsymbol{x}$ are extracted from a distribution $p(\boldsymbol{x})$, the average loss function [4] for one example is defined as

$$L(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) l(\boldsymbol{x}; \boldsymbol{\theta}). \tag{1.27}$$

Using the average loss function, the following strategy may be useful [5];

$$\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - c\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{1.28}$$

[4] The unit which learns so that this type of loss function becomes minimum is called the *AdaLine* (Adaptive Linear Neuron).

[5] This type of update is called the off-line or batch mode.

where c is a small positive constant.

If we define a small time step as Δt, the above algorithm can be written as

$$\boldsymbol{\theta}(t+\Delta t) = \boldsymbol{\theta}(t) - c\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{1.29}$$

After taking a limit of small time step $\Delta t \to 0$, we obtain the next differential equation of the motion for the parameter $\boldsymbol{\theta}$ as

$$\frac{\partial \boldsymbol{\theta}}{\partial t} = -\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{1.30}$$

In order to confirm that this algorithm works effectively, we take the derivative of $L(\boldsymbol{\theta})$ with respect to time t as

$$\frac{\partial L(\boldsymbol{\theta})}{\partial t} = \left(\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)\left(\frac{\partial \boldsymbol{\theta}}{\partial t}\right) = -c\left(\frac{\partial L(\boldsymbol{\theta})}{\partial t}\right)^2 \leq 0 \tag{1.31}$$

where we used (1.30). From this result, we conclude that the loss function $L(\boldsymbol{\theta})$ monotonically decreases according to the learning algorithm. In the above discussion, let us mention the following several points.

Firstly, we assume that the graded-response transfer function $f(x)$ increases monotonically, namely,

$$\frac{df(x)}{dx} \leq 0 \quad \text{for } \forall x. \tag{1.32}$$

This condition guarantees that the loss function monotonically decreases by the error-back propagation learning algorithm (1.28) based on the gradient decent.

Secondly, we assume that the transfer function $f(x)$ is bounded as

$$\lim_{x \to \pm 1} |f(x)| < 1. \tag{1.33}$$

Although we mentioned that the non-linear transfer function has an advantage of the robustness for some input noise, this property of the non-linear transfer function is not suitable for constructing a learning machine based on the error-back propagation learning algorithm. Let us consider that for the intermediate unit with the non-linear transfer function $\mathrm{sign}(x)$, the weight vector changes from $\boldsymbol{J}$ to $\boldsymbol{J} + \Delta\boldsymbol{J}$. Then, the output of the intermediate unit is given by $y \equiv \mathrm{sign}(\boldsymbol{J}\cdot\boldsymbol{x} + \Delta\boldsymbol{J}\cdot\boldsymbol{x})$ and the derivative of this output y with respect to $\boldsymbol{J}$ is represented as

$$\frac{\partial y}{\partial \boldsymbol{J}} = \delta\left(\boldsymbol{J}\cdot\boldsymbol{x} + \Delta\boldsymbol{J}\cdot\boldsymbol{x}\right)\boldsymbol{x} \tag{1.34}$$

where $\delta(x)$ means the Dirac's delta function. Therefore, the modification term appearing in the learning algorithm (1.28) becomes too sensitive only at $\boldsymbol{J}\cdot\boldsymbol{x} = -\Delta\boldsymbol{J}\cdot\boldsymbol{x}$ (∞ sensitivity) and insensitive otherwise (zero sensitivity). As the result, we conclude that the error-back propagation learning algorithm (1.28) does not work well if we choose the non-linear transfer function. Thus, we need the conditions (1.32) and (1.33) for the error-back propagation learning algorithm. One of the suitable candidates for the graded-response transfer function is, for example,

$$f(x) = \frac{1}{1+\exp(-\beta x)} \tag{1.35}$$

where β is a control parameter which has an analogy with the temperature as $\beta = T^{-1}$ and the above function goes to the signature function in the limit of $\beta \to \infty$.

Thirdly, it is hard to calculate the modification term appearing in (1.28) because $L(\boldsymbol{\theta})$ depends on every example on $p(\boldsymbol{x})$. In other word, at each step, we should check whether the output of the student coincides with that of the teacher or not for all examples. Therefore, we use the next strategy [3] as a substitute of (1.28) [6];

$$\boldsymbol{\theta} \to \boldsymbol{\theta} - c\frac{\partial l(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{1.36}$$

[6] This type of update is called the on-line mode.

According to the distribution $p(\boldsymbol{x})$, the above modification works stochastically and the algorithm is referred to as the *stochastic descent algorithm* [3]. The speed and accuracy of the convergence of the parameter θ were investigated in detail by Amari [3].

For a three layer perceptron (Figure 1.8), the derivatives $\partial l/\partial s_i$ or $\partial l/\partial J_{ij}$ are calculated as

$$\frac{\partial l}{\partial s_i} = r_i y_i \tag{1.37}$$

$$r_i \equiv \{z(\boldsymbol{x}) - z_{\mathrm{T}}(\boldsymbol{x})\} f'(\boldsymbol{s}\cdot\boldsymbol{y}) \tag{1.38}$$

$$\frac{\partial l}{\partial J_{ij}} = \frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial J_{ij}} = \tilde{r}_i x_j \tag{1.39}$$

$$\tilde{r}_i \equiv r_i s_i f'(\boldsymbol{J}\cdot\boldsymbol{x}). \tag{1.40}$$

Using these results, the parameters $\boldsymbol{s}$ and $\boldsymbol{J}$ update as follows.

$$s_i \rightarrow s_i - c r_i y_i \tag{1.41}$$

$$J_{ij} \rightarrow J_{ij} - c\tilde{r}_i x_j. \tag{1.42}$$

From the update rules (1.41) and (1.42), the error of output $z(\boldsymbol{x}) - z_{\mathrm{T}}(\boldsymbol{x})$ propagates from the output unit to the input units. Rumelhart *et al* [4] or other researchers [5, 6] re-discovered or re-formulated the above algorithm and called it the *delta rule* because δ corresponds to r_i or $\tilde{r}_i$ in their notations .

The choice of the learning rate c is also an important problem for the learning algorithm. The learning rule (1.36) updates stochastically every time the student receives an input $\boldsymbol{x}$, and $\boldsymbol{\theta}(t)$ (t means time) does not converge to a specific value $\boldsymbol{\theta}(t\rightarrow\infty) = \boldsymbol{\theta}_0$ as long as the learning constant c is finite. As the result, $\boldsymbol{\theta}(t)$ fluctuates around some value. We should notice that if we consider the ensemble of the N ($N\rightarrow\infty$) machines in which each machine updates according to the rule (1.36), the mean value of the parameter $< \boldsymbol{\theta}(t) >$ with respect to the distribution of the ensemble converges to $\boldsymbol{\theta}_0$ [72] [7]. However, as long as we consider the learning processes of only one machine, the convergence of the parameter is not guaranteed. If we would like to obtain the convergence of the parameter $\boldsymbol{\theta}(t)$ for one machine, we should introduce the time dependent learning rate c_t instead of constant c. In addition, the learning rate should satisfy the following two conditions [73];

$$\lim_{t\rightarrow\infty} c_t = 0 \tag{1.43}$$

and

$$\sum_{t=0}^{\infty} c_t = \infty. \tag{1.44}$$

The meaning of these two conditions are intuitively understood as follows. The first condition guarantees that after sufficient long time, the modification term goes to zero, the parameter $\boldsymbol{\theta}$ converges to some value. On the other hand, as the second condition means that the sum of the modification terms diverges to infinity, the second condition (1.44) guarantees that the parameter $\boldsymbol{\theta}$ can reach any place of the whole parameter $\boldsymbol{\theta}$ space after infinite time steps.

One of the candidates for learning rate c_t which satisfies the above two conditions (1.43) and (1.44) is $c\sim 1/(t_0+t)^{\gamma}$ ($\gamma\geq 1$) and this rate behaves $c\sim t^{-\gamma}$ in the limit of $t\rightarrow\infty$ [8].

One of the remarkable applications of the back propagation learning algorithm is the *NET-talk* by Sejnowski and Rosenberg [7]. They succeeded in training a neural network by the back propagation algorithm so that the trained network produces correct phoneme codes from corresponding English text character codes. They assumed that the pronouncement for a specific character which is an element of the word is determined by the three characters in front and behind of the character. For example, the character *W* appearing in the word *NETWORK* should be determined by *NET* and *ORK*. Taking this assumption into account, they tried to construct the neural network which is illustrated in Figure 1.9 and make this machine learn the rule of the pronouncement. This neural network consists of the input layer,

[7] The deviation of the parameter with respect to the distribution of the ensemble $< (\theta(t) - < \theta(t) >)^2 >$ is also guaranteed to converge to some value.

[8] This type of time dependent learning rate is referred to as the $t^{-\gamma}$-annealing.

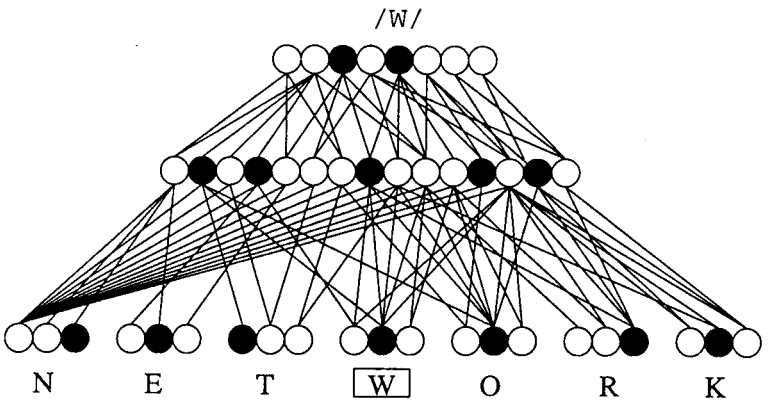

Figure 1.9: The learning machine *NET-talk.*

the intermediate layer and the output layer. The input layer has 7 units and each of the unit contains 29 neurons which correspond to 26 alphabets and ",", ".", "?" . Each neuron takes +1 if they coincides with one of the seven words and takes −1 otherwise. Therefore, an input signal consists in 29×7 = 203 neurons, and among them, only 7 neurons $N{\cdot}E{\cdot}T\cdot W{\cdot}O\cdot R{\cdot}K$ take +1 and the rest takes −1 [9] and the output signal means the pronouncement /w/ for the word W as the input. After 1024 words of English (inputs) and phoneme (outputs) are presented as examples, the network changed its own structure so that the network pronounces 95% of the examples correctly. In addition, they gave 10,000 new examples which did not belong to the training examples to the trained network. Surprisingly, the trained network produced correct answers for 85% of new examples. This result implies that the network can construct the complicated representations of information which is beyond our expectations by learning from a finite set of examples. Importance of their result has been increased not only because they showed a successful application of the back propagation learning algorithm but also because their result motivated a lot of researchers to investigate to what extent the network can generalize.

1.3.3 Boltzmann machine

For the recurrent networks (see Figure 1.10), Hinton and Sejnowski [8] introduced the *Boltzmann machine algorithm.* In general, the Boltzmann machine consists of three parts, namely, input units, output units and hidden units. In Figure 1.10, we represent the states of input, output and hidden units by $\boldsymbol{x}_{\mathrm{I}}$, $\boldsymbol{x}_{\mathrm{O}}$ and $\boldsymbol{x}_{\mathrm{H}}$, respectively [10] . For simplicity of the following discussions, let us denote a set of the three kinds of state by $\boldsymbol{x}$, namely, $\boldsymbol{x} = (\boldsymbol{x}_{\mathrm{I}}, \boldsymbol{x}_{\mathrm{O}}, \boldsymbol{x}_{\mathrm{H}})$. Then, the state of the the machine updates according to the following probability;

$$g(x_i = +1) = \frac{1}{1 + \exp(-2\beta h_i)} \tag{1.45}$$

where $h_i = \sum_j J_{ij}x_j$ and β is the inverse temperature given as $\beta = 1/T$. It is important for us to bear in mind that the Boltzmann machine has the next two remarkable features;

- Each unit updates *asynchronously.*
- The weight matrix is *symmetric*, namely, $J_{ij} = J_{ji}$ for all i,j.

Although the symmetric synaptic weight seems to be in contradiction with the function of synaptic efficacy in the real brain, this feature guarantees that the stochastic process (1.45) produces the following

[9] This type of the information cording in which the ratio of the +1 neurons to the −1 neurons is extremely small is called the *sparse coding.*

[10] The role of the hidden unit is to help the visible unit (the input-output unit) to realize the difficult task for the network. This role is similar to that of the units in the intermediate layer of the feed-forward neural network.

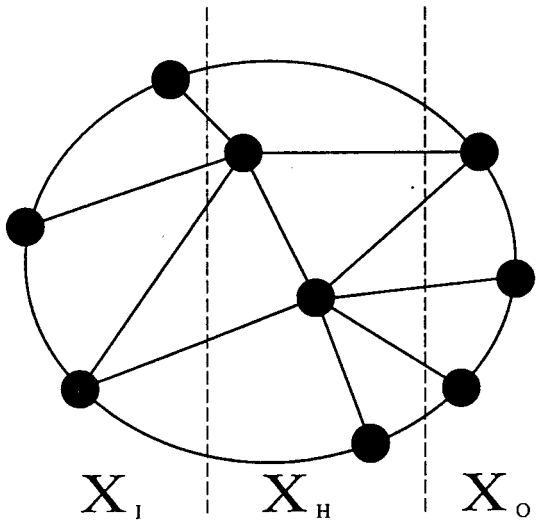

Figure 1.10: Conceptual figure of recurrent neural network.

equilibrium distribution with respect to the network state $\boldsymbol{x}$.

$$p(\boldsymbol{x}) = \frac{\exp[-\beta H(\boldsymbol{x})]}{Z} \tag{1.46}$$

where $z \equiv \sum_{\boldsymbol{x}} \mathrm{e}^{-\beta H}$ and H is the following *Lyapunov function*

$$H(\boldsymbol{x}) = \frac{1}{2}\sum_{i,j} J_{ij}x_i x_j. \tag{1.47}$$

If the local minima of $H(\boldsymbol{x})$ is located at $\boldsymbol{x}_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}_c$, these three states are inclined to appear and each probability for the appearance is proportional to $p(\boldsymbol{x}_a)$, $p(\boldsymbol{x}_b)$ and $p(\boldsymbol{x}_c)$. For this network, if the weight matrices J_{ij} change, the equilibrium distribution of the state $p(\boldsymbol{x})$ also changes. Then, the problem which we would like to solve by the above Boltzmann machine is what the best strategy is for us to modify the weight J_{ij} in order to bring the distribution $p(\boldsymbol{x})$ close to the given distribution $q(\boldsymbol{x})$. The distribution $q(\boldsymbol{x})$ means the environment and the machine learns so as to adapt itself to the environment. Then, the student modifies his weight matrix J_{ij} so as to approximate the given distribution $q(\boldsymbol{x})$ by $p(\boldsymbol{x})$ as correctly as possible.

We should notice that as each element $x_i (i = 1, \cdot, N)$ of the vector $\boldsymbol{x}$ produced by the distribution $q(\boldsymbol{x})$ takes a binary value ± 1 and $\boldsymbol{x}$ has a constraint $\sum_{\boldsymbol{x}} q(\boldsymbol{x}_a) = 1$, the maximum number of the possible configurations is $2^N - 1$. On the other hand, the the number of configurations of $\boldsymbol{x}$ which can be realized by the Boltzmann machine with the weight matrices J_{ij} is equal to the number of the elements of matrices of J_{ij}, namely, $N(N-1)/2$. As the result, our goal is not to obtain the distribution $q(\boldsymbol{x})$ exactly but to approximate the distribution $q(\boldsymbol{x}_a)$ as precisely as possible.

In order to measure the distance between $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, we introduce the next *Kullback-Leibler distance (KL divergence)*;

$$D(p\,;q) = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}. \tag{1.48}$$

We easily see that the KL divergence is a kind of the cost function and becomes zero if $p(x)$ coincides with $q(x)$.

Then, the Boltzmann machine learning algorithm is given as

$$J_{ij} \quad \rightarrow \quad J_{ij} + c\frac{\partial D(p\,;q)}{\partial J_{ij}}$$

$$\begin{aligned} &\rightarrow \quad J_{ij} + c\beta[< x_i x_j >_q - < x_i x_j >_p] \\ &\rightarrow \quad J_{ij} + c\beta[x_i x_j - < x_i x_j >_p], \end{aligned} \tag{1.49}$$

where $< \cdots >_p$ or $< \cdots >_q$ denote the average over the distribution $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$. We should notice that if $\boldsymbol{x}$ is extracted from $q(\boldsymbol{x})$, $x_i x_j$ becomes $< x_i x_j >_q$ on average. The term $- < x_i x_j >_p$ corresponds to the Hebbian un-learning in the associative memory. Using (1.49), we can modify the weight of the network so that the network puts the distribution of output $p(\boldsymbol{x})$ close to $q(\boldsymbol{x})$. It is clearly understood that if the distribution of the network $p(\boldsymbol{x})$ becomes the distribution of the environment $q(\boldsymbol{x})$, namely, $p(\boldsymbol{x}) = q(\boldsymbol{x})$, the modification term in (1.49) goes to zero and the machine stops learning. However, we need the equilibrium distribution to calculate the update rule (1.49) every time the weights are modified. Unfortunately, the time to reach the equilibrium state of J_{ij} is very long in general. Therefore, in order to obtain the equilibrium distribution as quickly as possible, we should control the temperature $T(t)$ in (1.45) effectively. The scheduling of the temperature is not only a problem of machine learning but also problems in the optimization by *simulated annealing* [9].

1.4 What is generalization ?

1.4.1 PAC learning

These algorithms we introduced in the previous section succeeded and have been applied to lots of problems in engineering. However, if the size of the networks becomes large and the machines have complicated structures, it is impossible to present all of the examples to the student network. In general, the student can not refer to all examples ($(y^P, \boldsymbol{x}^P)$, $P = \infty$) which is produced by the teacher. Therefore, the student should *estimate* or *predict* the structure of the teacher from a finite set of examples ($(y^P, \boldsymbol{x}^P)$, $P < \infty$) as correctly as possible. Actually, as we mentioned in the previous section, Sejnowski and Rosenberg [7] showed that the network which was trained by a finite number of examples has an ability to answer the new examples to some degree.

As the result, the goal of learning has been shifted from identifying the structure of the teacher network *perfectly* to estimating the teacher network *approximately*. This concept of the machine learning, namely, approximating the parameters of teacher, instead of identifying them perfectly, is called the *probably almost correct* (PAC) learning [11] [10].

1.4.2 Generalization error and learning curve

The point of machine learning is that the quality of a given learning algorithm should be evaluated in terms of how often the student makes mistakes for new examples (what we call the *test sets*), which do not belong to the training set, after learning a finite set of examples.

We call the abilities of the student networks, namely, the power of answering the new questions correctly the *generalization abilities*. It is important for us to bear in mind that this generalization ability should be distinguished from the training ability which is the ability of answering the training set only.

In practice, the most natural measure for generalization abilities is the *generalization error* which is the probability of disagreement between the teacher output and the student output for a new example.

As this probability can be regarded as a function of the number of examples, we call this relation between the generalization error and the number of examples the *learning curves*. For a rather specific situation in which the structure of the student and the teacher is same, the distance between the synaptic weight of the teacher (what we call the *target*) and the weight of the student is reflected to the generalization error. If the student vector becomes to coincide with the target, the generalization error becomes zero. We call the weight space of the student which shrinks to some value the *version space*. We illustrate a typical example of version space in Figure 1.11.

[11] Some people call PAC learning as an abbreviation of *probably approximately correct* learning.

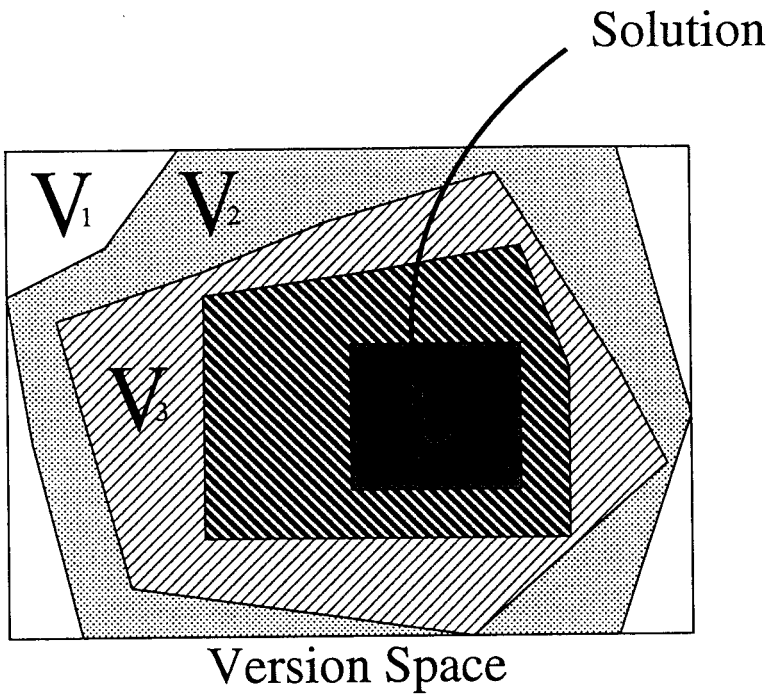

Figure 1.11: Conceptual figure of the version space. In general, the version space shrinks ($V_1 \to V_2 \to V_3 \to \cdots$) according to some specific learning rule. When the student finds a solution, the volume of the version space becomes zero if the task of the teacher is realizable. How fast the version space shrinks depends on the learning algorithm.

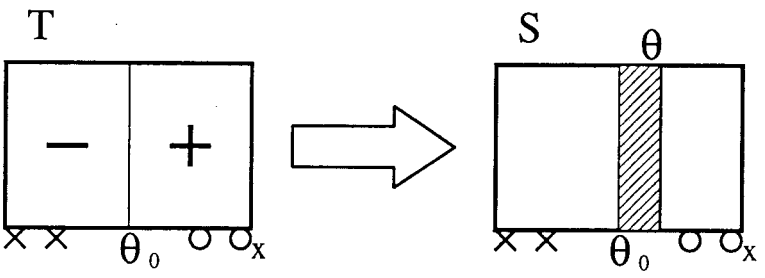

Figure 1.12: Learning of a decision boundary. The student network who estimates the parameter $\theta\,(\neq\theta_0)$ makes a mistake for an examples falling in the shaded area. As the distance between the nearest neighbor of examples is $\mathcal{O}(P^{-1})$, the generalization error behaves asymptotically as $\epsilon_g \sim P^{-1}$.

1.4.3 Learning a threshold parameter

In Figure 1.12, we show a simple case of machine learning. Let us consider that the teacher and the student have the same structure and they are specified by their threshold parameters θ_0 (teacher) and θ (student). The inputs x are extracted from a one-dimensional uniform distribution within $[0, 1]$. The output of the machines is $+1$ if the number x is bigger than the threshold and -1 otherwise. Let us suppose that the student estimates the parameter as $\theta\,(\neq\theta_0)$ after training from a finite number of examples. Then, the student makes a mistake for a new example which falls on the shaded area in Figure 1.12. Therefore, there exists a probability of disagreement between the student and teacher as long as the shaded area exists. The generalization error ϵ_g for this case of learning the decision boundary θ decays to zero asymptotically ($P\rightarrow\infty$, P : number of examples) as

$$\epsilon_g \sim P^{-1}. \tag{1.50}$$

This result is easily understood because the error of the student is proportional to the distance between arbitrary two inputs $\sim 1/P$.

1.4.4 Vapnik-Chervonenkis bounds for generalization

Baum and Haussler [11, 12] introduced the *Vapnik-Chervonenkis (VC) dimension* [13] into the framework of PAC learning and estimated to what degree the error for a new example can be small after P examples are given. They discussed the case in which a single layer perceptron without threshold (1.12) learns from a single perceptron of the same type. In this subsection, we briefly review the VC theory for the learning of a simple perceptron. Let us consider an ensemble $\Lambda(\boldsymbol{J})$ of the simple perceptrons. Moreover, consider an unspecified teacher perceptron with the weight vector $\boldsymbol{B}$. As we mentioned in the previous subsection, the purpose of the learning is to approximate the teacher $\boldsymbol{B}$ as precisely as possible with a simple perceptron $\boldsymbol{J} \in \Lambda(\boldsymbol{J})$ on the basis of a training set which is represented by the teacher perceptron $\boldsymbol{B}$. For every perceptron $\boldsymbol{J}$ of $\Lambda(\boldsymbol{J})$, one can evaluate the training error $\epsilon_t(P)$ which is defined as the number of training examples for which the output of the student disagrees with the teacher output. Another quantity of importance is the generalization error ϵ_g of the student, defined as the probability of disagreement with the teacher on a randomly chosen example from the teacher. It is obvious that the average value of the training error is precisely the generalization error, namely, $< \epsilon_t(P) >= \epsilon_g$.

The key point of the Vapnik-Chervonenkis theory is to evaluate how the fluctuations of the training error $\epsilon_t(P)$ around the generalization error ϵ_g decreases as the number of training examples P increases (see Figure 1.13). The VC theory provides a bound for the probability that a training set is chosen such that the maximum deviation between $\epsilon_t(P)$ and ϵ_g exceeds a threshold value $\varepsilon > 0$, namely,

$$\text{Prob}\left(\sup_{\boldsymbol{J}\in\Lambda(\boldsymbol{J})}|\epsilon_t(P) - \epsilon_g| > \varepsilon\right) \leq 4m(2P)\,\mathrm{e}^{-\varepsilon^2 P/8} \tag{1.51}$$

where $m(P)$ is defined as the maximum number of different classifications induced on P patterns by a simple perceptron $\boldsymbol{J}$. In the condition (1.51), we consider the maximum difference between the training error and the generalization error over all perceptrons $\boldsymbol{J} \in \Lambda(\boldsymbol{J})$. In other words, we choose the worst student for every particular choice of a training set. As the result, the VC-bound (1.51) is valid for any student.

Although we have to estimate the $m(P)$ in order to calculate the VC-bound (1.51), the explicit result was obtained by Cover [14]. His result is represented as follows.

$$m(P) = \begin{cases} 2^P & \text{if } P \leq N \\ 2\sum_{k=0}^{N-1} {}_{P-1}C_k & \text{if } P \geq N \end{cases} \tag{1.52}$$

It is important for us to bear in mind that all the possible classifications can be implemented for $P \leq N$, but not so for $P > N$. This N is referred to as the *VC-dimension* $d_{\rm VC}$ for the class of a simple perceptron.

From (1.52), one obtains the following condition;

$$m(P) \leq P^{d_{\rm VC}} + 1 = P^N + 1. \tag{1.53}$$

Using this condition, we can estimate the VC-bound (1.51) as follows.

$$\text{Prob}\left(\sup_{\boldsymbol{J} \in \Lambda(\boldsymbol{J})} |\epsilon_t(P) - \epsilon_g| > \varepsilon\right) \leq 4\left[(2P)^N + 1\right] \mathrm{e}^{-\varepsilon^2 P/8} \sim 4\exp\left(N\log(2P) - \varepsilon^2 P/8\right). \tag{1.54}$$

If we assume that the above upper bound $4\exp(N\log(2P) - \epsilon^2 P/8)$ is smaller than a small positive constant δ, namely, $4\exp(N\log(2P) - \epsilon^2 P/8) < \delta$, the following condition on N is obtain.

$$N < \frac{\epsilon^2 P - 8\log(4/\delta)}{8\log(2P)} \tag{1.55}$$

In the limit of $P, N \to \infty$, we finally obtain the condition on P as

$$P > \frac{8N\log N}{\epsilon^2}. \tag{1.56}$$

This result means that in order to make the error smaller than ϵ, one needs at least P examples which are larger than $8N\log N/\epsilon^2$. We should notice that this is the result of the *worst case analysis* and the above estimation is nothing but a lower bound for the generalization ability. Recently several authors succeeded in deriving the condition for the lower bound (1.56) by calculating the training error and the generalization error using the replica method. [15, 16].

1.4.5 Unrealizable task

The situation in which the structure of the student agrees with that of the teacher is rather ideal. In the real world, students and teachers have different constructions. Therefore, it is of great importance for us to investigate such cases. For machine learning, it is important to improve the learning scheme and minimize the generalization error even if it is impossible to exactly reproduce input-output relations of the teacher.

If the student is inferior to the teacher, the rule (or task) of the teacher is referred to as an *unrealizable rule* in opposition to a *realizable rule*. On the other hand, if the student is superior to the teacher, the teacher's rule is called the *over-realizable rule*.

We should not forget that the task of the teacher becomes unrealizable also due to some noises. In order to tell the former case from the latter, one sometimes refers to the former case as unrealizable rule due to *structural mismatch*.

For example, if the teacher output is reversed with some probability (which is referred to as an *output noise*) or if the input signal contains some noises (which is referred to as an *input noise*), the task of the teacher becomes unrealizable.

For the output noise, the generalization error asymptotically decays to the minimum value $\epsilon_{\rm min}$ as

$$\epsilon_g \sim \epsilon_{\rm min} + P^{-1} \tag{1.57}$$

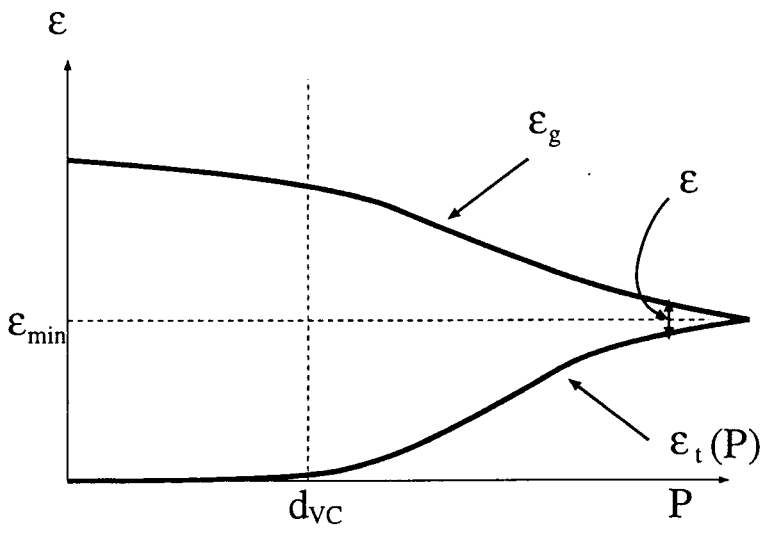

Figure 1.13: The fluctuation of the training error around the generalization error.

and

$$\epsilon_g \sim \epsilon_{\min} + P^{-2/3} \tag{1.58}$$

for the input noise [17, 18, 19, 20].

In contrast, the behaviors of the generalization error for the structural mismatch cases have been open questions. Only a few papers have appeared concerning learning of unrealizable rules for the structural mismatch cases, on the soft committee machine [21] and the problem for small number of examples [22]. This is because the method of the traditional mathematical statistics, which has been a main tool for parameter estimations, is hard to be applicable to the case of unrealizable tasks. In addition, it is difficult for us to apply the VC theory to the problem of unrealizable classes.

In this thesis, we investigate the problem of the structural mismatch cases using the statistical mechanical approach which was first introduced by Levin, Tishby and Solla [23]. Györgyi and Tishby [24] applied this method to the learning problem of neural network model. The statistical mechanical approach [12, 26, 27] is applicable even to the cases of the unrealizable rule.

As a typical situation of the structural mismatch problems, we take up the teacher and the student machines as follows. The student machine is a single layer perceptron whose synaptic weight vector is specified as $\boldsymbol{J} = (J_1, \cdots, J_N) \in \Re^N$ and his output S is represented as $S(u) = \mathrm{sign}(u)$ with $u \equiv \sqrt{N}(\boldsymbol{J}\cdot\boldsymbol{x})/|\boldsymbol{J}|$. On the other hand, the teacher is also a single layer perceptron with the coupling vector $\boldsymbol{B} = (B_1, \cdots, B_N) \in \Re^N$ but his transfer function (or activation function) is non-monotonic (or reversed wedge type) as $T_a(v) = \mathrm{sign}[v(a-v)(a+v)]$ with $v \equiv (\boldsymbol{B}\cdot\boldsymbol{x})/|\boldsymbol{B}|$ (see Figure 1.14).

The variables u or v are called the *internal potentials* or the *local fields*. We should notice that the structure of the student machine becomes to agree with the teacher when the width of the reversed wedge a goes to ∞.

We would like to emphasize that the output of the teacher's machine corresponds with that of a kind of a multi-layer perceptron which is usually called the $K = 3$ *parity machine*. This parity machine has three hidden units and each unit consists of a single layer perceptron with couplings $\boldsymbol{B} = (B_1, \cdots, B_N) \in \Re^N$ and each transfer function is represented as $y_1 = \mathrm{sign}(a - v)$, $y_2 = \mathrm{sign}(-v)$ and $y_3 = \mathrm{sign}(-a - v)$. The final output of the machine is given as a product of the outputs of the three hidden machines, $y_1 y_2 y_3$ (see Figure 1.15). This type of the machine has stronger abilities than the machines with a conventional monotonic transfer function [28, 29, 31]. In addition, for the recurrent type of the networks with symmetric weights, instead of the feed-forward networks, several authors have reported that the non-monotonicity of the transfer function improves the retrieval properties of the networks [32, 33, 37].

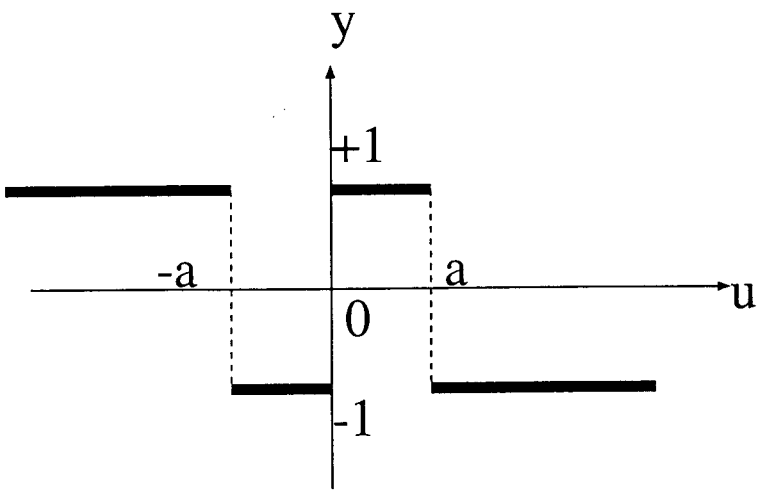

Figure 1.14: The non-monotonic transfer function. If the width of the reversed wedge a goes to ∞, the function corresponds to the monotonic one and the task of the teacher becomes realizable.

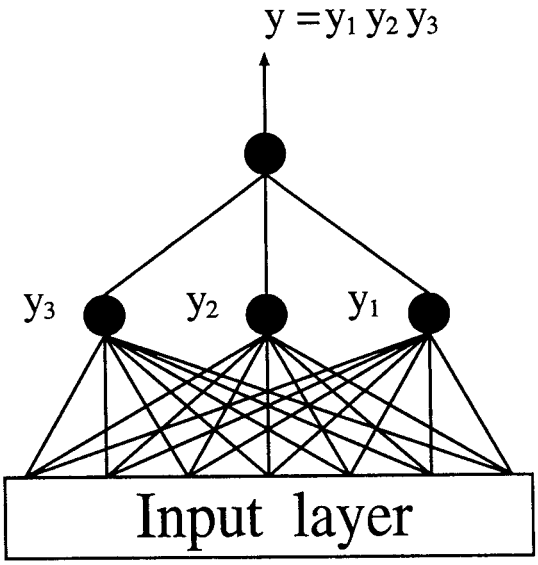

Figure 1.15: The $K = 3$ parity machine. The outputs of the three hidden units are $y_1 = \text{sign}(a - v)$, $y_2 = \text{sign}(-v)$ and $y_3 = \text{sign}(-a - v)$. The final output of this machine is given as $y_1 y_2 y_3$.

In our learning system, a single layer perceptron as a student learns from a multi-layer perceptron as a teacher who has greater abilities than the student. In general, it is hard to treat analytically the learning systems in which the machines belong to a class of multi-layer perceptrons. However, in our model system, as we treat a kind of maps for multi-layered perceptrons, it is easy for us to carry out the analysis.

1.4.6 Two modes of machine learning

In this thesis, we investigate our system according to the following two types of learning modes. The first one is referred to as the *off-line learning* or the *batch learning*. In this off-line learning mode, the student receives a finite number of examples at the same time and changes his own weight vector after learning the whole set of examples perfectly (memorize) or until he reaches some error tolerance. On the other hand, the other one is referred to as the *on-line learning*. In the on-line learning, the student changes his own weight using only the recent example. Therefore, the student does not have to memorize all the examples and one can save not only the training times but also memories on the computer hardware. However, if we devoted ourselves to improving the speed of training, the accuracy of the result is inclined to become worse. Therefore, we should improve the on-line learning algorithms. In this thesis, we investigate the generalization ability for the structural mismatch case according to the above two learning modes.

1.5 Optimization problem

We saw that the back propagation algorithm has succeeded and many researchers have improved the algorithm so that the network can obtain the high generalization ability as fast as possible. However, there is an essential difficulty in the back propagation algorithm. If the cost function we want to minimize has a simple structure, namely, the landscape of energy has a single valley, it is easy to find the minimum by the back propagation algorithm. Unfortunately, for a lot of practical problems we would like to solve by neural network, the cost function has lots of minima. Therefore, as the back propagation is based on the gradient descent, learning may stop in one of the local minima. This is referred to as the *problem of local minima*. This problem of local minima is not only the problem of the back propagation learning but also the difficulty for many other research fields appearing in physics, chemistry, operations research and so on. In these problems, our goal is minimizing the cost function (or the energy function, the objective function, *etc.*). Usually, these cost functions have very complicated structures and have a various number of local minima. These problems are specified by three classes of the difficulties to solve. In the next section, we explain these three classes by taking up a simple combinational optimization problem.

1.5.1 The NP, P and NP-complete problems

If the dimension of the solution space is N, the number of candidates of a solution (number of local minima) is of order $\sim e^N$ or $N!$. Among these problems, if one can find an algorithm to solve the problem within a time of polynomial order with respect to system size N using the deterministic algorithm, we refer to this problem as *polynomial (P)*. This problem P is a sub-class of the problem, named the *Non-deterministic polynomial (NP)*. We call the problem *NP* if the problem can be solved by the non-deterministic algorithm within a time of polynomial order. In the problem NP, there is the most difficult problem to solve, what we call, *NP-complete*. The problem of NP-complete can be solved by the non-deterministic algorithm within the polynomial time. Therefore, this problem of NP-complete belong to the class of NP. However, for this problem of NP-complete, we can not find the algorithm which can solve the problem faster than the polynomial time by the deterministic algorithm.

Let us make the intuitive concepts of the NP, P and NP-complete more clear. For this purpose, we take up the so-called *knapsack problem*. The knapsack problem we examine is defined as follows.

The knapsack problem

The positive real variables $b, a_1, a_2, \cdots a_N$ are given. Then the goal of the problem is to find a sub-sum of $a_1, a_2, \cdots a_N$ which satisfies $\sum_{i \in A} a_i = b$, $A \subseteq \{1, \cdots N\}$.

For this problem, it is obvious that there exist 2^N ways for the sub-sum of $a_1, \cdots a_N$ (each of the sub-sums is determined according to whether the sub-sum contains a_i $(i = 1, \cdots, N)$ or not). Therefore, for the worst case, we need the steps of $\mathcal{O}(2^N)$ in order to find the solution.

If there exists an algorithm to solve the problem within $\mathcal{O}(2^{N^k})$ (k:constant) time steps, we can say that the problem belongs to the class of the *EXPTIME (exponential time)*. In contrast, if there exists an algorithm to solve the problem within $\mathcal{O}(N^k)$ time steps, the problem belongs to the class, which is called the *P (deterministic polynomial time)*. We should notice that $P \subseteq EXPTIME$.

Although the knapsack problem can not be solved within a polynomial time by the deterministic algorithm, it becomes possible for us to solve the problem within a polynomial time if we prepare at most 2^N calculating machines and each of which calculate the problem by the following algorithm.

The non-deterministic algorithm for the knapsack problem

(1) sum←0
(2) **INPUT** b
(3) **WHILE** $\Omega_a \equiv \{a_1, \cdots, a_N\} \neq \emptyset$ **DO**
(4) **INPUT** $a_k \in \Omega_a$
(5) **EITHER** sum←sum + a_k **OR** sum←sum
(6) **IF** b = sum **THEN**
(7) **END**

It important for us to bear in mind that the maximum number of the steps to proceed with the instructions (3) ~ (6) appearing in the above algorithm is $\mathcal{O}(N)$. Therefore, we can confirm that at least one of the 2^N machine can obtain the solution of the knapsack problem within the $\mathcal{O}(N)$ time steps using this parallel algorithm. The key point of the above algorithm is the *parallel calculations* by the $\mathcal{O}(2^N)$ independent machines.

This procedure is also understood intuitively as follows. At first, a machine proceeds the **EITHER-OR** instruction in the above algorithm. Then, the machine produces the new two machines, namely, the machine which is made by the procedure sum←sum + a_k, and another machine is given by the procedure sum→sum. Next, each of these two machines proceed the **EITHER-OR** instruction, as the result, each of the two machines produces the new two machines. Therefore, the whole procedure forms the *tree structure* of the machines which is illustrated in Figure 1.16. In this figure, we see that although the number of the deepest *descendants* is 2^N, the number of *generation* is N. Therefore, we conclude that the most excellent machine can find the solution of the problem within $\mathcal{O}(N)$ steps. This algorithm is referred to as the *non-deterministic algorithm*. In contrast, the algorithm in which only one machine calculates at most all 2^N candidates is called the *deterministic algorithm*. We can say that the knapsack problem is solvable within a polynomial time if we use the *non-deterministic algorithm*, although the problem can not be solved within a polynomial time by the *deterministic algorithm*. Therefore, the problem which can be solved within a polynomial time by the non-deterministic algorithm is referred to as the NP (the non-deterministic polynomial). It is obvious that the P is a subset of the NP, namely, P⊆NP.

Next, we consider the transformation machine. This machine produces new input $f(x)$ from the input x (for example, $b, a_1, \cdots a_N$ in the knapsack problem) within the time steps of the polynomial of $|x|$, where $|x|$ denotes the size of input (for example, N in the knapsack problem). This function f is referred to as the *polynomial time computable function*. If the necessary and sufficient condition for the existence of a solution for the problem A in the input x space corresponds to the necessary and sufficient condition for the existence of a solution for the problem B in the new input space $f(x)$, the relation between the problem A and the problem B is referred to as that *the problem A is polynomial time reducible to the problem B*.

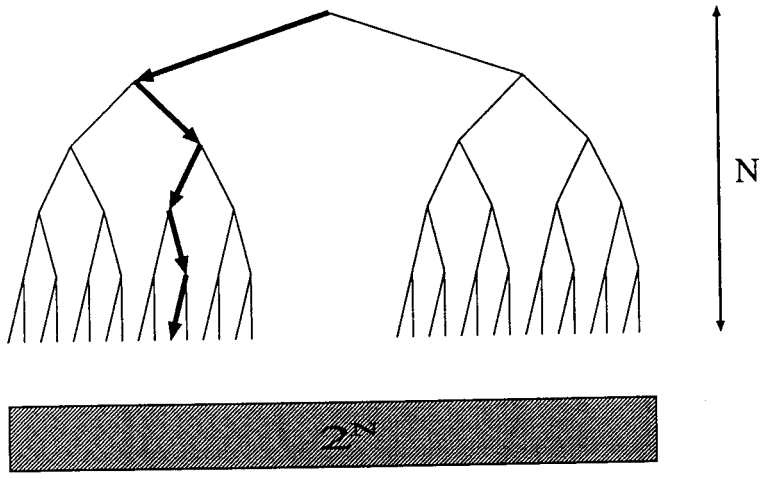

Figure 1.16: The tree structure. The number of the deepest descendants of the tree is 2^N and the number of generations is N. From this figure, we see that the parallel algorithm can solve the problem within $\mathcal{O}(N)$ steps.

Then, we can say that if the problem A is polynomial time reducible to the problem B and we can solve the problem B by the deterministic algorithm within a polynomial time (namely, solvable by P), the problem A also can be solved by P. Therefore, we can say that the problem B is more difficult than the problem A, because if we can solve the problem B within a polynomial time steps, the problem A is guaranteed to be solved by P.

We show that this statement is correct as follows. As the problem A is polynomial time reducible to the problem B, there exists the polynomial time computable function f and the necessary and sufficient condition for the existence of a solution for the problem A in the original input x space corresponds to the necessary and sufficient condition for the existence of a solution for the problem B in the translated input $f(x)$ space. As we mentioned, the function $f(x)$ is calculated from x within the polynomial time steps of $|x|$, there exists a constant k and the following condition holds.

$$|f(x)| \leq |x|^k \tag{1.59}$$

On the other hand, as the problem B can be solved by the P, we can check whether the problem B has a solution in the input space $f(x)$ or not within the polynomial time with respect to the size of the input space $|f(x)|^l$, where l is a constant. As the result, if there exists a solution for the problem B in the input space $f(x)$, there also exists a solution for the problem A in the input space x. In contrast, if we can not find a solution for the problem B in the input space $f(x)$, we also can not obtain a solution for the problem A in the input space x. The above decision takes

$$|f(x)|^l \leq |x|^{kl} \tag{1.60}$$

time steps, that is, we check the existence of the solution for the problem A within the polynomial time of $|x|$. Summarizing the above argument, we can conclude that the problem A can be solved by the polynomial time step of size of input space $|x|$.

Next, let us suppose that there are three problems A, B and C. If the problem A is polynomial time reducible to the problem B, and the problem B is polynomial time reducible to the problem C, the problem A is polynomial time reducible to the problem C. Therefore, the relation of the problem A, B and C is translational to one another.

On the bases of the above arguments, we can say that if a problem X is the NP-complete, the problem X should satisfy the following two conditions.

- The problem X can be solved by NP.

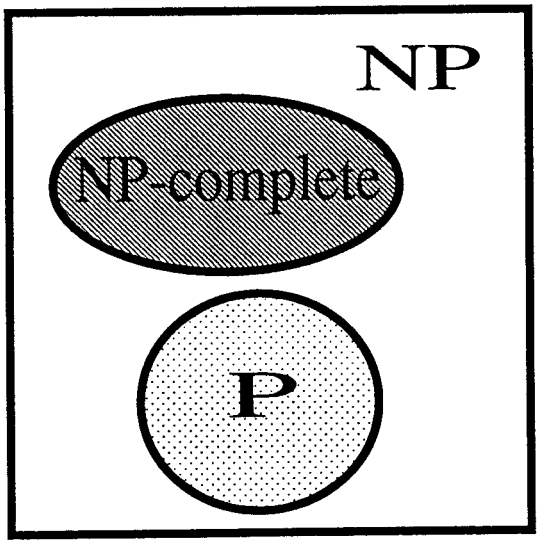

Figure 1.17: Relation between P and NP.

- Every problem which can be solved by NP is polynomial time reducible to the problem X.

Let us suppose that we can solve the NP-complete problem X within a polynomial time by the deterministic algorithm, namely, the problem X is P. Then, as every problem which can be solved by NP is polynomial time reducible to the problem X, these problem can be solved by P. Therefore, the condition NP $\subseteq$ P holds and as the result we can conclude

$$NP = P. \tag{1.61}$$

On the other hand, if it is impossible for us to solve the problem X by the P, the condition X $\in$ NP$-$P is satisfied and we can conclude

$$NP \neq P. \tag{1.62}$$

No one has been able to give the proof for deciding which is correct of NP=P and NP$\neq$P. We illustrated the relations between the NP, P and NP-complete in Figure 1.17. Most of the problems in which the variable takes discrete value belongs to the problem class NP-complete [38].

1.5.2 Traveling salesman problem

One of the famous NP-complete problems is the so-called *traveling salesman problem (TSP)*. Let us suppose that there are N cities and the distance between arbitrary two cities i and j is represented by d_{ij}. Then, the problem is defined as follows. What is the shortest closed way for the tourist to take under the condition that the tourist must visit each city only once. Suppose that there are N cities. Then, the number of closed paths is of order $\sim (N-1)!$. On the other hand, it takes about N steps for us to calculate the cost for one closed path. As the result, the calculation time for the TSP is about $\sim N!$. Therefore, it is obvious that the TSP belongs to the class NP. In addition, as it is possible for us to show that every problem which belongs to the NP is polynomial time reducible to the TSP, we are obliged to say that the TSP is a problem of class NP-complete. Although at present, there is no deterministic algorithm to solve the TSP within polynomial time, we cannot conclude that the condition NP$\neq$P holds because there is no rigorous proof for this statement. In order to define the problem mathematically, let us introduce the variable $n_{i,a}$ which takes 1 if the tourist visits the i-th city after $a-1$ other cities and 0 otherwise. The total distance of trip is written as

$$\mathcal{L} = \frac{1}{2} \sum_{ija} d_{ij} n_{i,a} (n_{j,a+1} + n_{j,a-1}). \tag{1.63}$$

We should minimize $\mathcal{L}$ under the next two conditions;

$$\sum_a n_{i,a} = 1 \text{ for } \forall i \tag{1.64}$$

$$\sum_i n_{i,a} = 1 \text{ for } \forall a \tag{1.65}$$

The first condition means that the tourist must visit each town only once during his trip. The second condition seems rather trivial and this condition restricts the tourist should be staying in a single city at any given time. Taking these conditions into account by introducing a Lagrange multiplier γ, the cost $\mathcal{L}$ is rewritten as

$$\mathcal{H} = \mathcal{L} + \frac{\gamma}{2}\left[\sum_a (1 - \sum_i n_{i,a})^2 + \sum_i (1 - \sum_a n_{i,a})^2\right]. \tag{1.66}$$

Therefore our problem is how we should minimize the above cost function (1.66). Besides of TSP, there are various combinational optimization problems which belong to the problem class NP complete, for example, the *weighted matching problem* [43], the *graph bipertitioning problem* [45] , *etc.*.

For the traveling salesman problem, Hopfield and Tank tried relaxing the variable appearing in the cost function, namely, they replaced the discrete variables $n_{i,a}$ with the continuous variables [40, 41, 43]. To this relaxation of the original problem, they applied the gradient descent method. However, as the problem size N becomes large (this is the practical case), the number of local minima of the cost becomes large and their method which depends on the gradient descent fails to obtain the optimal solution.

As no one has found a deterministic algorithm to solve such problems within polynomial time, a time of exponential order must be needed and usually we are obliged to use a heuristic strategy which strongly depends on each problem.

1.5.3 Simulated annealing

At present, the only general method for us to avoid this kind of difficulties may be simulated the annealing method [9] or the Boltzmann machine [8].

Therefore, we should investigate not only the learning algorithm for the specific learning systems but also the performance of this general method which may be applicable to other various cases of combinational optimization problems. According to this algorithm, the present state chooses a candidate for the new state and accepts the new state with probability 1 if the cost of the new state is lower than that of the old state. The problem is the case in which the cost of the new state is higher than that of the old state. If we do not accept the new state at all, this procedure coincides with the gradient decent method and we can not escape from the local minima. Therefore, we need some mechanism in order to escape from one of the local minima. For this purpose, we assume that the movement from the state with the lower cost function to the state with the higher cost function is allowed to some degree. Thanks to this stochastic hill-climbing processes, one can escape from a local minimum. As the acceptance probability, one usually use the next Metropolis type of the transition probability;

$$p = \min[1, \exp(-\Delta E/T)] \tag{1.67}$$

where ΔE is the change in the cost function and T is a parameter analogous to the temperature. As we mentioned in the brief review of the Boltzmann machine learning, it is an essential question for us to consider how fast the system should be cooled down in order to obtain the global minimum without being trapped in the local minima. It is obvious that if one cools down the system too quickly, one fails to obtain the global minimum due to trapping in one of local minima. One the other hand, if one decreases the temperature too slowly, it is difficult for us to apply the method to the practical problems although this schedule guarantees that we obtain a ground state of the system.

In order to understand this intuitively, let us consider the two level system which is illustrated in Figure 1.18. This energy landscape has two minima, namely, the local minium A and the global minimum B. If we choose the initial state at random and use the gradient decent method from the initial state, we obtain the energy A and B with equal probability $p_A = p_B = 1/2$ on condition that two basins of attraction

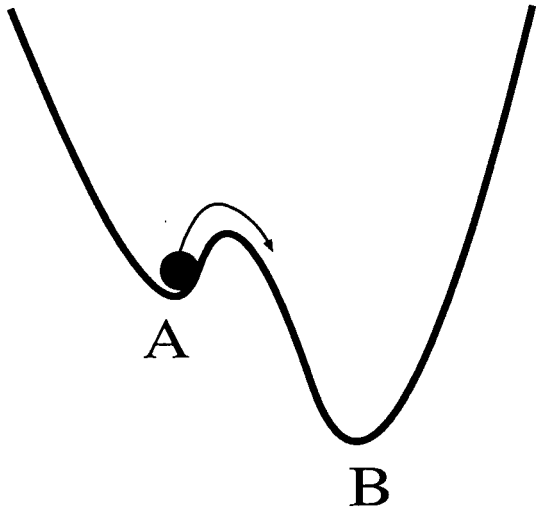

Figure 1.18: Energy landscape with the two minima (the global minimum is B). Thanks to the thermal fluctuation, the state can escape from the local minimum A.

for the minima A and B have the same size. Now, we should vibrate the system so as to obtain the probabilities p_A and p_B which satisfy $p_B \ll p_A$, or if possible, $p_A = 0$ and $p_B = 1$. For this purpose, if we vibrate the system with a large amplitude, the processes A→B and B→A occur with the same frequencies, and as the result, $p_A = p_B$ is obtained. In contrast, if the system is fluctuated with a small amplitude, it is guaranteed the probabilities $p_A = 0$ and $p_B = 1$, although it takes too long time. The most effective way to vibrate the system for us to obtain the probabilities $p_A = 0$ and $p_B = 1$ is decreasing the amplitude of the fluctuation from large amplitude to zero sufficient slowly. In this argument, the amplitude of the fluctuation corresponds to the temperature in the physical systems. Therefore, it is obvious that one of the essential points for this method is how one controls the temperature from high temperature to zero.

Geman and Geman [46], who are applied mathematicians, proved that one should employ the temperature schedule $T(t) \sim c/\log t$ to obtain the global minimum with probability unity. They used the transition probability which is proportional to the Boltzmann factor (1.67). The residual energy decreases to zero as $\sim 1/\log t$ using their temperature schedule. However, there is a possibility for us to obtain a faster convergence to the global minimum using a different temperature scheduling for other types of transition probability. This seems a very interesting open question. A part of this thesis is devoted to the examinations to answer the above important question.

1.6 Overview of this thesis

This thesis is composed of seven chapters.

In the next chapter 2, we introduce our model system as a typical example for the structural mismatch case and show general properties of the generalization error. For our model system, the best possible value of the generalization error and the optimal overlap between the teacher and student are represented explicitly as a function of the width of reversed wedge a. Therefore, our main goal is finding or improving learning algorithms so as to obtain the theoretical lower bound of the generalization error as fast as possible for all a values.

In chapter 3, in order to investigate the generalization error in the off-line mode, we introduce the

statistical mechanical formulations to calculate the learning curves using the *replica trick* which is originally applied to the random spin systems like spin-glasses [48] and recently used by the researchers who investigate the various complex systems [49, 50, 51]. Several interesting behaviors of the learning curves are revealed. For example, for some values of the parameter a, the solution of the saddle point equation becomes thermodynamically unstable and the generalization error shows a first order phase transition. In other words, for this parameter region of the teacher, the student suddenly acquires the excellent generalization ability. For a small range of parameter a, the generalization error converges to its minimum value as $\sim P^{-2/3}$ (P ; number of examples). This exponent of decay has not been known by the researcher working in the traditional statistics community. As we use a specific approximation when we estimate the saddle point of the free energy, namely, the *replica symmetric (RS) ansatz*, we check the validity of the solution by two different ways. We first investigate the fluctuation around the RS solution (order parameters), namely, we calculate so-called the *Almeida-Thouless (AT) line* [52] for our model system. We show rigorously that the AT line coincides with the critical storage capacity of the student network. As the critical storage capacity is finite, we should handle the asymptotic analysis with kid gloves. In order to see the validity of the asymptotic form of the generalization error, we also perform the computer simulation for the two-dimensional version of our model system. We find that our result is valid at least concerning the exponent of the learning curves.

In chapter 4, we investigate the model system according to the on-line learning mode. In this learning mode, the learning processes and the corresponding generalization error can be described by coupled differential equations with respect to two relevant macroscopic order parameters, namely, the length of the student weight vector and the overlap between teacher and student. We investigate the learning processes of the student who is trained by the conventional *perceptron*, *Hebbian* and *AdaTron (Adaptive perceptron)* learning algorithms and calculate the learning curves for both small example region ($P \sim \mathcal{O}(1)$) and asymptotic region ($P \to \infty$) respectively. We reveal that each of these conventional three learning algorithms can not obtain the theoretical lower bound of the generalization error for a specific range of parameter a and for such a range, learning of the student fails and the learning dynamics stops at some fixed point. In order to overcome this difficulty, we also try to modify these conventional learning algorithms using various ways (optimization of the learning rate, the weight decay term, query and so on) so that the dynamics escapes from fixed points. We also compare the generalization performances of the off-line learning with those of the on-line learning.

In chapter 5, in order to investigate the performance of the learning of a non-monotonic perceptron, we devote this chapter to considering the case in which a non-monotonic perceptron learns from a non-monotonic perceptron in the on-line learning mode. We find that even for this realizable case, the student sometimes can not generalize the task of the teacher for small a. We call this phenomenon *anti-learning*. In order to avoid the anti-learning, we consider several learning algorithms. Especially, we introduce the learning function in the learning dynamics and optimize the function using the well-known *Bayes formula*.

In chapter 6, we test a different type of transition probability from the Boltzmann-Gibbs one with the optimal temperature scheduling and give a rigorous proof of the faster convergence to the global minimum. One of the candidates for the transition probability was introduced by Tsallis and Stariolo [54]. We prove weak ergodicity of the inhomogeneous Markov process generated by the generalized transition probability of Tsallis and Stariolo [54] with power-law decay of the temperature. We thus have a mathematical basis to conjecture convergence of simulated annealing processes with the generalized transition probability to the minimum of the cost function. An explicitly solvable example in one dimension is analyzed in which the generalized transition probability leads to a fast convergence of the cost function to the optimal value. We also investigate how far our arguments depend upon the specific form of the generalized transition probability proposed by Tsallis and Stariolo [54]. It is shown that a few requirements on analyticity of the transition probability are sufficient to assure fast convergence in the case of the solvable model in one dimension.

In the last chapter 7, we summarize all the results obtained in this thesis.

Chapter 2

The Model System for Unrealizable Rule

This chapter is devoted to introducing our model system and generic properties of the generalization error.

Our problem is defined as follows. The teacher signal is provided by a single-layer perceptron with an N-dimensional weight vector $\boldsymbol{B}$ and a non-monotonic (reversed wedge) transfer function

$$T_a(v) = \mathrm{sign}\Big[v(a-v)(a+v)\Big] \tag{2.1}$$

where $v \equiv \sqrt{N}(\boldsymbol{B}\cdot\boldsymbol{x})/|\boldsymbol{B}|$, $\boldsymbol{x}$ is the input vector normalized to unity, a is the width of the reversed wedge, and sign denotes the sign function. The student is a simple perceptron with the weight vector $\boldsymbol{J}$ whose output is

$$S(u) = \mathrm{sign}(u) \tag{2.2}$$

where $u \equiv \sqrt{N}(\boldsymbol{J}\cdot\boldsymbol{x})/|\boldsymbol{J}|$. In this thesis we assume that the components of $\boldsymbol{x}$ are drawn independently from a uniform distribution on the N-dimensional unit sphere. The student can learn the rule of the teacher perfectly if and only if $a = \infty$. For finite a, the student fails to reproduce examples correctly. In fact, as we mentioned in Introduction, the non-monotonic perceptron may be regarded as a variant of parity machine which has a structure quite different from a simple perceptron. Gardner's capacity of the non-monotonic associative memory dramatically increases according to statistical-mechanical calculations [28, 29, 31]. It should also be added here that the associative memory of the Hopfield-type with a non-monotonic transfer function can store more patterns than the conventional Hopfield model [32, 33, 37].

In order to evaluate the achievement of learning by a student, it is convenient to introduce the following two order parameters. One is the overlap between $\boldsymbol{B}$ and $\boldsymbol{J}$

$$R = \frac{\boldsymbol{B}\cdot\boldsymbol{J}}{|\boldsymbol{B}||\boldsymbol{J}|} \tag{2.3}$$

and the other is the norm of the student weight vector

$$l = \frac{|\boldsymbol{J}|}{\sqrt{N}}. \tag{2.4}$$

As a consequence of the central limit theorem, the random variables u and v obey the normal distribution

$$P_R(u,v) = \frac{1}{2\pi\sqrt{1-R^2}} \exp\left[-\frac{u^2+v^2-2Ruv}{2(1-R^2)}\right]. \tag{2.5}$$

The generalization error ϵ_g, or the student probability of producing a wrong answer, can be obtained by integrating the above distribution over the region satisfying $T_a(v) \neq S(u)$ in the two-dimensional u-v

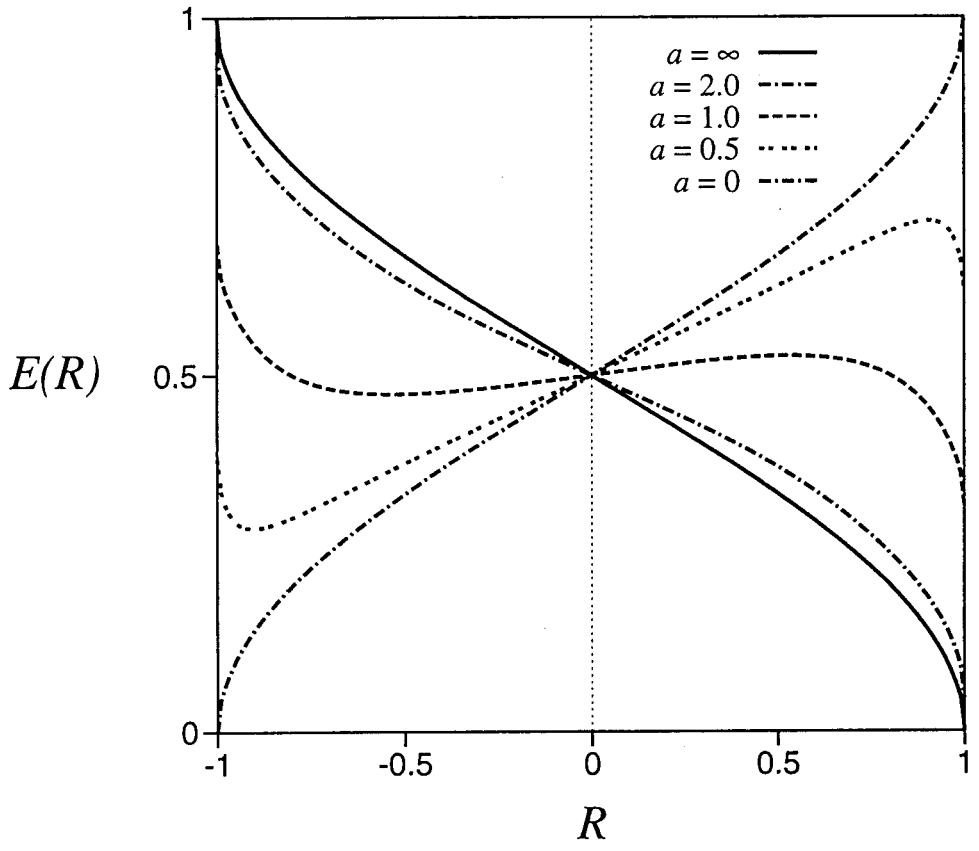

Figure 2.1: Generalization error as a function of the overlap R for $a = \infty$, 2.0, 1.0, 0.5 and 0. For $a = \infty$ where the student and the teacher have the same structure, the generalization error decreases to zero as the student is trained so that R goes to 1. On the other hand, for $a = 0$ (student's output is the inverse of teacher's), the generalization error decays to zero as the student is trained so that R goes to -1 instead of 1.

space. After simple calculations we find

$$\epsilon_g \equiv E(R) = 2\int_a^\infty \mathrm{D}v\, H\left(\frac{-Rv}{\sqrt{1-R^2}}\right) + 2\int_0^a \mathrm{D}v\, H\left(\frac{Rv}{\sqrt{1-R^2}}\right) \tag{2.6}$$

where $H(x) = \int_x^\infty Dv$ and $\mathrm{D}v \equiv dv \exp(-v^2/2)/\sqrt{2\pi}$. The above expression can be rewritten as

$$E(R) = \int_{-\infty}^0 Dt\, \Omega(R:t) \tag{2.7}$$

where

$$\Omega(R:t) \equiv \int_{-\infty}^\infty Dz \Big[\Theta(-z\sqrt{1-R^2} - Rt - a) + \Theta(z\sqrt{1-R^2} + Rt) - \Theta(z\sqrt{1-R^2} + Rt - a)\Big]. \tag{2.8}$$

In Figure 2.1 we plot $E(R)(= \epsilon_g)$ for several values of the parameter a. From this figure, we see that for $a = \infty$ (the learnable limit), ϵ_g goes to zero when R approaches 1. In contrast, for $a = 0$, ϵ_g goes to zero when R reaches -1. This result is trivially expected because for $a = 0$ the teacher input-output

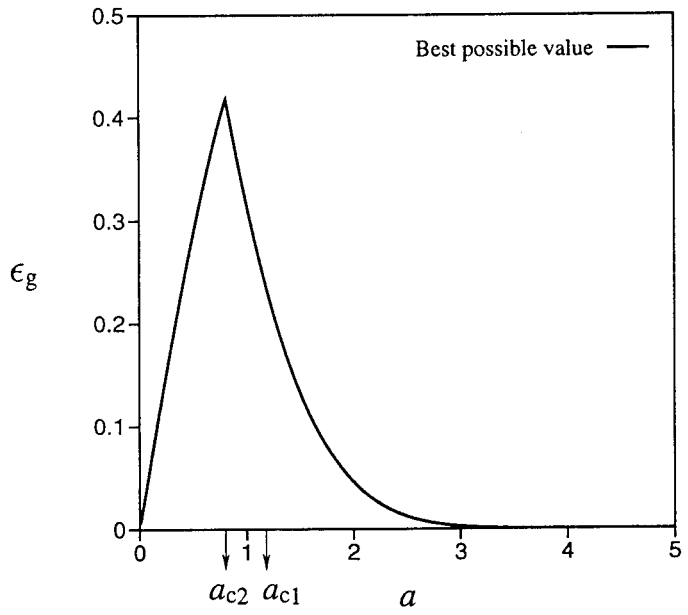

Figure 2.2: The global minimum of $E(R)$. From the definition of $E(R)$, this corresponds to the optimal value of the generalization error $\epsilon_{\rm opt}$.

relation is completely the opposite of that of the student. In the parameter region between $a = 0$ and $a = \infty$, the generalization error shows highly non-trivial behavior. The critical value R_* of the order parameter is defined as the point where $E(R)$ is locally minimum. Explicitly,

$$R_* = -\sqrt{\frac{2\log 2 - a^2}{2\log 2}} \tag{2.9}$$

which exists for $a \leq a_{c1} = \sqrt{2\log 2} = 1.18$. The local minimum value $E_{\rm local} = E(R_*)$ is

$$E_{\rm local} = 2\int_a^\infty Dv\, H\left(\frac{\sqrt{2\log 2 - a^2}}{a} v\right) + 2\int_0^a Dv\, H\left(-\frac{\sqrt{2\log 2 - a^2}}{a} v\right). \tag{2.10}$$

We plot in Figure 2.2 the value of the global minimum of $E(R)$ which means the smallest possible generalization error irrespective of learning algorithms. In Figure 2.3, we show the value of R which gives the global minimum. We notice that for $a < a_{c2} \equiv 0.80$, $E(R = R_*) = E_{\rm local}$ is also the global minimum. On the the other hand, for $a > a_{c2}$, the global minimum shifts to $E(R = 1)$. In addition, these two figures tell us that the optimal generalization error is obtained by training the student weight vector $\boldsymbol{J}$ so that R goes to 1 for $a > a_{c2}$. Therefore, for $a > a_{c2}$, the student achieves the optimal generalization ability by mimicking the teacher vector, $\boldsymbol{J} = \boldsymbol{B}$. This critical value a_{c2} is given by the condition $E(R = 1) = E_{\rm local}$. Explicitly,

$$H(a) = \int_a^\infty Dv\, H\left(\frac{\sqrt{2\log 2 - a^2}}{a} v\right) + \int_0^a Dv\, H\left(-\frac{\sqrt{2\log 2 - a^2}}{a} v\right) \tag{2.11}$$

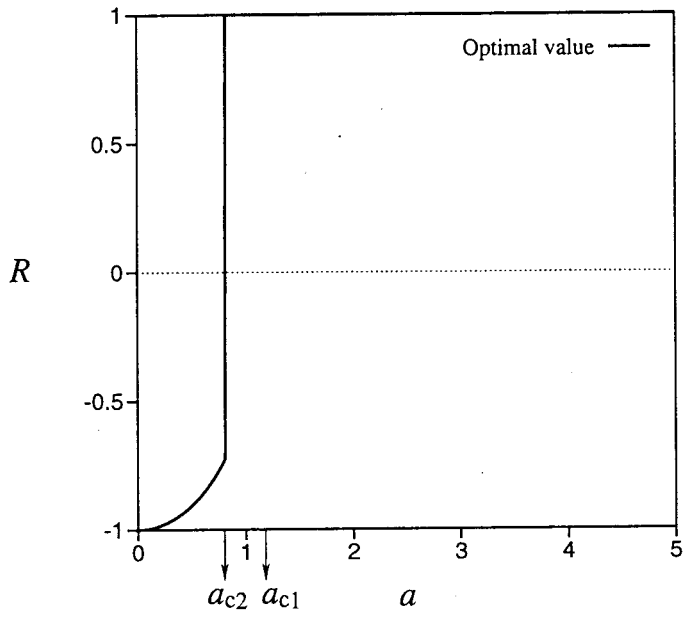

Figure 2.3: The optimal order parameter R which gives the global minimum, namely, the optimal generalization error ϵ_{opt}. The system shows a discontinuous phase transition at $a = a_{c2} = 0.80$ from the phase described by $R = 1$ to the phase described by $R = R_*$.

On the other hand, for $a < a_{c2}$, the optimal generalization cannot be achieved even if the student succeeds in finding B completely. In this curious case, the optimal generalization is obtained by training the student so that the student finds his weight vector which satisfies $R = R_*$ instead of $R = 1$. This is a very interesting situation because conventional learning theories have mainly put their focus upon constructing or improving learning algorithms to find the teacher vector as efficiently as possible.

Another remarkable feature is that this system shows a first order phase transition at $a = a_{c2}$ from a phase described by $R = 1$ to a phase characterized by $R = R_*$. At $a = a_{c2}$ the generalization error has the maximum value as seen in Figure 2.2.

Chapter 3

Off-Line Learning

In this chapter, we investigate the learning properties in the off-line (or batch) mode. In the next section 3.1, we explain the statistical mechanical formulation for calculating the learning curves. In section 3.2, we introduce the *minimum error algorithm* to find a target and calculate the learning curves in the framework of statistical mechanics. In particular, the asymptotic behavior of the learning curves is investigated analytically. In section 3.3, we examine the computer simulation for the two-dimensional version of our model system in order to check the validity of the analytical calculations. In section 3.4, we discuss the validity of our approximation which is used in section 3. Section 5 is devoted to summary.

3.1 Statistical mechanics

In the off-line learning, the student receives a set of examples $\xi^P \equiv \{(y_\mu, \boldsymbol{x}_\mu); \mu = 1, \cdots P\}$ at the same time. Then, the *cost function* (or the *energy function*) is defined as the frequency of disagreement of teacher and student outputs. As a simple selection of the cost function, we use the next *Gardner-Derrida cost function* [55, 56];

$$E(\boldsymbol{J}|\xi^P) = \sum_{\mu=1}^{P} \Theta\left(-T_a(v^\mu)\cdot S(u^\mu)\right) \tag{3.1}$$

where $\Theta(x)$ is the Heviside step function. There are several cost functions [57] except for the Gardner-Derrida cost function, for example, the *perceptron cost function* [2, 58], the *relaxation cost function* [59, 58] or the *AdaTron cost function* [60]. In the fields of statistics or mathematical engineering, the the graded-response transfer functions have been used because this type of transfer functions is suitable for dealing with mathematically. For these grade-response transfer functions, the cost function (what we call the *AdaLine cost function*) [61] is also defined as

$$E(\boldsymbol{J}|\xi^P) = \frac{1}{2}\sum_{\mu=1}^{P} [y_P - f(u)]^2 \tag{3.2}$$

where the graded-response transfer function f has been selected as $\tanh(u)$ or the error function $\mathrm{erf}(u)$.

In the off-line learning, the student should continue to learn until he reaches to some error torelance [1]. It must be noticed that $\boldsymbol{J}$ is a dynamical variable and can move during the training process according to some rule. Therefore, we should construct a learning algorithm so that this dynamical variable adapts to a rule of the teacher. In contrast, the training set $\boldsymbol{\xi}^P$ consists of quenched random variables and once the training set is given, the variables are frozen and do not move during the training. The shape of the complicated energy landscape is caused by the quenched random variables. In order to find a solution $\boldsymbol{J}_*$ which gives some error tolerance of the cost function, one usually uses the gradient decent method in

[1] This error tolerance is decided by the learning system. For the unrealizable task, we can not make the cost function zero in principle and we should set the torelance for the student.

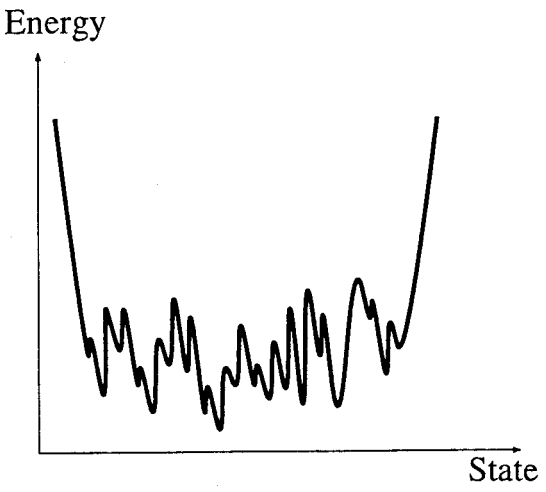

Figure 3.1: A typical energy landscape of the cost function. There exist a lot of local minima. If one applies the gradient decent method from an arbitrary initial state, one is trapped in a local minimum and cannot obtain the global minimum.

the weight space defined as

$$\frac{\partial \boldsymbol{J}}{\partial t} = -\nabla_{\boldsymbol{J}} E(\boldsymbol{J}|\xi^P). \tag{3.3}$$

In general, the cost function $E(\boldsymbol{J}|\xi^P)$ has a lot of local minima as we show in Figure 3.1. If the number of examples P is of order N, namely, $P = \alpha N$, and the size of the problem N is extremely large $N \to \infty$, the landscape of the cost function has a very complicated structure like spin glasses at low temperature (see Figure 3.1). Therefore, if one uses the gradient decent method (3.3), the present state is trapped in a local minimum and fails to obtain the global minimum. In order to escape from these local minima, we add a white noise η to the right hand side of equation (3.3) and obtain the next relaxational *Langevin equation*;

$$\frac{\partial \boldsymbol{J}}{\partial t} = -\nabla_{\boldsymbol{J}} E(\boldsymbol{J}|\xi^P) + \eta(t) \tag{3.4}$$

where η has a variance

$$< \eta_i(t)\, \eta_j(t') > = 2T\delta_{ij}\delta(t - t'). \tag{3.5}$$

The above dynamics is inclined to decrease the cost function, but occasionally the cost function may increase due to the thermal noise. Obviously, at $T = 0$, the noise term drops out and the gradient decent is recovered.

The statistical mechanics tells us that the above stochastic dynamics converges to the Gibbs-Boltzmann distribution of the weight vector $\boldsymbol{J}$;

$$\rho(\boldsymbol{J}) = \frac{\exp\left[-\beta E(\boldsymbol{J}|\xi^P)\right]}{Z} \tag{3.6}$$

where $\beta \equiv T^{-1}$ and Z is the normalization factor. In Figure 3.2 we illustrate the shape of the above distribution for the cases of high temperature (left) and low temperature (right). From these figures we see that as the temperature decreases the distribution becomes to have a sharp peak around the solution $\boldsymbol{J}_*$. Therefore, if we decrease the temperature as slowly as possible, it may be possible to obtain the optimal distribution which has a delta peak at the solution $\boldsymbol{J}_*$[2].

The above situation is also understood intuitively as follows. At first we prepare a lot of ($\sim \mathcal{O}(N)$) independent student networks. Then, these networks change their weights according to the stochastic

[2]It is well known that one can obtain the global minimum with probability unity if the system is cooled down according to the schedule $T \sim 1/\log t$. This is explained in detail in chapter 6.

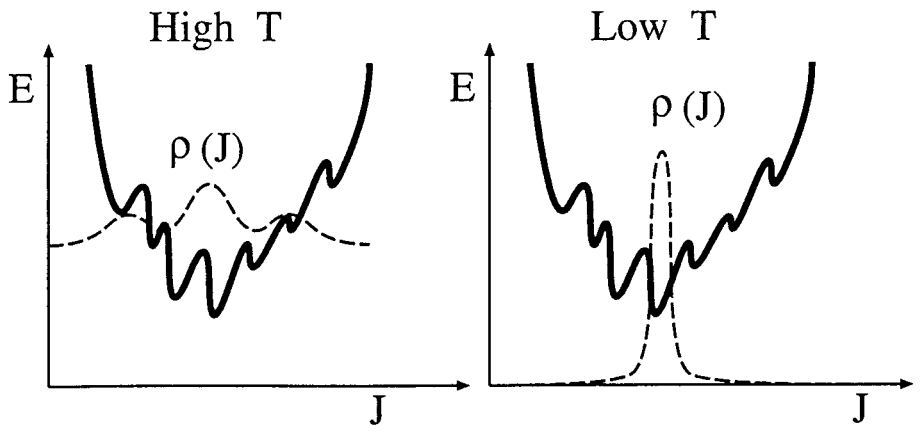

Figure 3.2: The distributions of the weight vector. The solid line is the energy landscape of the cost function. The broken lines are two cases of the distribution. The left is a high temperature case and the right is a low temperature case.

process (3.4). When the temperature is high, the networks which have various weights are produced. However, as the system is cooled down, we can obtain the networks which are close to the most excellent student with the solution weight $\boldsymbol{J}_*$ and almost all of the networks become the most excellent student when the temperature comes to zero effectively.

Although we have restricted ourselves to the case of continuous weight, it is possible for us to extend the above algorithm to the case of discrete weight. In such a case we should use the discrete time Monte Calro simulation instead of (3.4), namely, one should consider the stochastic process according to the next transition probability (what we call the *Metropolis sampling*);

$$P(\boldsymbol{J}\to\boldsymbol{J}') = \min\left\{1, \exp\left[-\beta\left(E(\boldsymbol{J}'|\xi^P) - E(\boldsymbol{J}|\xi^P)\right)\right]\right\} \tag{3.7}$$

where $\boldsymbol{J}'$ means the state into which an arbitrary node of $\boldsymbol{J}$ is flipped.

In the next section, we apply this statistical mechanical analysis to our model system.

3.2 Replica calculations of learning curves

In this section we calculate the generalization abilities of the student using statistical mechanics, especially, the replica method.

In the off-line learning scenario, we need not information about the weight length of the student. As we explain in the next chapter, the length of the student weight vector reflects the number of the presented examples. Therefore, if we would like to investigate the dynamical processes of the learning in detail, we need the information about the dynamical behavior of the student weight vector. For example, if the length of the student weight vector monotonically increases in proportion to the increases of the presented examples, we can conclude that there do not exist the local minima or the fixed points, and the teacher task is realizable for the student. In addition, if the length of the student weight vector stops growing or decreases although the teacher continues to present the new examples, we can see that the task of the teacher is unrealizable for the student.

On the other hand, for the off-line learning we investigate as follow, as we treat the equilibrium properties on condition that a finite number of examples is given, we do not have to mention the dynamical properties of the length of the student weight. In addition, it is convenient for us to treat the normalized student weight when we use the concept of the version space.

For these reasons, in this chapter we fix the length as $l = |\boldsymbol{J}|/\sqrt{N} = 1$ and the weight of the student is normalized as $|\boldsymbol{J}| = \sqrt{N}$.

As we saw in the previous section, the number of false predictions on the given set of examples ξ^P, namely, the cost function ([55, 56]) is rewritten as

$$E(\boldsymbol{J}|\xi^P) = \sum_{\mu=1}^{P} \Theta(-y_\mu \cdot u_\mu), \tag{3.8}$$

where $u_\mu \equiv (\boldsymbol{J} \cdot \boldsymbol{x}_\mu)/\sqrt{N}$ and $y_\mu \equiv T_a(v_\mu)$ with $v_\mu = (\boldsymbol{B} \cdot \boldsymbol{x}_\mu)/|\boldsymbol{B}|$. We call the learning algorithm following this strategy the *minimum error algorithm*. The cost function (3.8) is identical to that of Gardner and Derrida and the learning process of the minimum error algorithm is investigated by their analysis as follows [55, 56] [3]. From the *energy* defined by equation (3.8), the *partition function* with the inverse temperature β is given by

$$\begin{aligned} Z(\beta) &= \int d\boldsymbol{J}\delta(|\boldsymbol{J}|^2 - N)\exp[-\beta E(\boldsymbol{J}|\xi^P)] \\ &= \int d\boldsymbol{J}\delta(|\boldsymbol{J}|^2 - N)\prod_{\mu=1}^{P}[e^{-\beta} + (1 - e^{-\beta})\Theta(y_\mu \cdot u_\mu)]. \end{aligned} \tag{3.9}$$

Minimization of $E(\boldsymbol{J}|\xi^P)$ corresponds to the limit $\beta \to \infty$ in the Gibbs-Boltzmann distribution and we focus on this limit hereafter.

3.2.1 Below α_c

Let us remind that a simple perceptron with N input units at most stores $p_c = 2N$ random patterns by choosing the optimal weight vector. This result was first obtained by Cover [14] using the combinational mathematics. Recently, Gardner succeeded in calculating this maximum number of the storage patterns for a simple perceptron by the replica method. Let us define the critical storage capacity of the network as $\alpha_c \equiv P_c/N$. We translate the above statement into the problem of supervised learning, in which a simple perceptron as the student learns from the same type of the network as the teacher, as follows.

If we regard that the random patterns are produced by the teacher network, it is impossible for the student to make the cost function, which consists in more than $\alpha_c = 2$ examples, zero in principle. The above statement is also rewritten as follows. The student can not find the solution weight in his version space if the size of the training set exceeds $\alpha_c = 2$. The reason why this limitation exists for this teacher-student learning scenario is that there is no correlation between the teacher weight and the student weight. Let us consider the extreme case in which the teacher weight coincides with that of the student. It is obvious that the student can store all examples which were produced by the teacher. Therefore, if we translate the above fact into the problem of information storage for a simple perceptron, we can say that the critical storage capacity of the student perceptron is infinity, namely, $\alpha_c = \infty$ It is important for us to remember that the solution exist in the version space as long as the student has the same structure as the teacher. Although we mentioned about the realizable case, the above argument can be extended to the unrealizable problem. As we explained in the introduction, the critical storage capacity of a simple perceptron with the non-monotonic transfer function is larger than that of a simple perceptron. The critical storage capacity for the non-monotonic perceptron is calculated as $\alpha_c = 10.5$ within the replica symmetric assumption. Therefore, for our model system, the student can not reproduce a part of the input-output relations even if the student has the same weight as that of the teacher. In other words, it is impossible for us to find the solution in the version space if the size of the training set which was produced by the non-monotonic teacher exceeds the critical storage capacity of the student $\alpha_c = 2$. However, there remain some weights which completely reproduce the input-output relations among the training set ξ^P until the ratio $\alpha = P/N$ increases up to some critical capacity α_c even if teacher relation is unrealizable. This enables us to calculate the learning curve below α_c by evaluating the logarithm of the *Gardner-Derrida volume* $V_{GD} = Z(\infty)$ through the replica formula

$$\frac{\ln V_{GD}}{N} = \frac{\ll \ln Z(\infty) \gg_{\xi^P}}{N} = \frac{1}{N}\lim_{n\to 0}\frac{\ll Z^n(\infty) \gg_{\xi^P} - 1}{n}, \tag{3.10}$$

[3] Although we use the Gardner-Derrida cost function, it is also possible to apply the other cost functions to our model system. However, in order to derive a typical behavior of the learning curves, it is enough to examine the Gardner-Derrida cost function.

where $\ll \cdots \gg_{\xi^P}$ represents the average over the example set ξ^P. $Z^n(\infty)$ is the simultaneous partition function of n-replicated systems sharing the same random variables ξ^P and becomes a function of order parameters

$$R_a = \frac{\boldsymbol{B}\cdot\boldsymbol{J}_a}{N}, \tag{3.11}$$

$$q_{ab} = \frac{\boldsymbol{J}_a\cdot\boldsymbol{J}_b}{N}, \tag{3.12}$$

where $a = 1, \cdots, n$ and $b = 1, \cdots, n$ are the indices which represent replicated systems.

Under the replica symmetric (RS) ansatz

$$R_a = R, \tag{3.13}$$

$$q_{ab} = q, \tag{3.14}$$

we can show (see Appendix A) that the equation (3.10) is evaluated as

$$\mathrm{ext}_{\{R,q\}}\left\{2\alpha\int_{-\infty}^{\infty}Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)\ln\Xi(q:t)+\frac{1}{2}\ln(1-q)+\frac{q-R^2}{2(1-q)}\right\}, \tag{3.15}$$

where $\Omega(R:t)$ is defined as in equation (2.8) and

$$\Xi(q:t)\equiv\int_{-\infty}^{\infty}Dz\,\Theta(\sqrt{1-q}z+\sqrt{q}t). \tag{3.16}$$

From the identities for the arbitrary function $F(t)$

$$\frac{\partial}{\partial R}\int_{-\infty}^{\infty}Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)F(t) = \frac{1}{R}\int_{-\infty}^{\infty}Dt\,\frac{\partial\Omega}{\partial t}\left(\frac{R}{\sqrt{q}}:t\right)\frac{\partial F(t)}{\partial t}, \tag{3.17}$$

$$\frac{\partial}{\partial q}\int_{-\infty}^{\infty}Dt\,\Xi(q:t)F(t) = \frac{1}{2q}\int_{-\infty}^{\infty}Dt\,\frac{\partial\Xi}{\partial t}(q:t)\frac{\partial F(t)}{\partial t}, \tag{3.18}$$

we find that equation (3.15) yields the following set of saddle point equations

$$2\alpha\int_{-\infty}^{\infty}Dt\,\Omega\times\left(\frac{\Omega_t}{\Omega}\right)\times\left(\frac{\Xi_t}{\Xi}\right) = \frac{R^2}{1-q}, \tag{3.19}$$

$$2\alpha\int_{-\infty}^{\infty}Dt\Omega\times\left(\frac{\Xi_t}{\Xi}\right)^2 = \frac{q(q-R^2)}{(1-q)^2}, \tag{3.20}$$

where F_t represents the abbreviation of the partial derivative of a function F with respect to t. By solving the saddle point equations (3.19) and (3.20), we can investigate the learning process below α_c.

As the number of the examples increases, the number of the candidates of the solution, which is able to reproduce the examples perfectly, decreases. From the definition of the critical storage capacity, for the critical number of examples α_c, there only exists a solution $\boldsymbol{J}_*$ in the version space and the volume of the version space shrinks to zero. Therefore, at the critical storage capacity α_c, the solution $\boldsymbol{J}_*$ should be independent of the replica index. Taking this assumption into account, we obtain the condition on the order parameter q_{ab} at the critical storage capacity as follows.

$$q_{ab} = q = \frac{\boldsymbol{J}\cdot\boldsymbol{J}}{N}\to 1. \tag{3.21}$$

Taking this limit $q\to 1$ in the saddle point equations (3.19) and (3.20), we obtain a couple of equations which determine α_c as

$$-2\alpha_c\int_{-\infty}^{0}Dt\,t\,\Omega_t(R_c:t) = {R_c}^2, \tag{3.22}$$

$$2\alpha_c\int_{-\infty}^{0}Dt\,t^2\,\Omega(R_c:t) = 1-{R_c}^2, \tag{3.23}$$

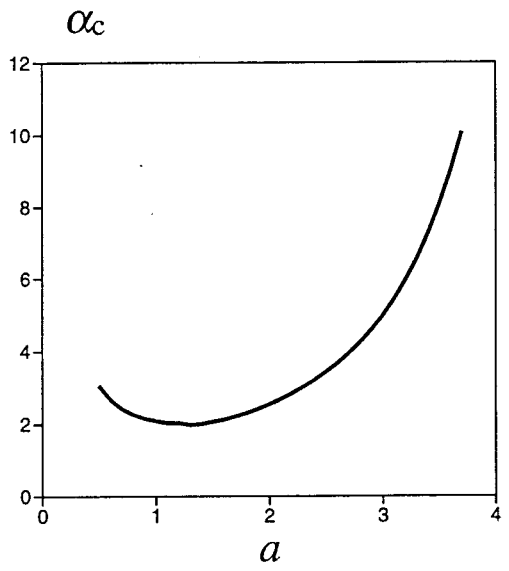

Figure 3.3: The critical storage capacity α_c as a function of the width of the reversed wedge a.

where R_c is the value of R at the critical capacity α_c. Here, we have used the relation

$$\frac{\Xi_t}{\Xi} \sim -\frac{q}{1-q}\, t\,\Theta(-t), \tag{3.24}$$

which is valid in the limit $q \to 1$. The critical storage capacity α_c and the critical overlap R_c which are obtained from these equations are plotted as functions of a in Figures 3.3 and 3.4. From Figure 3.3, we see that the critical storage capacity of the student takes a minimum value $\alpha_c = 2$ at $a = a_{c2} = \sqrt{2\log 2}$. We also see from Figure 3.4 that the critical overlap which gives α_c becomes zero at $a = a_{c2}$. From these results, it is obvious that at $a = a_{c2}$, there is no correlation between the weights of teacher and student. As the results, for the student, the teacher signal seems to be random data. Therefore, α_c at $a = a_{c2}$ corresponds to the critical storage capacity of a simple perceptron, what we call the *Gardner capacity* [55]. On the other hand, when $a = \infty$, namely, $T_a = S$, the student can reproduce all teacher's input-output relations perfectly and as a result, the storage capacity α_c diverges.

3.2.2 Beyond α_c

The solution of equations (3.19) and (3.20) disappears for $\alpha > \alpha_c$. This reflects the fact that beyond α_c there is no weight which completely reproduces the input-output relations among ξ^P. This makes the Gardner-Derrida volume V_{GD} shrink to zero. Therefore, we can not investigate the learning process by evaluating eq. (3.10). Instead, the *free energy*

$$-f = \lim_{\beta\to\infty} \frac{\ll \ln Z(\beta) \gg_{\xi^P}}{N\beta} = \lim_{\beta\to\infty} \lim_{n\to 0} \frac{\ll Z^n(\beta) \gg_{\xi^P} - 1}{nN\beta} \tag{3.25}$$

gives us a solution for $\alpha > \alpha_c$.

$\ll Z^n(\beta) \gg_{\xi^P}$ also becomes a function of order parameters R_a and q_{ab}. Under the RS ansatz (3.13) and (3.14), it can be shown (see Appendix A) that equation (3.25) with finite β is evaluated as

$$\mathrm{ext}_{\{R,q\}} \left\{ \frac{2\alpha}{\beta} \int_{-\infty}^{\infty} Dt\, \Omega\left(\frac{R}{\sqrt{q}} : t\right) \ln \Xi_\beta(q:t) + \frac{1}{2\beta}\ln(1-q) + \frac{q-R^2}{2\beta(1-q)} \right\}, \tag{3.26}$$

where

$$\Xi_\beta(q:t) \equiv e^{-\beta} + (1-e^{-\beta})\Xi(q:t). \tag{3.27}$$

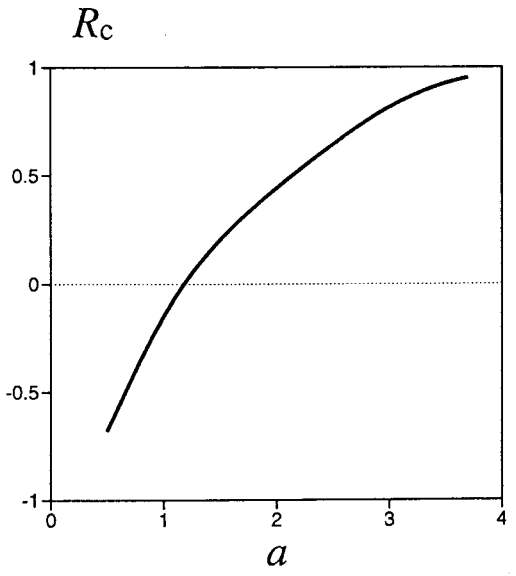

Figure 3.4: The critical overlap R_c which gives the critical storage capacity α_c as a function of the width of the reversed wedge a.

In the limit $\beta \to \infty$, a non-trivial result is obtained only when $q \to 1$ keeping $x \equiv \beta(1-q)$ finite. Then, equation (3.26) becomes

$$\mathrm{ext}_{\{R,x\}} \left\{ -2\alpha \left[\int_{-\infty}^{0} Dt\, \Omega(R:t) \{\Theta(-t-\sqrt{2x}) + \frac{t^2}{2x}\Theta(t+\sqrt{2x})\} \right] + \frac{1-R^2}{2x} \right\}, \tag{3.28}$$

which yields the following saddle point equations

$$-2\alpha \int_{-\sqrt{2x}}^{0} Dt\, t\, \Omega_t(R:t) = R^2, \tag{3.29}$$

$$2\alpha \int_{-\sqrt{2x}}^{0} Dt\, t^2\, \Omega(R:t) = 1-R^2. \tag{3.30}$$

In the derivation of equation (3.28), we have used the relation

$$\Xi_\beta(q:t) \sim \begin{cases} e^{-\beta}, & \text{for } t < -\sqrt{\frac{2\beta(1-q)}{q}} \\ -\frac{\sqrt{1-q}}{\sqrt{2\pi q}t} e^{-\frac{q}{2(1-q)}t^2}, & \text{for } -\sqrt{\frac{2\beta(1-q)}{q}} < t < 0 \\ 1, & \text{for } 0 < t \end{cases} \tag{3.31}$$

which is valid in the limit $\beta \to \infty$ and $q \to 1$.

Before proceeding further, we mention the stability of the RS solution obtained from the saddle points equations (3.29) and (3.30). Unfortunately, our RS solution becomes thermodynamically unstable for $\alpha > \alpha_c$. We show this condition on the stability of the RS solution, that is, the AT line is exactly identical to the critical storage capacity α_c in Appendix B using the same technique as Bouten [62].

Therefore, we have to take the replica symmetry breaking (RSB) into account in order to obtain a stable solution. However, it is much more involved to compute RSB solutions and such a computation is beyond the scope of this thesis. In addition, Whyte and Sherrington showed that the one-step RSB solution is also unstable for the problem of storing random patterns and it is conjectured that any finite step of RSB is not sufficient in order to obtain a thermodynamically stable solution [63, 64, 65]. For these reasons, we here only present unstable RS solutions hoping that they are still good approximations and discuss their validity by comparing them with the results obtainable in a low-dimensional version of the present problem in the next section.

By solving the saddle points equations (3.29) and (3.30), we found that the feature of the learning is classified into the following five types depending on a.

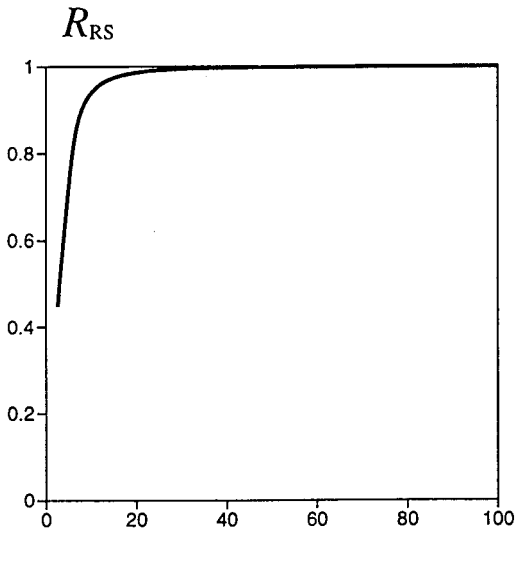

Figure 3.5: The order parameter R as a function of α for the case of $a > a_{c0} \sim 1.53$.

(1) $a = \infty$, 0 **(realizable cases)**

The teacher becomes realizable for $a = \infty$ because the teacher is identical to the student with $R = 1$ ($\boldsymbol{J} = \boldsymbol{B}$). In addition, the teacher is also realizable for $a = 0$. This is because for $a = 0$ its input-output relation is completely opposite to that of $a = \infty$, which means the student with $R = -1$ ($\boldsymbol{J} = -\boldsymbol{B}$) exactly mimics the teacher. For these special values of a, α_c becomes infinity and the learning is described by equations (3.19) and (3.20) even in the limit $\alpha \to \infty$. The solution of these equations is thermodynamically stable and the learning curve is identical to that obtained in a realizable problem [24, 75] which has the asymptote

$$\epsilon_g \sim 0.624\,\alpha^{-1}. \tag{3.32}$$

This is consistent with the universal scaling observed in general realizable problems[76].

(2) $a > a_{c0} \sim 1.53$

In this parameter region, we found that the order parameter R monotonically increases to 1 as $\alpha \to \infty$ (Figure 3.5). On the other hand, x once decreases from ∞ to some value, and after that, approaches up to $a^2/2$ in the limit $\alpha \to \infty$ (Figure 3.6).

In order to investigate how fast R and x converge to these limiting values, we expand equations (3.29) and (3.30) with small parameters $\varepsilon = 1 - R$ and $\Delta x = a^2/2 - x$. This yields the following equations

$$\alpha \left[\frac{ae^{-a^2/2}}{\sqrt{2\pi}} H\left(\frac{\Delta x}{a\sqrt{2\varepsilon}}\right) + \frac{(1-e^{-a^2/2})}{2\pi}\sqrt{2\varepsilon}\right] \sim 1, \tag{3.33}$$

$$\alpha \left[\frac{\varepsilon^{3/2}}{\sqrt{2\pi}} + \frac{a^2 e^{-a^2/2}}{\sqrt{2\pi}} H\left(\frac{\Delta x}{a\sqrt{2\varepsilon}}\right)\right] \sim 2\varepsilon, \tag{3.34}$$

where $H(x) \equiv \int_x^\infty dt \exp[-t^2/2]/\sqrt{2\pi}$ and these imply the following scalings

$$\varepsilon \sim \alpha^{-2}, \tag{3.35}$$

$$\Delta x \sim (\ln \alpha)^{1/2} \alpha^{-1}. \tag{3.36}$$

From equations (2.7) and (2.8), we found the relation

$$E(1-\varepsilon) - E(R=1) \sim \mathcal{O}(\varepsilon^{1/2}). \tag{3.37}$$

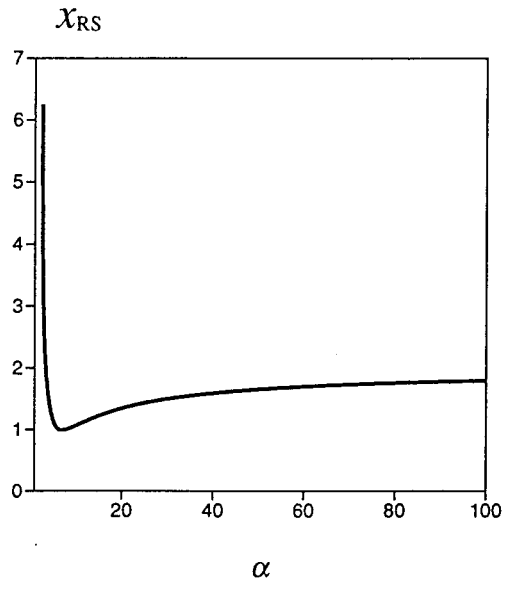

Figure 3.6: The parameter x as a function of α for the case of $a > a_{c0} \sim 1.53$.

As we showed in Chapter 2, $E(R = 1)$ is the minimum value of $\varepsilon(R)$ for this parameter region. Substituting equation (3.35) into equation (3.37), we obtain the learning curve

$$\epsilon_g - \epsilon_{\min} \sim \alpha^{-1}, \tag{3.38}$$

which is identical to the scaling law discovered in the problem of learning disrupted by output noise [17].

(3) $a_{c0} > a > a_{c1}$

A discontinuous transition from the poor generalization phase to the good generalization phase is observed at $\alpha \sim \mathcal{O}(1)$ in this parameter region. In Figures 3.9, 3.10, 3.11 and 3.12, we plot R, x, f and $\epsilon_g = E(R)$ for $a = 1.3$ as functions of α, respectively.

We can observe that there are three solutions for $12.5 < \alpha < \alpha_{sp} \sim 24.2$. For $\alpha < \alpha_{th} \sim 14.7$, the solution which has the smallest R among the three has the lowest free energy and therefore is the globally stable solution. As α is increased beyond α_{th}, the solution which has the largest R becomes the global minimum of free energy. Namely, a thermodynamic phase transition takes place at $\alpha = \alpha_{th}$. Nevertheless, the solution with the smallest R persists until the spinodal point α_{sp} is reached. The solution with the middle R is the local maximum of free energy and represents an unstable solution. A similar transition is also reported by Engel and Reimer in a problem that a non-monotonic perceptron learns the same type of non-monotonic perceptron, although teacher's rule is realizable in their problem [77]. In the limit $\alpha \to \infty$, R approaches $+1$ which achieves the global minimum of the generalization error in this parameter region. The asymptotic behavior of the learning curve is identical to equation (3.38).

(4) $a_{c1} > a > a_{c2}$

The discontinuous transition from the poor generalization phase to the good generalization phase is observed similarly to the previous subsection (3). However, the spinodal point α_{sp} becomes infinity for $a < a_{c1}$, which means that the quasi stable solution beyond α_{th} persists even in the limit $\alpha \to \infty$.

This is easily understood by the following consideration. In thermodynamical systems, physical quantities correspond to the minimum point of the free energy which consists of *energy* and *entropy*. In our system, the energy (3.8) increases with α, although the effect of the entropy in the free energy is not proportional to α. Therefore, in the limit $\alpha \to \infty$, it is expected that properties of the system are determined almost only by the energy. As a result of the central limit theorem, the energy is nearly proportional to

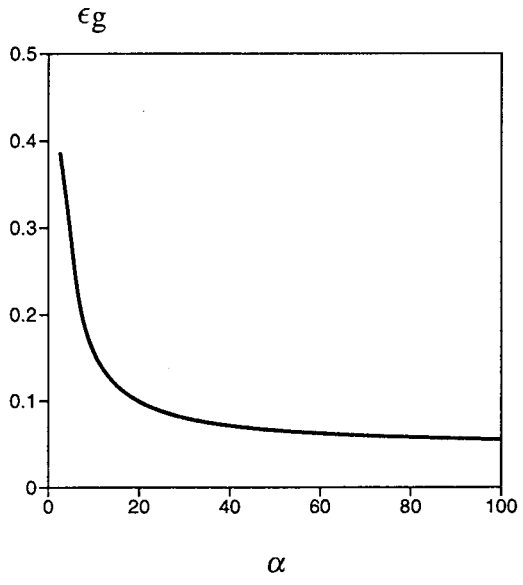

Figure 3.7: Learning curves as a function for the case of $a > a_{c0} \sim 1.53$.

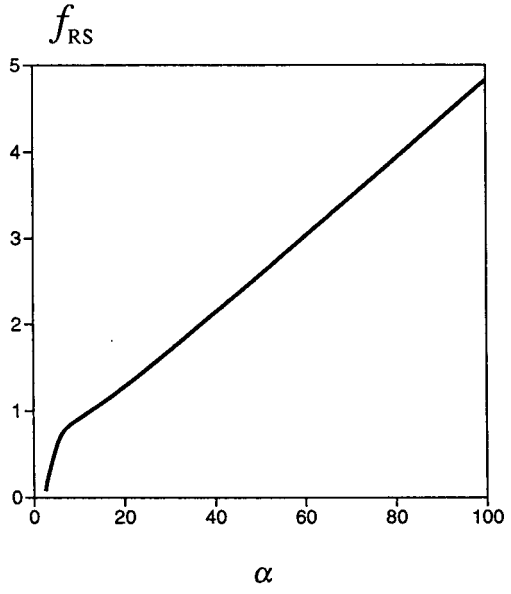

Figure 3.8: Free energy for the case of $a > a_{c0} \sim 1.53$.

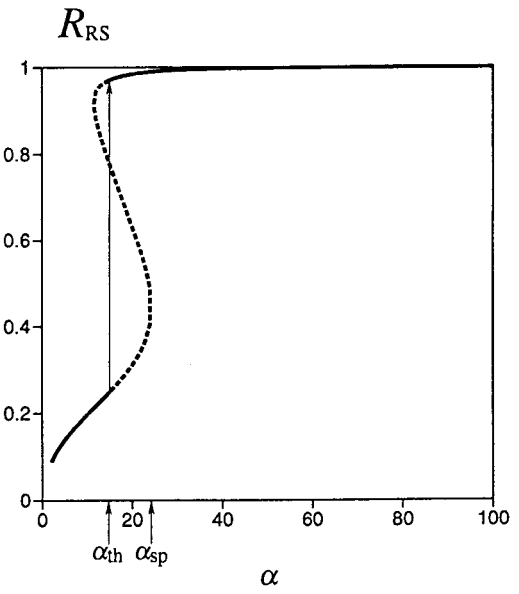

Figure 3.9: The order parameter R for the case of $a_{c0} > a > a_{c1}$ $(a = 1.3)$.

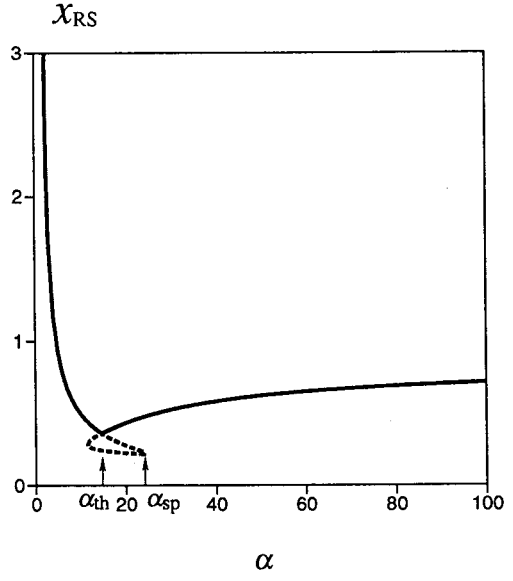

Figure 3.10: The parameter x for the case of $a_{c0} > a > a_{c1}$ $(a = 1.3)$.

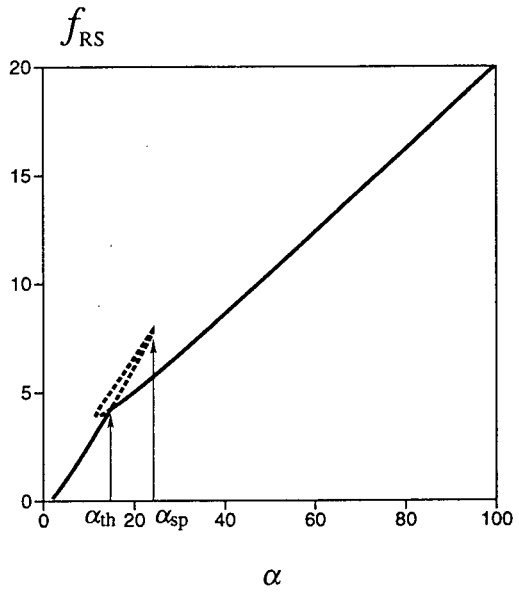

Figure 3.11: Free energy f for the case of $a_{c0} > a > a_{c1}$ $(a = 1.3)$.

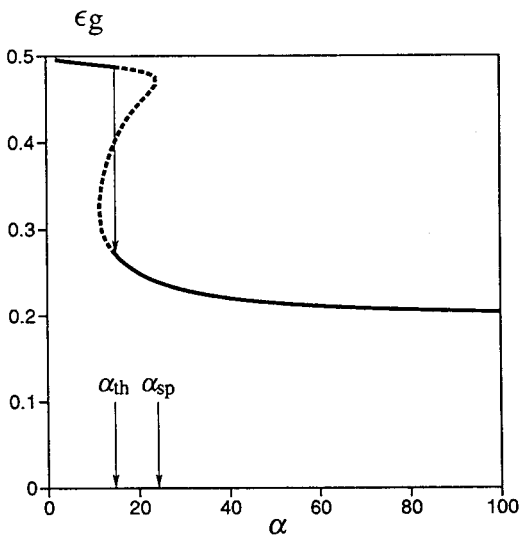

Figure 3.12: Generalization error ϵ_g for the case of $a_{c0} > a > a_{c1}$ $(a = 1.3)$. First order phase transition is observed at $\alpha = \alpha_{th}$. Going beyond this threshold, the student suddenly acquires the excellent generalization ability.

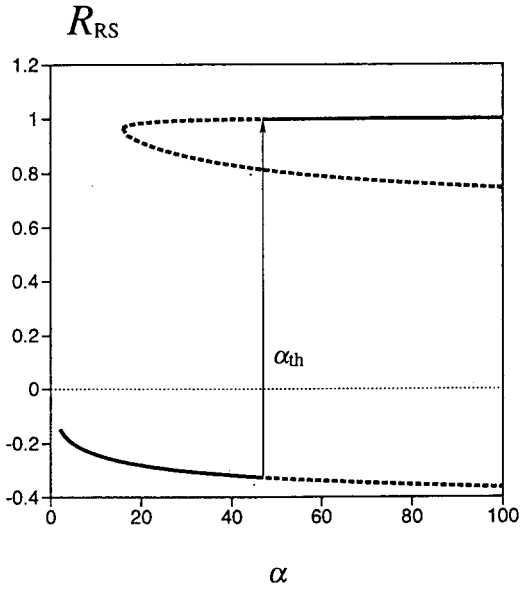

Figure 3.13: The order parameter R for the case of $a_{c1} > a > a_{c2}$ ($a = 1.0$).

the generalization error $E(R)$ for $\alpha \gg 1$. This implies that R obtained from the saddle point equations (3.29) and (3.30) in the limit $\alpha \to \infty$ is identical to the extremal points of $E(R)$. For $a > a_{c1}$, $R = 1$ is the unique minimum point of $E(R)$. Hence, other solutions of the SP equations should disappear as $\alpha \to \infty$ even if they exist for $\alpha \sim \mathcal{O}(1)$, which explains why α_{sp} is finite for $a_{c1} < a < a_{c0}$. On the other hand, for $a_{c2} < a < a_{c1}$, $\varepsilon(R)$ has two extremal points $R = R_-(a)$ and $R = R_+(a)$ besides $R = 1$ which remains the global minimum of $\varepsilon(R)$. This suggests that the SP equations have three solutions in the limit $\alpha \to \infty$ corresponding to the three extremal points of $\varepsilon(R)$, *i.e.* $R = 1$, $R_-(a)$ and $R_+(a)$, which means that α_{sp} is infinity.

The solutions for $a = 1.0$ are plotted in Figures 3.13, 3.14, 3.15 and 3.16. In these figures, we find three solutions which all persist in the limit $\alpha \to \infty$. One solution (solution (I)) starts from $\alpha_c = 2.05$ and reaches the local minimum of $E(R)$ as

$$\text{solution (I)} \qquad \begin{cases} R \to R_-(a), \\ x \to 0, \end{cases} \tag{3.39}$$

in the limit $\alpha \to \infty$.

On the other hand, other two solutions emerge at $\alpha \sim 16$. One of them (solution (II)) approaches the local maximum of $E(R)$ as

$$\text{solution (II)} \qquad \begin{cases} R \to R_+(a), \\ x \to 0, \end{cases} \tag{3.40}$$

in the limit $\alpha \to \infty$. This solution corresponds to the local maximum of the free energy and therefore unstable. The last one (solution (III)) is another (local) minimum of the free energy approaching the global minimum of $E(R)$, $R = 1$ as

$$\text{solution (III)} \qquad \begin{cases} R \to 1, \\ x \to a^2/2, \end{cases} \tag{3.41}$$

in the limit $\alpha \to \infty$.

For $\alpha < \alpha_{th} \sim 47$, the solution (I) is the global minimum of the free energy. As α increases beyond α_c, the solution (III) becomes the global minimum of the free energy, which means that a thermodynamical transition from the solution (I) to the solution (III) takes place at α_{th}. Hence, we obtain $R \to 1$ as $\alpha \to \infty$, which achieves the global minimum of $E(R)$ for this parameter region of a. The asymptotic learning curve of this solution obeys the same power law as equation (3.38).

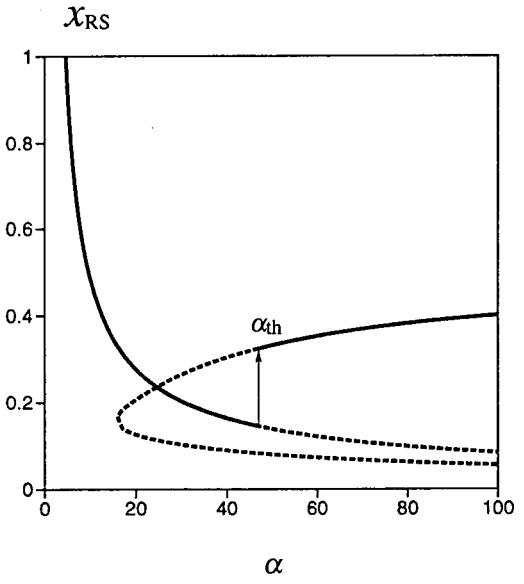

Figure 3.14: The parameter x for the case of $a_{c1} > a > a_{c2}$ $(a = 1.0)$.

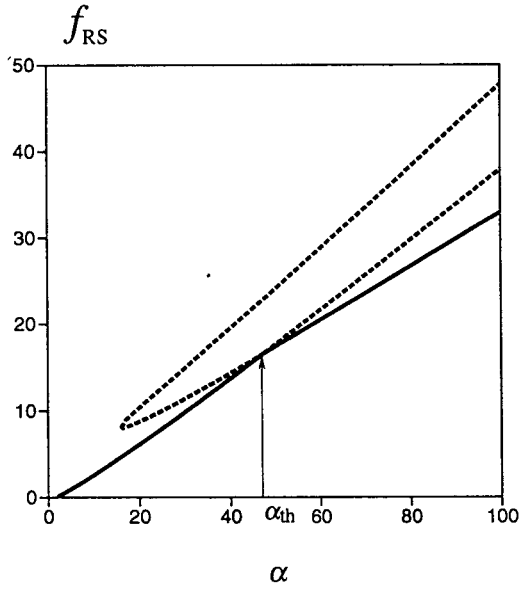

Figure 3.15: Free energy f for the case of $a_{c1} > a > a_{c2}$ $(a = 1.0)$.

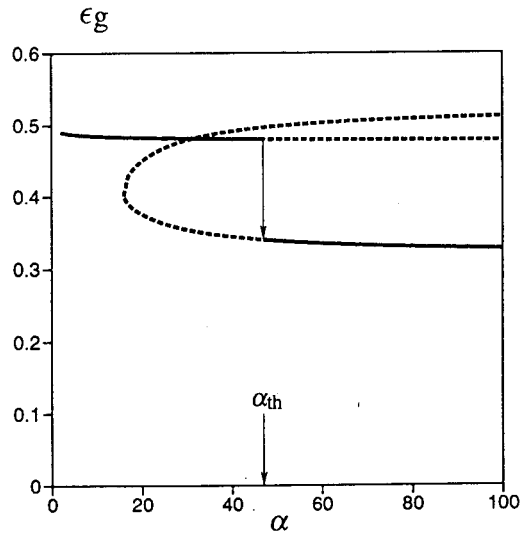

Figure 3.16: Generalization error ϵ_g for the case of $a_{c1} > a > a_{c2}$ $(a = 1.0)$. ϵ_g shows discontinuous phase transition. Spinodal point α_{sp} has gone to infinity.

In addition to the globally stable solution (III), we now have a locally stable solution (I) in the limit $\alpha \to \infty$. R of this solution approaches $R_-(a)$ which locally minimizes $E(R)$. In order to investigate how fast R and x converge as equation (3.39), we expand equations (3.29) and (3.30) with respect to small parameters $\varepsilon = R - R_-(a)$ and x. After some algebra, we obtain the following equations.

$$2\alpha \frac{\Omega_{ttt}(R_-(a):0)}{\sqrt{2\pi}R_-(a)^2}\varepsilon x \sim R_-(a)^2, \tag{3.42}$$

$$2\alpha \frac{x^{3/2}}{3\sqrt{\pi}} \sim 1 - R_-(a)^2, \tag{3.43}$$

which suggest the scalings

$$\varepsilon \sim \alpha^{-1/3}, \tag{3.44}$$

$$x \sim \alpha^{-2/3}. \tag{3.45}$$

From eqs. (2.7) and (2.8), it is found that the relation

$$E(R_-(a) + \varepsilon) - E(R_-(a)) \sim \mathcal{O}(\varepsilon^2), \tag{3.46}$$

holds for small ε. Substituting equation (3.44) into equation (3.46), we obtain the scaling

$$\epsilon_g - \epsilon_{\text{l.min}} \sim \alpha^{-2/3}, \tag{3.47}$$

where $\epsilon_{\text{l.min}} = E(R_-(a))$. It should be remarked that this scaling form is identical to the exponent which was discovered in the problem of learning disrupted by input noise [17], although $\epsilon_{\text{l.min}}$ is not the global but the local minimum of generalization error (Figure: 3.15).

(5) $a_{c2} > a > 0$

In this parameter region, $E(R)$ is minimized not at $R = 1$ but at $R = R_-(a)$. As a result, the solution (I) obtained in the previous subsection (5) remains the global minimum of free energy until $\alpha \to \infty$. As the result, the thermodynamic transition from the solution (I) to the solution (III) disappears and the learning curve decays smoothly to its minimum as

$$\epsilon_g - \epsilon_{\text{min}} \sim \alpha^{-2/3}, \tag{3.48}$$

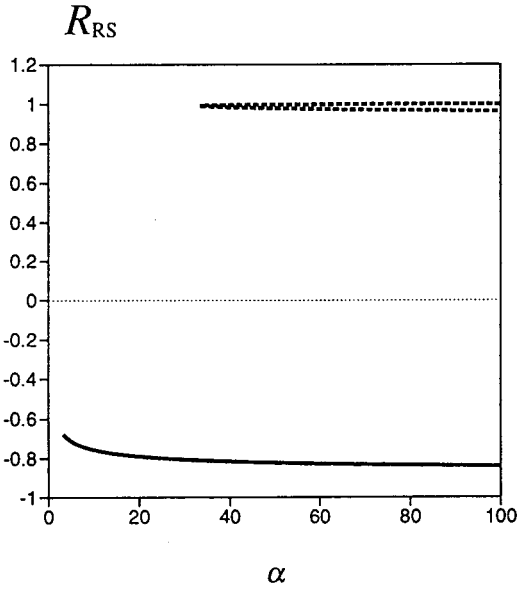

Figure 3.17: The order parameter R for the case of $a_{c2} > a > 0$ ($a = 0.5$).

where $\epsilon_{\min} = E(R_{-}(a))$ is the minimum value of $E(R)$ in this parameter region. We plot the order parameters $R_{\rm RS}$, $x_{\rm RS}$, the generalization error ϵ_g and the free energy f_{RS} in Figures 3.13, 3.14, 3.15 and 3.16.

Remark that equation (3.48) suggests that the exponent of the learning curve in the limit $a \to 0$ is different from that of $a = 0$ (realizable case) in equation (3.32). In contrast, as for the other realizable case $a = \infty$, the exponents of learning curves for $a \to \infty$ are the same as that of $a = \infty$. This implies that the non-monotonic teacher with small a is more difficult for a simple perceptron to learn than that with large a.

3.3 Simulations for the two-dimensional case

In this section, we discuss the validity of the results obtained under the RS ansatz in the previous section. Firstly, we comment on the critical values of a, *i.e.* $a_{\rm c0}$, $a_{\rm c1}$ and $a_{\rm c2}$. $a_{\rm c0} \sim 1.53$ is the point below which a discontinuous transition appears in the learning curve. This value is intrinsic of the RS ansatz and therefore will be changed if we proceed to RSB calculations. However, we conjecture that $a_{\rm c1}$, which is defined as the point below which $\alpha_{\rm sp}$ becomes infinity, and $a_{\rm c2}$, which is defined as the point below which $\alpha_{\rm th}$ becomes infinity, will be unchanged by RSB calculations because they result from changes in the shape of $\varepsilon(R)$ which is independent of the ansatz on the replica calculations.

Secondly, we mention the critical values of α, *i.e.* $\alpha_{\rm c}$, $\alpha_{\rm th}$ and $\alpha_{\rm sp}$. $\alpha_{\rm c}$ is the point at which $q \to 1$. In our case, this value is identical to $\alpha_{\rm AT}$ beyond which the RS solution becomes unstable (see Appendix B). Therefore, this is invariant if we take the RSB into account. However, α_{th} and $\alpha_{\rm sp}$ will be changed by RSB calculations because they are intrinsic of the RS solution.

Finally, we discuss the asymptotic behaviors of learning curves. In the previous section, we found two types of asymptotic learning curves with exponents 1 and 2/3 for unrealizable cases. Although they are unstable RS solutions, the two exponents 1 and 2/3 are consistent with those obtainable without using the replica method in a two-dimensional version of the present problem as follows. This suggests that our results are good approximations even if they are not exact.

Let us consider the following two-dimensional learning problem. In this problem, the teacher is a two-dimensional non-monotonic perceptrons with the weight $\boldsymbol{B} = (B_1, B_2)$ which returns the output

$$y = T_a(\boldsymbol{B} \cdot \boldsymbol{x}), \tag{3.49}$$

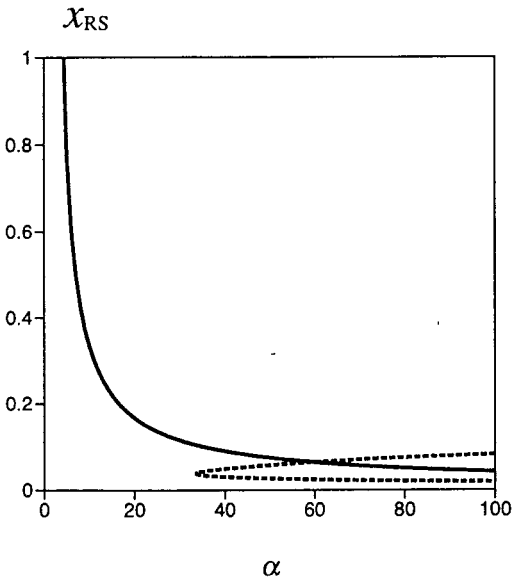

Figure 3.18: The parameter x for the case of $a_{c2} > a > 0$ ($a = 0.5$).

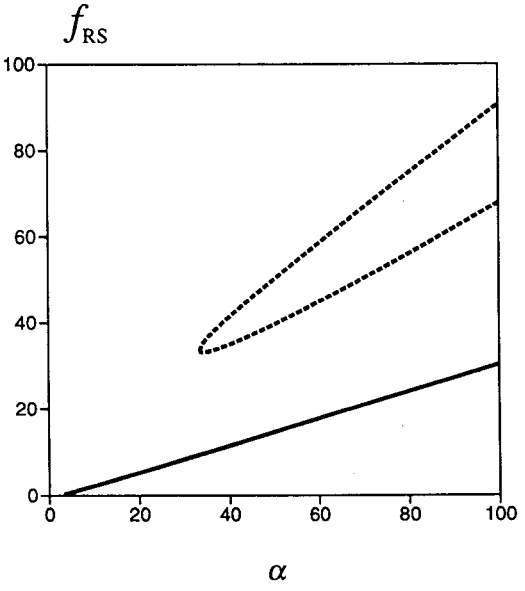

Figure 3.19: Free energy f for the case of $a_{c2} > a > 0$ ($a = 0.5$).

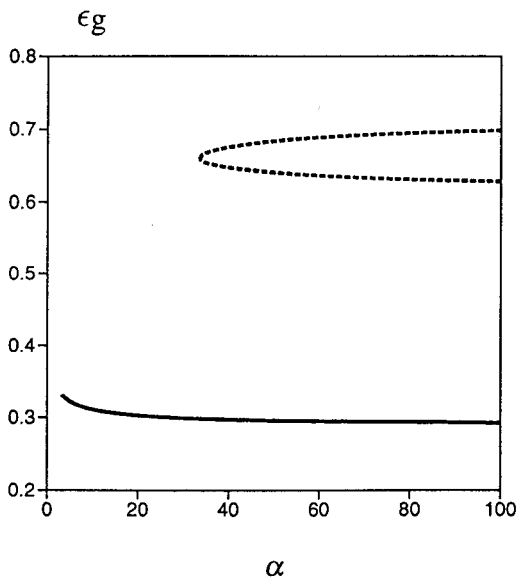

Figure 3.20: Generalization error ϵ_g for the case of $a_{c2} > a > 0$ ($a = 0.5$).

for a two-dimensional input $\boldsymbol{x}$. Here, $T_a(x)$ is defined as $T_a(x) = \mathrm{sign}[x(a-x)(a+x)]$ and it is assumed that $|\boldsymbol{B}| = 1$. On the other hand, the student is a two-dimensional simple perceptron with weight $\boldsymbol{J} = (J_1, J_2)$ (Figure 3.21). In order to acquire a good generalization ability, this student learns from a given set of examples $\xi^P = \{(\boldsymbol{x}_\mu, y_1); \mu = 1, \cdots, P\}$ in which the inputs $\boldsymbol{x}_\mu$ are assumed to be independently and identically drawn from the two-dimensional Gaussian distribution $\exp[-({x_1}^2 + {x_2}^2)/2]/(2\pi)$, following the error minimum algorithm (Figure 3.21).

From the two-dimensional nature of the problem, the system can be specified by a single parameter ϕ which is the angle between $\boldsymbol{B}$ and $\boldsymbol{J}$. In Figure 3.22, we plot the number of false predictions on a realization of ξ^P, E_P, versus ϕ together with its expectation $< E_P(\phi) >= P \times \varepsilon(\phi)$ for $a = 1.0$. We should notice that $\varepsilon(\phi)$ is calculated analytically. Let us consider the conditional probability $P_T(+1|\theta)$ which means the probability that the teacher output takes $+1$ on condition that the angle between the output $\boldsymbol{x}$ and the teacher weight $\boldsymbol{B}$ is θ. The $P_T(+1|\theta)$ is easily calculated as

$$P_T(+1|\theta) = 2 - 2H\left(\frac{a}{\cos\theta}\right) - \Theta(\cos\theta). \tag{3.50}$$

It is obvious that $P_T(-1|\theta) = 1 - P_T(+1|\theta)$. For the same inputs $\boldsymbol{x}$, the probability for the student is given as

$$P_S(+1|\theta) = \Theta(\cos(|\theta - \phi|)), \tag{3.51}$$

and $P_S(-1|\theta) = 1 - P_S(+1|\theta)$. Therefore, $\varepsilon(\phi)$ is calculated as

$$\varepsilon(\phi) = \frac{1}{\pi}\int_0^\pi d\theta\,[P_T(+1|\theta)P_S(-1|\theta) + P_T(-1|\theta)P_S(+1|\theta)]\,. \tag{3.52}$$

Here, $\varepsilon(\phi)$ is the generalization error as a function of ϕ. In the figure, we only plot the graphs for positive ϕ because these graphs are statistically symmetric under the reverse operation $\phi \leftrightarrow -\phi$.

From this figure, it is found that $E_P(\phi)$ takes a global minimum at $\phi = 0$. At the same time, $E_P(\phi)$ is locally minimized around $\phi = \phi_* \sim 2.13$, which is the local minimum point of $\varepsilon(\phi)$. Let us estimate how these (local) minimum points fluctuate around $\phi = 0$ or $\phi = \phi_*$ by the following consideration. A similar method was once applied to explain the asymptotic learning curve of a stochastic learning problem [18].

$E_P(\phi)$ is the number of examples which satisfy the condition that $y_\mu = 1$ and $\boldsymbol{B} \cdot \boldsymbol{x}_\mu < 0$ or $y_\mu = -1$ and $\boldsymbol{B} \cdot \boldsymbol{x}_\mu > 0$ ($\mu = 1, 2, \cdots, P$). First, we evaluate how the expectation of $E_P(\phi)$ increases around $\phi = 0$, and $\phi = \phi_*$. From Figure 3.22, we find that this increases as

$$< E_P(\phi) - E_P(0) > \sim P \times |\phi|, \tag{3.53}$$

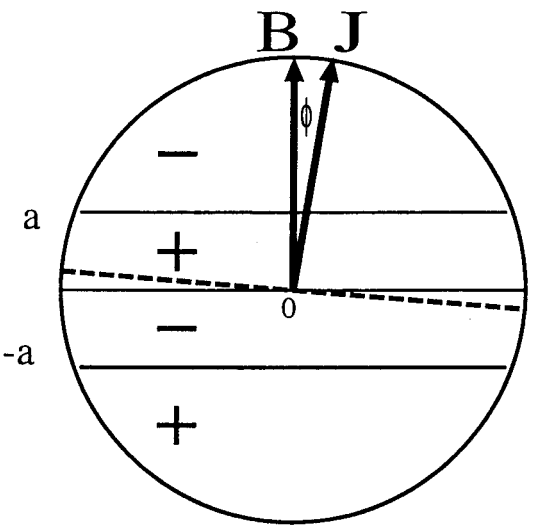

Figure 3.21: The two-dimensional learning problem. From the two-dimensional nature of the problem, the system is specified by a single angle ϕ

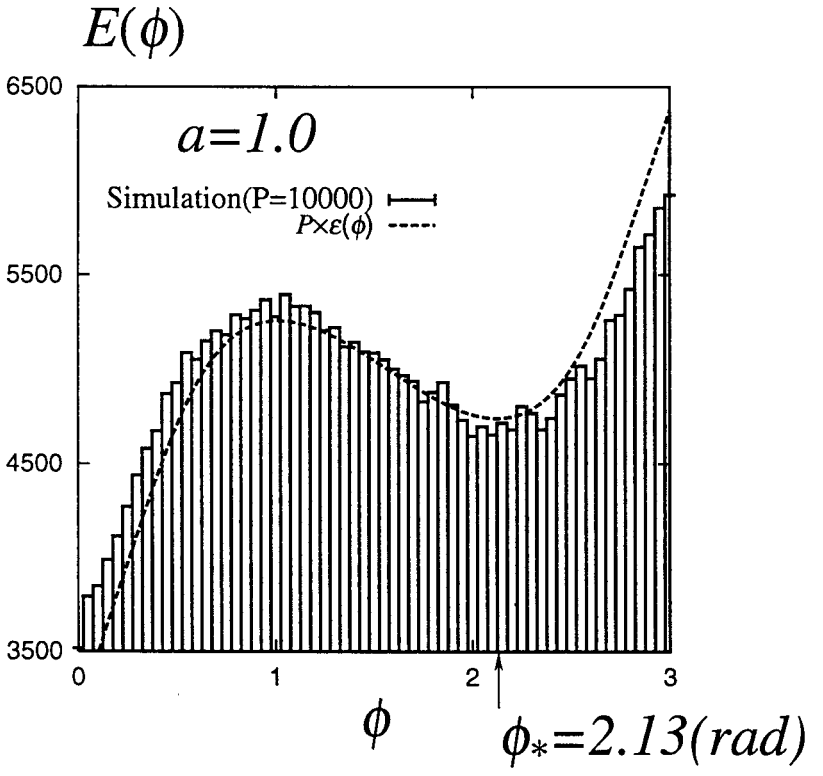

Figure 3.22: $E_P(\phi)$ for a realization of an example set ξ^P. This graph is for $a = 1.0$ and $P = 10000$. The broken line stands for the expectation $< E_P(\phi) >= P \times \varepsilon(\phi)$, where $\varepsilon(\phi)$ is the generalization error.

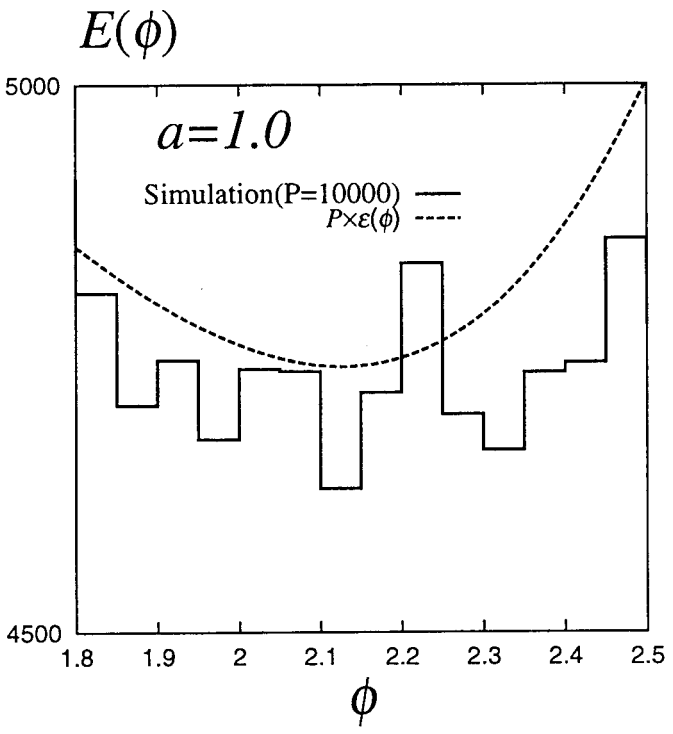

Figure 3.23: Around $\phi \sim \phi_*$.

around $\phi = 0$. On the other hand, $< E_P(\phi) >$ quadratically increases around the local minimum $\phi = \phi_*$, as

$$< E_P(\phi) - E_P(\phi_*) > \sim P \times (\phi - \phi_*)^2. \tag{3.54}$$

Next, we estimate the fluctuation of eqs. (3.53) and (3.54). Suppose ϕ moves from $\phi = 0$ to $\phi = \pi$. Every time decision the boundary of $\boldsymbol{J}$, namely, borderline which is perpendicular to $\boldsymbol{J}$ comes across an input $\boldsymbol{x}_\mu$ $(1 < \mu < P)$, $E_P(\phi)$ increases or decreases discontinuously by 1. Around $\phi = 0$, $E_P(\phi)$ almost always increases because positive and negative examples are clearly separated around the boundary $\boldsymbol{B} \cdot \boldsymbol{x} = 0$. This means that the fluctuation of equation (3.53) is very small and the minimum point fluctuates of the order of P^{-1} which is a rough estimate of width between two neighboring examples. Therefore, we obtain

$$\epsilon_g - \epsilon_{\min} \sim \varepsilon(\phi) - \varepsilon(0) \sim |\phi| \sim P^{-1}, \tag{3.55}$$

which has the same exponent 1 as that of equation (3.38).

In contrast, $E_P(\phi)$ increases or decreases almost randomly around $\phi = \phi_*$. This motion of $E_P(\phi)$ is analogous to that of a *random walk* if we regard ϕ as *time*. From this analogy, we obtain the following relation with respect to the fluctuation of equation (3.54)

$$\Delta(E_P(\phi) - E_P(\phi_*)) \sim \sqrt{P \times |\phi - \phi_*|}. \tag{3.56}$$

The balance between equations (3.54) and (3.56), namely, the following condition

$$P \times (\phi - \phi_*)^2 = \sqrt{P \times (\phi - \phi_*)} \tag{3.57}$$

determines the fluctuation of the (local) minimum point of $E_P(\phi)$ around $\phi = \phi_*$. This gives the scaling $|\phi - \phi_*| \sim P^{-1/3}$, which yields the learning curve

$$\epsilon_g - \epsilon_{\text{l.min}} \sim \varepsilon(\phi) - \varepsilon(\phi_*) \sim (\phi - \phi_*)^2 \sim P^{-2/3}, \tag{3.58}$$

which shares the same exponent 2/3 with equations (3.47) and (3.48).

In order to confirm the above discussion, we performed numerical experiments for $a = 1.0$ (large a) where $\phi = 0$ is the global minimum of $\varepsilon(\phi)$, and for $a = 0.1$ (small a) where $\phi = \phi_* = 2.13$ globally

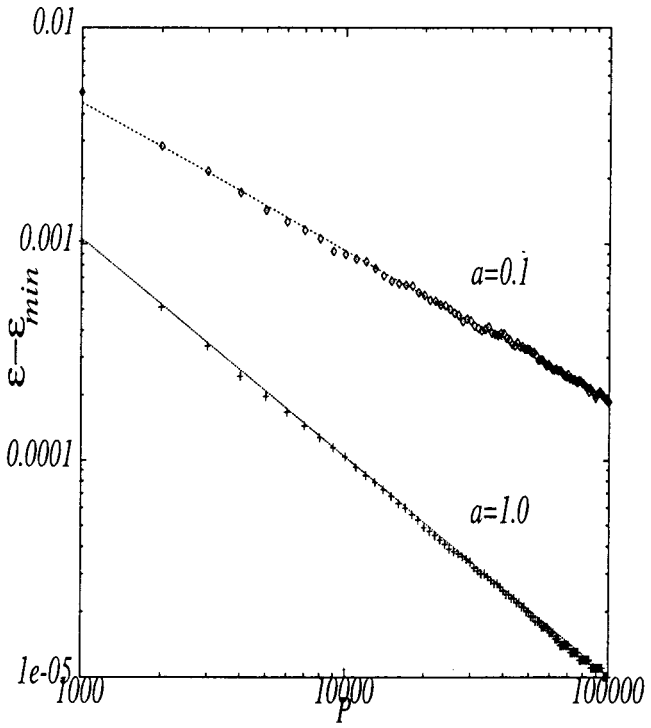

Figure 3.24: Scaling relations $\epsilon_g - \epsilon_{min} \sim \mathcal{O}(P^{-\gamma})$. $\gamma = 0.68 \pm 0.02$ ($a = 0.1$), $\gamma = 1.01 \pm 0.01$ ($a = 1.0$).

minimizes $\varepsilon(\phi)$. The numerically obtained data exhibit the following behaviors. As the number of examples P increases, the parameter obtained by learning converges to the global minimum of $\varepsilon(\phi)$, namely, to 0 for $a = 1.0$ and to ϕ_* for $a = 0.1$. The average of the generalization error ε taken over 1000 sets of examples is plotted in Figure 3.24. This figure indicates scaling relations $\epsilon_g - \epsilon_{\min} \sim \mathcal{O}(P^{-\gamma})$ for both cases. The exponents obtained from the least square method are $\gamma = 1.01 \pm 0.01$ for $a = 1.0$ and $\gamma = 0.68 \pm 0.02$ for $a = 0.1$, which are highly consistent with our theoretical predictions $\gamma = 1$ for large a and $2/3$ for small a.

3.4 Summary

The results obtained in this chapter are summarized as follows. It is clear that our non-monotonic perceptron is realizable for the two limiting values of a, $a = 0$ and $+\infty$. In these two special cases, the learning curve obeys the scaling law

$$\epsilon_g \sim \alpha^{-1}. \tag{3.59}$$

Except for these values of a, the behavior of learning is found to be classified into the following four categories depending on a: For $a > a_{c0} \sim 1.53$, the learning curve smoothly decays to its minimum and its asymptote obeys the relation

$$\epsilon_g \sim \epsilon_{\min} + \alpha^{-1}. \tag{3.60}$$

For $a_{c0} > a > a_{c1} = 1.17$, a discontinuous transition from the poor generalization phase to the good generalization phase takes place at some value of $\alpha = \alpha_{\rm th} \sim \mathcal{O}(1)$ and the quasi stable solution disappears at the spinodal point $\alpha = \alpha_{\rm sp} > \alpha_{\rm th}$. The asymptotic learning curve has the form of equation (3.60). For $a_{c1} > a > a_{c2} = 0.8$, the discontinuous transition from the poor generalization phase to the good generalization phase also takes place at some value of $\alpha = \alpha_{\rm th} \sim \mathcal{O}(1)$. However, the spinodal point $\alpha_{\rm sp}$ becomes infinity and the quasi stable solution persists even in the limit $\alpha \to \infty$. This quasi stable

solution exhibits the slow convergence;

$$\epsilon_g \sim \epsilon_{\min} + \alpha^{-2/3} \tag{3.61}$$

in the asymptotic region $\alpha \gg 1$, although the asymptotic form of the globally stable solution obeys eq. (3.60). For $a_{c2} > a > 0$, the discontinuous transition disappears and the learning curve decays to its minimum smoothly exhibiting the slow convergence (3.61) in the asymptotic region. These results suggest that the scaling relations obtained in the problems of learning from noisy examples [17] generally appear in the problem of learning unrealizable rules as well. We should also address that the globally stable solution obtained by the minimum error algorithm realizes the optimal generalization error in the limit $\alpha \to \infty$ for an arbitrary a.

The above results are obtained by using the replica method under the replica symmetric (RS) ansatz. Unfortunately, it is known that the RS solution of the zero-temperature learning with the Gardner-Derrida cost function becomes thermodynamically unstable when the teacher's rule is unrealizable [24, 22, 62]. Furthermore, it is conjectured that any finite step of replica symmetry breaking is not sufficient to obtain a thermodynamically stable solution [63]. Nevertheless, we have a conjecture that our results offer a good approximation at least qualitatively because the same exponents of asymptotic learning curves, 1 and 2/3, are also obtainable without using the replica method in a two-dimensional version of our learning model.

Chapter 4

On-Line Learning

4.1 Background

In the previous chapter, we discussed the learning properties of our model system by the off-line learning scenario. As we mentioned in detail, in the off-line learning mode, the student network does not change his weight vector $\boldsymbol{J}$ until the cost is minimized so that statistical mechanical equilibrium with respect to the dynamical variable $\boldsymbol{J}$ is obtained. The cost function has a lot of local minima in proportion to the $P = \alpha N \sim \mathcal{O}(N)$ examples and it takes tremendous times ($\sim \mathcal{O}(N) \to \infty$ in the thermodynamical limit $N \to \infty$) to minimize the cost. It is hard for us to apply the memory-based off-line learning algorithms to the situations in engineering, for example, robotics [66, 67], signal processing like an *Independent Component Analysis(ICA)* [68, 69, 70, 71] or other adaptive systems. In these practical situations, the student machine should adapt his own parameters to the environment (in our context, the weight of the teacher or threshold) which changes slowly or quickly. For such cases, it is impossible for the student to gather and memorize a set of examples and changes his parameter a little bit every time the environment changes. It is also hard to suppose that the fashion of information processing in real brain is in the off-line mode.

In general, we should use two different criteria in estimating the performance of learning, namely, *sample complexity* and *time complexity.*

In short, these two criteria are summarized as follows.

Two criteria of success in learning

Sample complexity The number of necessary examples to obtain some generalization ability.

Time complexity Computational times for a specific training.

Until the 1990s, most of the researches with respect to the generalization of machine learning have traditionally been devoted to the off-line learning mode considering the sample complexity only. In this context, the time complexity has been neglected. Taking the time complexity into account, recent progress in this field has been directed toward the on-line learning scheme where the weights of the student are updated after presentation of each example. The current weights are modified at the next step of learning according to the difference between the present student output and the teacher signal. During learning processes, an example is used only once and not repeatedly presented. Such a learning process can be regarded as dynamics if we identify the number of presented examples with time.

The foundation of on-line learning was established in 1967 by Amari [3]. He discussed the learning algorithm based on the stochastic descent and investigated the distribution of the weight vector and estimated the mean value and standard deviation of the weight around the target vector [3, 72]. Recently, Heskes and Kappen [74] succeeded in investigating the learning processes in the on-line mode by representing the stochastic process of weight vector by the macroscopic order parameter which is well-defined

in the thermodynamical limit, namely, $N \to \infty$ and $P \to \infty$.

Theoretical investigation of the on-line learning has the advantage of mathematical simplicity. In on-line learning, the typical behavior of the learning process and the generalization error are described by a few order parameters.

Using this on-line learning scheme, a lot of interesting learning problems have been investigated, such as multi-layer neural networks (committee machine [21, 78] and parity machine [80]), learning from noisy data (output noise [81, 82] and input noise [82]), learning from clustered input examples [83], and learning from a time-dependent rule [82]. Recently, Barkai, Seung and Sompolinsky [81] investigated the on-line learning of a single-layer perceptron from noisy examples in which the teacher signal is inverted at a rate of p for each example (output noise). For this case they concluded that the optimal prediction error behaves asymptotically as $\epsilon_g \sim p+(2p/\pi\alpha)^{1/2}$, where α is the number of presented examples per size of the input space. Kim and Sompolinsky [84, 85, 86] reported that the prediction error decays as $\epsilon_g \sim p+A\alpha^{-1}$ by the on-line Gibbs algorithm. They also investigated the case of the on-line Gibbs learning with input noise and concluded that the generalization error converges to its minimum value as $\epsilon_g \sim \epsilon_{\min} + A\alpha^{-1/2}$. The effects of noise have also been investigated for parity machines [80] and optimized learning [82]. In these studies, the teacher and the student networks have the same structure, and the student cannot learn the rule perfectly due to output noise.

In this chapter we study the learning properties of our model system using the on-line learning algorithm. For the realizable case ($a \to \infty$), several authors reported that the asymptotic form of the learning curve given by the on-line algorithm is $\epsilon_g(\alpha) \sim \alpha^{-1/3}$ for the perceptron learning [81], $\epsilon_g(\alpha) \sim \alpha^{-1/2}$ for the Hebbian learning [82] and $\epsilon_g \sim \alpha^{-1}$ for the AdaTron learning [98].

On the other hand, it is still an open question how the learning curve depends on the width a of the reversed wedge and how difficult it is for the student to predict the teacher rule. We investigate these problems under various conditions in this chapter.

This chapter is composed of nine sections. The perceptron, Hebbian and AdaTron learning algorithms in the on-line scheme are investigated in the next section 4.2. For each learning scheme, we calculate the asymptotic behavior of the learning curve in the realizable limit $a \to \infty$ as well as for the unrealizable case of finite a. In section 4.3 we investigate the effects of output noise on learning processes. In sections 4.4 and 4.5 we introduce the optimal learning rate for the student and calculate the optimal generalization error. The optimal learning rate obtained in section 4.4 contains an unknown parameter for the student, namely, the width a of the reversed wedge of the teacher transfer function. This is somewhat contradictory to the idea of learning because the learning process depends upon the unknown teacher parameter. Therefore, in section 4.6 we introduce a learning rate independent of the unknown parameter and optimize the rate to achieve a faster convergence of the generalization error. In section 4.7, we allow the student to ask queries under the Hebbian learning algorithm. It is shown that learning is accelerated considerably if the learning rate is optimized. In section 4.8 we optimize the learning dynamics by a weight-decay term to avoid an over-training problem in the Hebbian learning for some parameter range observed in section 4.2. The last section contains summary and discussions.

4.2 Dynamics of noiseless learning

We now investigate the learning dynamics with specific learning rules. The noiseless case with constant learning rate is treated in this section.

4.2.1 Perceptron learning

We first investigate the perceptron learning

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m - \Theta(-T_a(v)S(u))\,\mathrm{sign}(u)\,\boldsymbol{x} \tag{4.1}$$

where Θ is the step function and m stands for the discrete time step of dynamics or the number of presented examples. The synaptic weight vector of the student is modified by the amount $-\mathrm{sign}(u)\boldsymbol{x}$ if his answer is incorrect. It is straightforward to obtain the recursion equations for the overlap $R^m = (\boldsymbol{J}^m \cdot \boldsymbol{B})/|\boldsymbol{J}^m||\boldsymbol{B}|$ and the length of the student weight vector $l^m = |\boldsymbol{J}^m|/\sqrt{N}$. In the limit $N \to \infty$,

these two dynamical quantities become self-averaging with respect to the random training data $\mathbf{x}$. For continuous time $\alpha = m/N$ in the limit $N \to \infty$, $m \to \infty$ with α kept finite, the evolutions of R and l are given by the following differential equations:

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{E(R)}{2} - F(R)l\right] \tag{4.2}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2}E(R) + (F(R)R - G(R))\,l\right] \tag{4.3}$$

where $E(R) = \ll 1 \gg_R$, $F(R) = \ll u\,\mathrm{sign}(u) \gg_R$ and $G(R) = \ll v\,\mathrm{sign}(u) \gg_R$. The brackets $\ll \cdots \gg_R$ stand for the averaging with respect to the distribution P_R

$$\ll \cdots \gg_R = \int\int_\Omega du dv (\cdots) P_R(u,v). \tag{4.4}$$

In Appendix C, we explain the derivation of the above differential equations with respect to l and R for the general weight function $f(T_a(v), u)$. In Appendix C, we explain the derivation of the above differential equations with respect to l and R for the general weight function $f(T_a(v), u)$.

Hence the definition of $E(R)$ coincides with that of the generalization error, $E(R) = \epsilon_g$, as used in the previous section. The other quantities $F(R)$ and $G(R)$ are evaluated in a straightforward manner as

$$F(R) = -\frac{R}{\sqrt{2\pi}}(1 - 2\Delta) + \frac{1}{\sqrt{2\pi}} \tag{4.5}$$

and

$$G(R) = -\frac{1}{\sqrt{2\pi}}(1 - 2\Delta) + \frac{R}{\sqrt{2\pi}} \tag{4.6}$$

where $\Delta = \mathrm{e}^{-a^2/2}$.

Numerical analysis of differential equations

We have numerically solved equations (4.2) and (4.3). The resulting flows of R and l are shown in Figure 4.1 for $a = \infty$ under several initial conditions. This figure indicates that R reaches 1 (perfect generalization state) in the limit of $\alpha \to \infty$ and $l \to \infty$ for any initial condition. For finite α, however, behavior of the flow strongly depends on the initial condition. If we take a large l as the initial value, the perfect generalization state ($R = 1$) is achieved after l decreases at intermediate steps. As l represents stiffness against change of the direction of the student vector, approach to the perfect generalization state cannot start until the stiffness decreases to some extent. If we choose initial R close to 1 and small l, the perfect generalization is achieved after a decrease of R is observed. Similar phenomena have been reported in the $K = 2$ parity machine [80]. The generalization error is shown later in Figure 4.4 as a function of α for the initial condition $(R, l) = (0.01, 0.10)$.

Next we display the flows of R and l for unrealizable cases, for example, $a = 2.0$ and $a = 0.5$ in Figures 4.2 and Figure 4.3 respectively. It is seen that there exists an a-dependent fixed point (R_0, l_0) and the flow is attracted into this fixed point. The generalization of the student halts at this fixed point even if the flow of R and l starts from $R = 1$ and large l. In Figure 4.4 we plot the generalization error for these unrealizable cases. This figure suggests that the generalization error converges to a finite value $\epsilon_{\min}$ very rapidly, presumably exponentially, if a is finite. In the next subsection we investigate the asymptotic behavior in detail to confirm these qualitative observations.

Asymptotic analysis of the learning curve

Let us first check the asymptotic form of the generalization error for the realizable case ($a = \infty$) [87]. As the order parameter R increases monotonically near $R = 1$ as seen in Figure 4.1, we scale the parameters as $R = 1 - \varepsilon$ and $l = 1/\delta$ in order to see the behavior of the differential equations (4.2) and (4.3) for small ϵ and δ in the limit $\alpha \to \infty$. Then $E(R)$, $F(R)$ and $G(R)$ are scaled as $E(\epsilon) = \epsilon_g \sim \sqrt{2\varepsilon}/\pi$, $F(\epsilon) \sim \varepsilon/\sqrt{2\pi}$

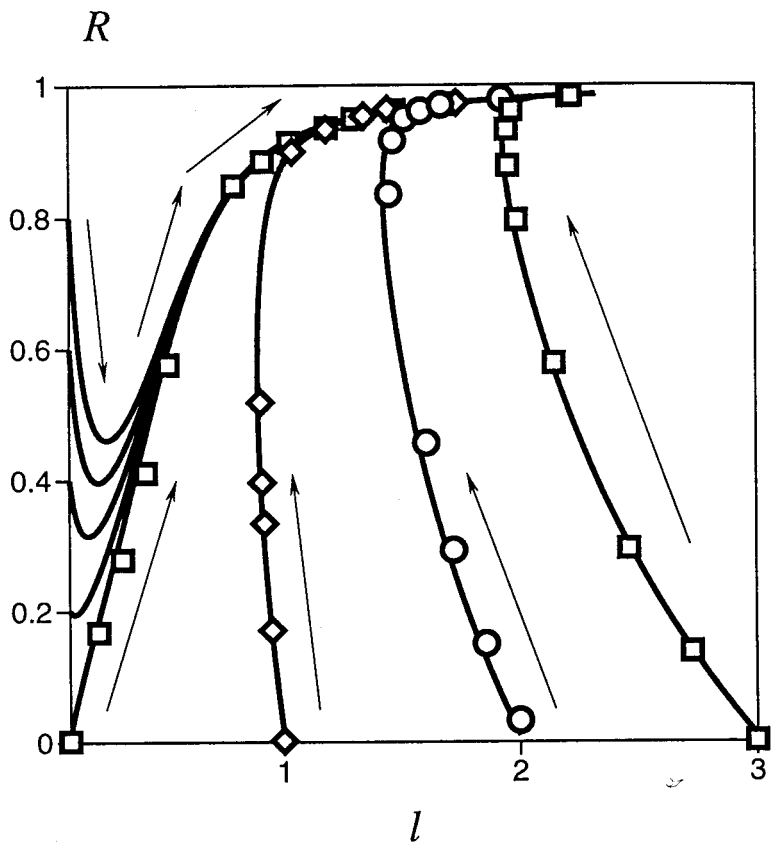

Figure 4.1: Flows of the order parameter R and l for the realizable case ($a = \infty$) by the perceptron learning. If one starts from large l, the student begins to generalize after the length of the weight vector l decreases to some value. White points are the results of the computer simulations for the system size $N = 2000$. In each time step, the student receives $P = 100$ examples. The results of theory and simulation show the excellent matches.

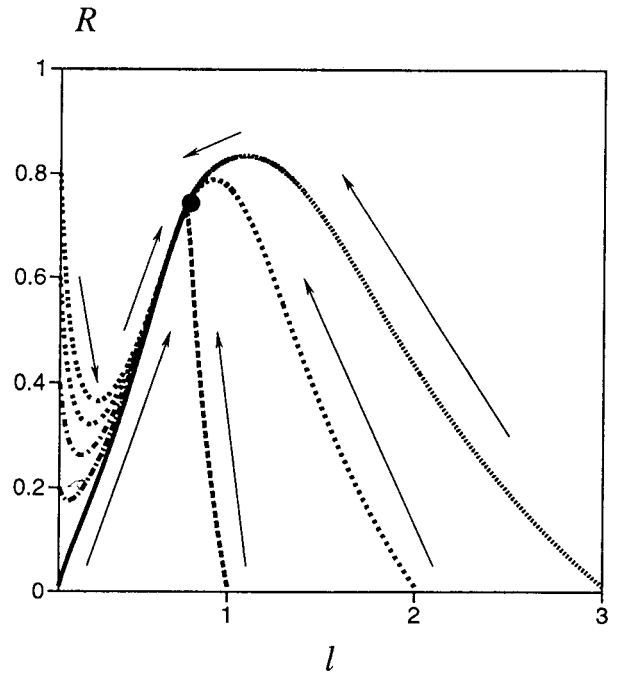

Figure 4.2: Flows of the order parameters R and l for the unrealizable case ($a = 2.0$). The flows are attracted to a fixed point.

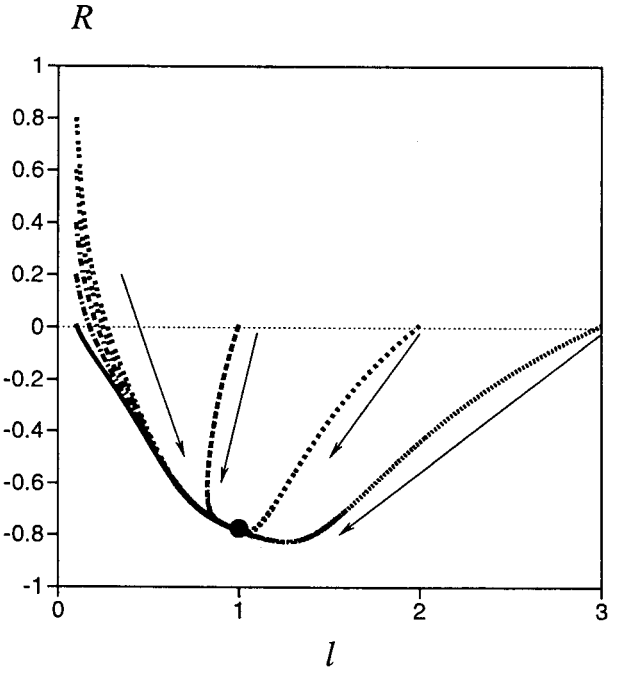

Figure 4.3: Flows of the order parameters R and l for the unrealizable case ($a = 0.5$). The flows are attracted to a fixed point.

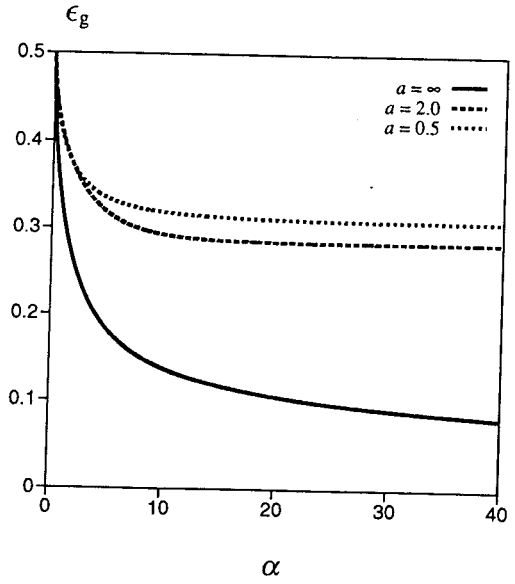

Figure 4.4: Generalization error ϵ_g for the learnable case ($a = \infty$) and for the two unlearnable cases ($a = 2.0, 0.5$) by the perceptron learning. The learning curves for the cases of $a = \infty$ and $a = 0$ perfectly agree each other. For the unlearnable case, the generalization error converges to an a-dependent finite value.

and $G(\epsilon) \sim -\varepsilon/\sqrt{2\pi}$. These scaling forms are easily obtained. For example, the scaling form of $E(R)$ is calculated as follows. We first define $E_1(R)$ and $E_2(R)$ as

$$\begin{aligned} E(R) &= \int_0^a Dv \,\mathrm{erfc}\left(\frac{Rv}{\sqrt{2(1-R^2)}}\right) + \int_a^\infty Dv \,\mathrm{erfc}\left(\frac{-Rv}{\sqrt{2(1-R^2)}}\right) \\ &\equiv E_1(R) + E_2(R) \end{aligned} \tag{4.7}$$

Using the scaling relation $R = 1 - \varepsilon$ and $\varepsilon \to 0$, E_1 is rewritten as

$$\begin{aligned} E_1(\varepsilon) &= \int_0^{\frac{a}{2\sqrt{\varepsilon}}} \frac{dt}{\sqrt{2\pi}} 2\sqrt{\varepsilon}\,\mathrm{erfc}(t) \\ &= \frac{2\sqrt{\varepsilon}}{\sqrt{2\pi}} \left\{ [t\,\mathrm{erfc}(t)]_0^{\frac{a}{2\sqrt{\varepsilon}}} + \int_0^{\frac{a}{2\sqrt{\varepsilon}}} t\,dt e^{-\frac{t^2}{2}} \right\} \\ &\sim \frac{\sqrt{2\varepsilon}}{\pi}. \end{aligned} \tag{4.8}$$

By similar calculations, E_2 is rewritten as $E_2(\varepsilon) \sim 2H(a)$ and this goes to zero in the limit of $a \to \infty$. From these two results of E_1 and E_2, we obtain the scaling form with respect to $E(R)$ as $E(\varepsilon) \sim 2H(a) + \sqrt{2\varepsilon}/\pi \sim \sqrt{2\varepsilon}/\pi$ $(a \to \infty)$.

Substituting these asymptotic expressions into the differential equations (4.2) and (4.3), we obtain

$$\frac{d\delta}{d\alpha} = -\frac{\sqrt{2}}{2\pi}\sqrt{\varepsilon}\delta^3 + \frac{\varepsilon}{\sqrt{2\pi}}\delta^2 \tag{4.9}$$

$$\frac{d\varepsilon}{d\alpha} = \frac{\sqrt{2}}{2\pi}\sqrt{\varepsilon}\delta^2 - \sqrt{\frac{2}{\pi}}\varepsilon\delta. \tag{4.10}$$

Taking the ratio of these two equations, we find

$$\frac{d\delta}{d\varepsilon} = \frac{-\delta + \sqrt{\pi\varepsilon}}{\delta - 2\sqrt{\pi\varepsilon}}\delta. \tag{4.11}$$

If we assume $\delta \gg \sqrt{\varepsilon}$ or $\delta \ll \sqrt{\varepsilon}$, the above differential equation leads to $\delta \sim \mathrm{e}^{-\epsilon}$ in contradiction to $|\delta| \ll 1$. Therefore, δ and $\sqrt{\varepsilon}$ are of the same order and we assume the relation $\delta = 2\sqrt{\pi}\sqrt{\varepsilon} + b\varepsilon^c$. Substitution of this relation into equation (4.11) leads to $c = 3/2$ and $b = -\sqrt{\pi}/2$. Consequently, the relation between ε and δ is

$$\delta = 2\sqrt{\pi\varepsilon}(1 - \varepsilon). \tag{4.12}$$

Inserting this equation into equation (4.10) we get $\varepsilon = (1/3\sqrt{2})^{2/3}\alpha^{-2/3}$ and $\delta = 2\sqrt{\pi}(1/3\sqrt{2})^{1/3}\alpha^{-1/3}$. Substituting this ε into the asymptotic form of $E(R)$, we can confirm the asymptotic form of the generalization error [87] as

$$\epsilon_{\mathrm{g}} = \frac{\sqrt{2}}{\pi}\left(\frac{1}{3\sqrt{2}}\right)^{1/3}\alpha^{-1/3}. \tag{4.13}$$

Similar results were obtained in the two-layer $K = 2$ parity machine [80].

We next investigate the unrealizable case of finite a. We expand various quantities near the fixed point as

$$\begin{aligned} R(\varepsilon) &= R_0 + \varepsilon \\ l(\varepsilon) &= l_0 + l \\ E(\varepsilon) &= E_0 - E_1\varepsilon \\ F(\varepsilon) &= F_0 - F_1\varepsilon \\ G(\varepsilon) &= G_0 - G_1\varepsilon \end{aligned} \tag{4.14}$$

where

$$E_0 = 2\int_a^\infty \mathrm{D}v\, H\left(\frac{-R_0 v}{\sqrt{1-R_0^2}}\right) + 2\int_0^a \mathrm{D}v\, H\left(\frac{R_0 v}{\sqrt{1-R_0^2}}\right) \tag{4.15}$$

$$E_1 = \frac{R_0^2}{\pi\sqrt{1-R_0^2}}\left[1 - 2\exp\left(-\frac{a^2}{2(1-R_0^2)}\right)\right] \tag{4.16}$$

$$F_0 = -\frac{R_0}{\sqrt{2\pi}}(1 - 2\Delta) + \frac{1}{\sqrt{2\pi}} \tag{4.17}$$

$$F_1 = -\frac{1}{\sqrt{2\pi}}(1 - 2\Delta) \tag{4.18}$$

$$G_0 = -\frac{1}{\sqrt{2\pi}}(1 - 2\Delta) + \frac{R_0}{\sqrt{2\pi}} \tag{4.19}$$

$$G_1 = -\frac{1}{\sqrt{2\pi}}. \tag{4.20}$$

Then the differential equations (4.2) and (4.3) are linearized around the fixed point (R_0, l_0) as

$$\begin{aligned} \frac{dl}{d\alpha} &= A_{11}l + A_{12}\varepsilon \\ \frac{d\varepsilon}{d\alpha} &= A_{21}l + A_{22}\varepsilon \end{aligned} \tag{4.21}$$

where

$$A_{11} = -\frac{E_0}{2l_0^2} \tag{4.22}$$

$$A_{12} = \frac{1}{l_0}\left(F_1 l_0 - \frac{E_1}{2}\right) \tag{4.23}$$

$$A_{21} = \left[\frac{R_0 E_0}{l_0^3} - \frac{(F_0 R_0 - G_0)}{l_0^2}\right] \tag{4.24}$$

$$A_{22} = -\left[\frac{1}{2l_0^2}(E_0 - R_0 E_1) + \frac{1}{l_0}(F_1 R_0 - F_0 - G_1)\right]. \tag{4.25}$$

Let us recall that E_0, F_0, G_0 and l_0 should satisfy the following fixed point condition according to equations (4.2) and (4.3):

$$\frac{E_0}{2} - F_0 l_0 = 0 \tag{4.26}$$

$$-\frac{R_0}{2} E_0 + (F_0 R_0 - G_0) l_0 = 0 \tag{4.27}$$

These two conditions lead to $G_0 = 0$ and therefore from equation (4.19), we obtain the fixed point of R as

$$R_0 = (1 - 2\Delta). \tag{4.28}$$

Substituting this R_0 into $E(R)$, we get the minimum value of the generalization error $E_0 = \epsilon_{\min}(a)$. In Figures 2.3 and 2.2, we show R_0 and $E_0 = \epsilon_{\min}(a)$ as functions of a. Figure 2.3 indicates that the learning for $a = a_{c1} \equiv \sqrt{2\log 2}$, which is obtained from the condition $R_0 = 0$, is equivalent to a random guess, $\epsilon_{\min}(a_{c1}) = 0.5$.

Next we investigate the linearized differential equations in the vicinity of the fixed point (R_0, l_0) in the limit of large a. For large a we find

$$A_{11} \simeq -\frac{4}{\pi}\Delta^{3/2} + \mathcal{O}(\Delta^2) \tag{4.29}$$

$$A_{12} \simeq \left(\frac{1}{\sqrt{2\pi}} - \frac{1}{2\pi}\right) + \mathcal{O}(\Delta^{1/2}) \tag{4.30}$$

$$A_{21} \simeq 16\left(\frac{1}{\pi} - \frac{1}{\sqrt{2\pi}}\right)\Delta^2 + \mathcal{O}(\Delta) \tag{4.31}$$

$$A_{22} \simeq -\left(\frac{1}{2\pi} + \frac{2}{\sqrt{2\pi}}\right)\Delta^{1/2} + \mathcal{O}(\Delta). \tag{4.32}$$

Therefore, the eigenvalues of the matrix $\boldsymbol{A}$ are estimated as

$$\lambda = \frac{A_{11} + A_{22} \pm \sqrt{(A_{11} - A_{22})^2 + 4A_{12}A_{21}}}{2} \sim A_{11},\ A_{22}. \tag{4.33}$$

Since $|A_{11}| \ll |A_{22}|$, we conclude that for large a (or small Δ), the generalization error decays toward the minimum value

$$E(R) \simeq 2H(a) \simeq \frac{1}{\pi}\Gamma\left(\frac{1}{4}\right)\Delta^{3/4} \tag{4.34}$$

as

$$\frac{\sqrt{2\epsilon}}{\pi} \simeq \frac{\sqrt{2}}{\pi}\exp\left(-\frac{2}{\pi}\Delta^{2/3}\alpha\right). \tag{4.35}$$

In the limit $\Delta = 0$ (the realizable case), the eigenvalue vanishes and the generalization error becomes to decay to zero as $\sim\alpha^{-1/3}$.

4.2.2 Hebbian learning

In the Hebbian rule the dynamics of the student weight vector is

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + T_a(v)\,\boldsymbol{x}. \tag{4.36}$$

This recursion relation of the N-dimensional vector $\boldsymbol{J}$ is reduced to the evolution equations of the order parameters as

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{1}{2} + \frac{2R}{\sqrt{2\pi}}(1 - 2\Delta)l\right] \tag{4.37}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2} + \frac{2}{\sqrt{2\pi}}(1 - 2\Delta)(1 - R^2)l\right]. \tag{4.38}$$

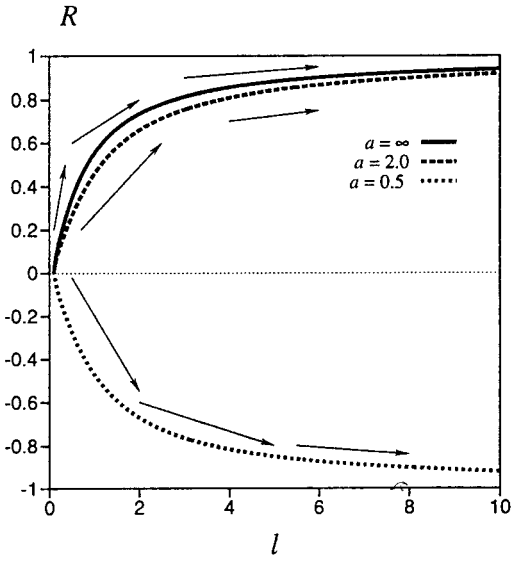

Figure 4.5: Flows of R and l for $a = \infty, 2.0$ and 0.5 by the Hebbian learning. For the cases of $a = \infty$ and 2.0, R reaches 1 and l goes to ∞. On the other hand, for $a = 0.5$, R reaches -1 as l goes to ∞.

Numerical analysis of differential equations

In Figures 4.5 and 4.6 , we plot the flows in the R-l plane and the generalization error for $a = \infty$, 2.0 and $a = 0.5$. Here we started the dynamics with the initial condition $(R_{\rm init}, l_{\rm init}) = (0.01, 0.1)$. Figure 4.5 shows that R reaches 1 for large a. On the other hand, R approaches -1 for small a. In order to find this bifurcation point near $R = 0$, we approximate equation (4.38) around $R \sim 0$ as

$$\frac{dR}{d\alpha} \simeq \frac{2}{\sqrt{2\pi}\, l}(1 - 2\Delta). \tag{4.39}$$

If the parameter a satisfies $a > a_{\rm c1} = \sqrt{2\log 2} = 1.18$, the derivative $dR/d\alpha$ is positive, and consequently R increases and eventually reaches 1 in the limit $\alpha \to \infty$. On the other hand, if $a < a_{\rm c1}$, R reaches -1 as $\alpha \to \infty$. Figure 4.6 shows how the generalization error behaves according to the width a of the reversed wedge. We learn from this figure that for $a = 0.5 (< a_{\rm c1})$, $\epsilon_{\rm g}$ has a minimum at some intermediate α. When the generalization error $\epsilon_{\rm g}$ passes through this value, $\epsilon_{\rm g}$ begins to increase toward the limiting value $\epsilon_{\rm min}(a) = 1 - 2H(a)$. Therefore, if the student learns excessively, he cannot achieve the lowest generalization error which is located at the global minimum of $E(R) = \epsilon_{\rm g}$ (over-training, see Figure 2.1) [88, 27]. This curious behavior may be understood as follows.

The Hebbian couplings have equal strengths for all the examples. However, for small a, the teacher changes his opinion frequently, and the examples presented some time ago destroy the most recent and important information for generalization. From Figure 2.1 we see that R must pass through a local minimum of $E(R)$ at $R = R_*$ in order to go to the state $R = -1$. If the parameter a satisfies $a < a_{\rm c2} = 0.80$, this local minimum is also the global minimum. Therefore, if $a < a_{\rm c1}$, although the generalization error decreases until R reaches R_*, it begins to increase as soon as R passes through the minimum point $R = R_*$ and finally reaches a larger value at $R = -1$. When the parameter a lies in the range $a_{\rm c2} < a < a_{\rm c1}$, the global minimum is located at $R = 1$. However, since R goes to -1 for $a < a_{\rm c1}$ (see equation (4.39)), the generalization error increases monotonically from 0.5 (random guess) to $1 - 2H(a) (> 0.5)$ for the parameter range $a_{\rm c2} < a < a_{\rm c1}$. We can regard this as a special case of over-training. From this fact, we can conclude that over-training appears for $a < a_{\rm c1}$.

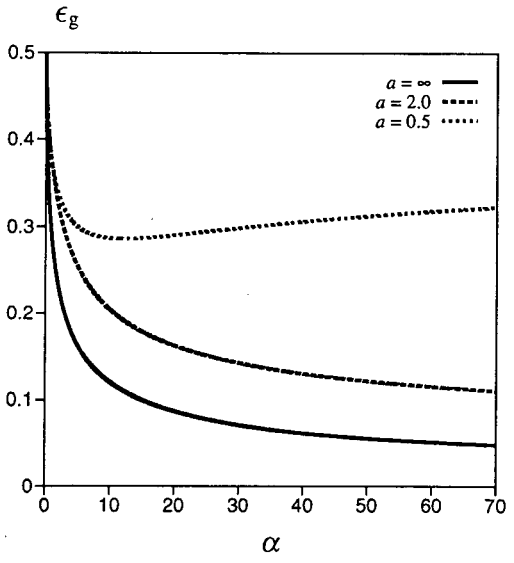

Figure 4.6: Generalization error ϵ_g for $a = \infty, 2.0$ and 0.5 by the Hebbian learning. For $a = \infty$ and 2.0, the generalization error converges to the optimal value $2H(a)$. However, in the case of $a = 0.5$, the generalization error begins to increase when the student learns too much (over-training).

Asymptotic analysis of the learning curve

For the realizable case $a = \infty$, we scale the differential equations as $R = 1 - \varepsilon$ and $l = 1/\delta$ and obtain

$$\frac{d\delta}{d\alpha} \simeq -\frac{2}{\sqrt{2\pi}}\delta^2 \tag{4.40}$$

$$\frac{d\varepsilon}{d\alpha} \simeq \frac{1}{2}\delta^2 - \frac{4}{\sqrt{2\pi}}\varepsilon\delta. \tag{4.41}$$

With the same technique as in the previous section, we obtain the asymptotic form of the generalization error in the limit $\alpha \to \infty$ as

$$\epsilon_g = \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\alpha}} \tag{4.42}$$

which is a well-known result [89].

We next investigate the unrealizable case of finite a . Simple manipulations as before show that for $a > a_{c1}$ the stable fixed point is at $R = 1$ and the differential equations (4.37) and (4.38) yield the generalization error

$$\epsilon_g = \frac{1}{\sqrt{2\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + 2H(a). \tag{4.43}$$

The limiting value $2H(a)$ is the best possible value. On the other hand, for $a < a_{c1}$, we should use the scaling relation $R = \varepsilon - 1$ and $l = 1/\delta$ because the order parameter R decreases monotonically toward -1. The corresponding generalization error turns out to be

$$\epsilon_g = \frac{1}{\sqrt{6\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + 1 - 2H(a) \tag{4.44}$$

for $a < a_{c1}$. The coefficients of $1/\sqrt{\alpha}$ in equations (4.43) and (4.44) diverge as $a \to a_{c1}$, signalling a crossover between the two types of asymptotic forms. The rate of approach to the asymptotic value, $1/\sqrt{\alpha}$, in equations (4.43) and (4.44) agrees with the corresponding behavior in the Gibbs learning of unrealizable rules [84].

4.2.3 AdaTron learning

In this subsection we investigate the generalization ability of student trained by the on-line AdaTron learning algorithm with examples generated by the above-mentioned non-monotonic rule. The AdaTron learning is a powerful method for realizable rules both in on-line and off-line modes in the sense that this algorithm gives a fast decay, proportional to α^{-1}, of the generalization error [98, 60, 83], in contrast to the $\alpha^{-1/3}$ and $\alpha^{-1/2}$ decays of the perceptron and Hebbian algorithms. We investigate the performance of the AdaTron learning algorithm in the unrealizable situation and discuss the asymptotic behavior of the generalization error.

The on-line training dynamics of the AdaTron algorithm is

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m - g(\alpha)\, u\, \Theta(-T_a(v)S(u))\, \boldsymbol{x}. \tag{4.45}$$

Using the same technique as the previous subsections, the evolutions of R and l are given by the following differential equations:

$$\frac{dl}{d\alpha} = \frac{g^2 E_{\rm Ad}}{2l} - gE_{\rm Ad} \tag{4.46}$$

$$\frac{dR}{d\alpha} = -\frac{Rg^2 E_{\rm Ad}}{2l^2} + \frac{gE_{\rm Ad}R - G_{\rm Ad}}{l}, \tag{4.47}$$

where

$$E_{\rm Ad} \equiv \ll u^2 \Theta(-T_a(v)S(u)) \gg = \sqrt{\frac{2}{\pi}} \int_0^\infty u^2 Du\, H_a(u,R) \tag{4.48}$$

with

$$H_a(u,R) \equiv H\left(\frac{a-Ru}{\sqrt{1-R^2}}\right) + H\left(\frac{Ru}{\sqrt{1-R^2}}\right) - H\left(\frac{a+Ru}{\sqrt{1-R^2}}\right) \tag{4.49}$$

and

$$\begin{aligned} G_{\rm Ad} &\equiv \ll uvT_a(v)\Theta(-T_a(v)S(u)) \gg \\ &= \frac{1}{\pi}(1-R^2)^{3/2}\left[2\exp\left(-\frac{a^2}{2(1-R^2)}\right) - 1\right] \\ &+ \sqrt{\frac{2}{\pi}} Ra(\sqrt{1-R^2})\Delta\left[1 - 2H\left(\frac{Ra}{\sqrt{1-R^2}}\right)\right] + RE_{\rm Ad}. \end{aligned} \tag{4.50}$$

Equations (4.46) and (4.47) determine the learning process.

Realizable case

We first consider the case of $g(\alpha) = 1$ and $a = \infty$, the realizable rule. We investigate the asymptotic behavior of the generalization error when R approaches 1, $R = 1 - \varepsilon$, $\varepsilon \to 0$ and $l = l_0$, a constant. From equations (4.48) and (5.36), we find $E_{\rm Ad} \sim c\varepsilon^{3/2}$ and $G_{\rm Ad} \sim (c - 2\sqrt{2}/\pi)\varepsilon^{3/2}$ with $c = 8/(3\sqrt{2}\pi)$. Then equation (4.47) is solved as $\varepsilon = (2/k)^2 \alpha^{-2}$ with

$$k \equiv \frac{2l_0 - 1}{2l_0^2} c + \frac{2\sqrt{2} - c\pi}{\pi l_0}. \tag{4.51}$$

Using this equation and $E(\varepsilon) = \epsilon_g \sim \sqrt{2\pi\varepsilon}/\pi$, we obtain the asymptotic form of the generalization error as

$$\epsilon_{\rm g} = E(R) \sim \frac{\sqrt{2\varepsilon}}{\pi} = \frac{2\sqrt{2}}{\pi k}\frac{1}{\alpha}. \tag{4.52}$$

The above expression of the generalization error depends on l_0, the asymptotic value of l, through k. Apparently l_0 is a function of the initial value of l as shown in Figure 4.7. A special case is $l_0 = 1/2$ in

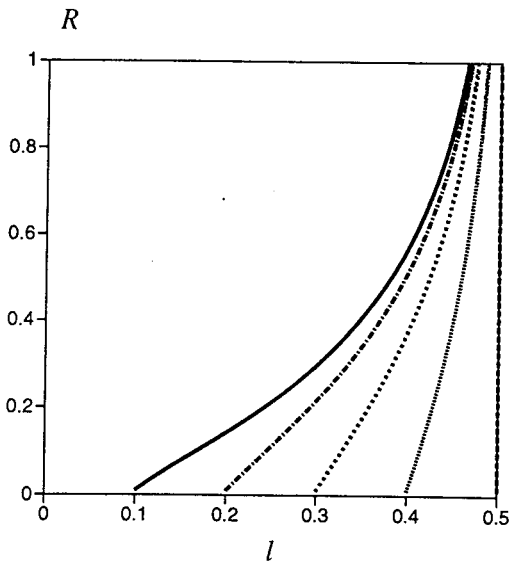

Figure 4.7: R-l trajectories of the AdaTron learning for the learnable case $a = \infty$. The fixed point depends on the initial value of $l = l_{\rm init}$. For the special case of $l_{\rm init} = 0.5$, the flow of l becomes independent of α.

which case l does not change as learning proceeds as is apparent from Eq. (4.46) as well as from Figure 4.7. Such a constant-l problem was studied by Biehl and Riegler [98] who concluded

$$\epsilon_{\rm g} = \frac{3}{2\alpha} \tag{4.53}$$

for the AdaTron algorithm. Our formula (4.52) reproduces this result when $l_0 = 1/2$. If one takes l_0 as an adjustable parameter, it is possible to minimize $\epsilon_{\rm g}$ by maximizing k in the denominator of equation (4.52). The smallest value of $\epsilon_{\rm g}$ is achieved when $l_0 = \pi c/2\sqrt{2}$, yielding

$$\epsilon_{\rm g} = \frac{4}{3\alpha} \tag{4.54}$$

which is smaller than equation (4.53) for a fixed l. We therefore have found that the asymptotic behavior of the generalization error depends upon whether or not the student weight vector is normalized and that a better result is obtained for the un-normalized case. We plot the generalization error for the present realizable case with the initial value of $l_{\rm init} = 0.1$ in Figure 4.8. We see that the Hebbian learning has the highest generalization ability and the AdaTron learning shows the slowest decay among the three algorithms in the initial stage of learning. However, as the number of presented patterns increases, the AdaTron algorithm eventually achieves the smallest value of the generalization error. In this sense the AdaTron learning algorithm is the most efficient learning strategy among the three in the case of the realizable rule.

Unrealizable case

For the unrealizable case, there can exist only one fixed point $l_0 = 1/2$. This reason is, for finite a, $E_{\rm Ad}$ appearing in equation (4.46) does not vanish in the limit of large α and $E_{\rm Ad}$ has a finite value for $a \neq \infty$. For this finite $E_{\rm Ad}$, the above differential equation has only one fixed point $l_0 = 1/2$. In contrast, for the realizable case, $E_{\rm Ad}$ behaves as $E_{\rm Ad} \sim c\,\varepsilon^{3/2}$ in the limit of $\alpha \to \infty$ and thus $dl/d\alpha$ becomes zero irrespective of l asymptotically. We plot trajectories in the R-l plane for $a = 2$ in Figure 4.9 and the corresponding generalization error is plotted in Figure 4.10 as an example. From Figure 4.9 we see that the destination of l is $1/2$ for all initial conditions. Figure 4.10 tells us that for the unrealizable case $a = 2$, the AdaTron learning has the lowest generalization ability among the three. We should notice

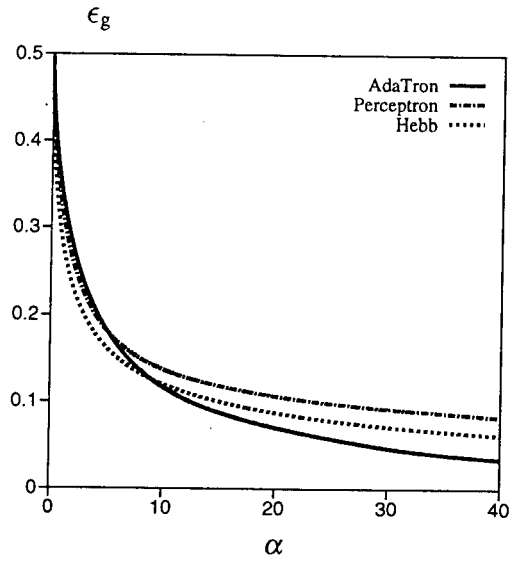

Figure 4.8: Generalization errors of the AdaTron, perceptron and Hebbian learning algorithms for the learnable case $a = \infty$. The initial value of l is $l_{\text{init}} = 0.1$ for all algorithms. The AdaTron learning shows the fastest convergence among the three.

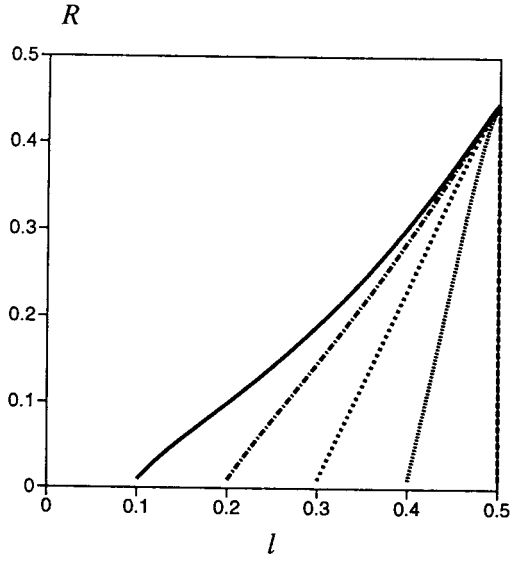

Figure 4.9: R-l trajectories of the AdaTron learning for the unlearnable case $a = 2$. All flows of l converge to the fixed point at $l_0 = 1/2$.

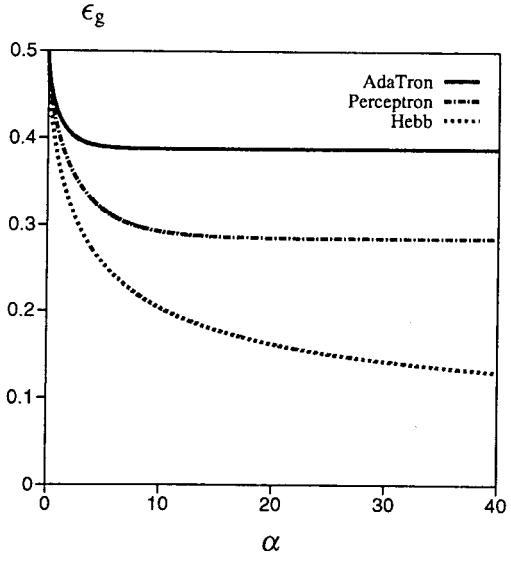

Figure 4.10: Generalization errors of the AdaTron, perceptron and Hebbian learning algorithms for the unlearnable case $a = 2$. The AdaTron learning shows the largest residual error among the three.

that the generalization error decays to its asymptotic value, the residual error $\epsilon_{\min}$, as $\epsilon_g - \epsilon_{\min} \sim \alpha^{-1/2}$ for the Hebbian learning and decays exponentially for perceptron learning [37]. The residual error of the Hebbian learning $\epsilon_{\min} = 2H(a)$ is also the best possible value of the generalization error for $a > a_{c2}$ as seen in Figure 4.11. In Figure 4.12 we also plot the generalization error of the AdaTron algorithm for several values of a. For the AdaTron learning of the unrealizable case, the generalization error converges to a non-optimal value $E(R_0)$ exponentially.

For all unrealizable cases, the R-l flow is attracted into the fixed point $(R_0, 1/2)$, where R_0 is obtained from

$$\left.\frac{dR}{d\alpha}\right|_{l=\frac{1}{2},\, R=R_0} = -2G_{\rm Ad}(R_0) = 0. \tag{4.55}$$

The solution R_0 of the above equation is not the optimal value because, as we mentioned in Chapter 2, the optimal value of the present learning system is $R_{\rm opt} = 1$ for $a > a_{c2}$ and $R_{\rm opt} = R_* = -\sqrt{(2\log 2 - a^2)/2\log 2}$ for $a < a_{c2}$ [90].

From Figures 4.11 and 4.10, we see that the residual error $\epsilon_{\min}$ of the AdaTron learning is larger than that of the conventional perceptron learning. Therefore, we conclude that if the student learns from the unrealizable rules, the on-line AdaTron algorithm becomes the worst strategy among three learning algorithms as we discussed above although for the realizable case, the on-line AdaTron learning is a sophisticated algorithm and the generalization error decays to zero as quickly as the off-line learning [91].

4.3 Learning under output noise in the teacher signal

We now investigate the problem of learning in the presence of output noise. The output of the teacher is inverted randomly with a rate $\lambda\,(\leq 1/2)$ for each example. This problem has been investigated for a simple perceptron by the non-normalized [82] and normalized [81] perceptron algorithms, and for the $K = 2$ parity machine by the least action algorithm [80]. Here we show that the width a of the reversed wedge plays essentially the same role as output noise in the teacher signal. In this section, we take up the perceptron learning and Hebbian learning algorithms.

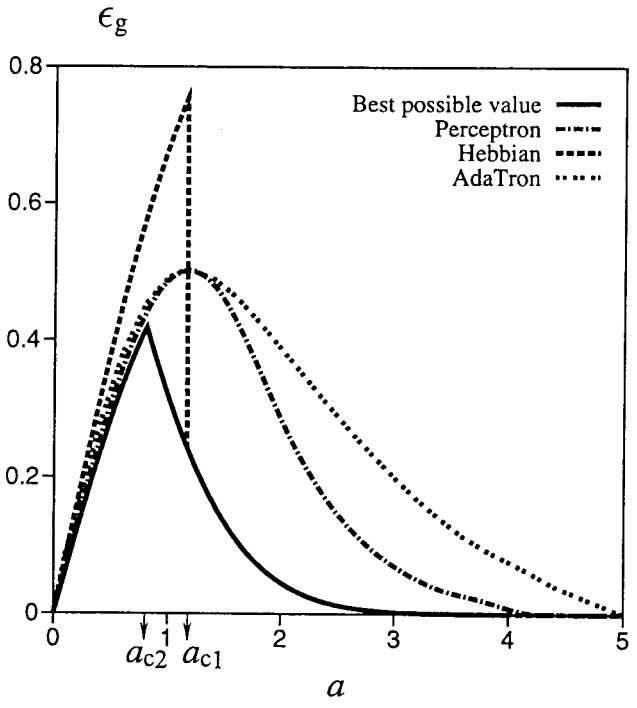

Figure 4.11: Best possible value of the generalization error and the residual error for Hebbian, perceptron and AdaTron learning algorithms.

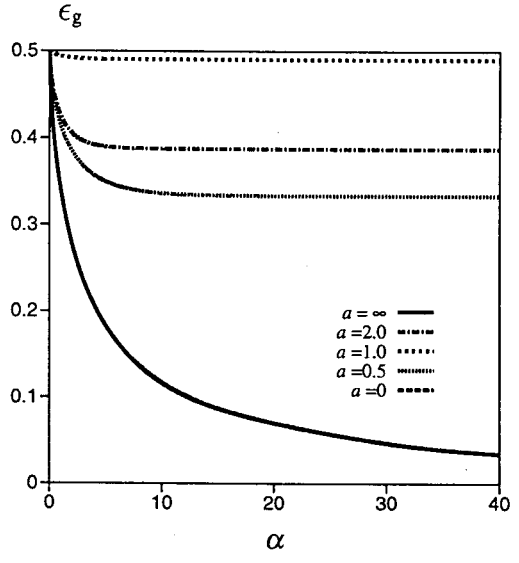

Figure 4.12: Generalization errors of the AdaTron learning algorithm for the cases of $a = \infty$, 2, 1 and 0.5.

4.3.1 Perceptron learning

According to references [80, 81, 82], the effect of output noise is taken into account in the differential equations (4.2) and (4.3) by replacing $E(R)$, $F(R)$ and $G(R)$ with $\tilde{E}_\lambda(R)$, $\tilde{F}_\lambda(R)$ and $\tilde{G}_\lambda(R)$ as follows

$$\tilde{E}_\lambda(R) = (1-\lambda)E(R) + \lambda E^c(R) \tag{4.56}$$
$$\tilde{F}_\lambda(R) = (1-\lambda)F(R) + \lambda F^c(R) \tag{4.57}$$
$$\tilde{G}_\lambda(R) = (1-\lambda)G(R) + \lambda G^c(R) \tag{4.58}$$

where $E^c(R) = \ll 1 \gg_R^c$, $F_R^c = \ll \mathrm{sign}(u)u \gg_R^c$ and $G_R^c = \ll \mathrm{sign}(u)v \gg_R^c$ with the average defined as

$$\ll \cdots \gg_R^c = \int\int_{\Omega_c} (\cdots) du dv P_R(u,v). \tag{4.59}$$

Here Ω_c is the region in the u-v plane satisfying $T_a(v) = S(u)$. The derivation of equations (4.57),(4.58) and (4.58) is understood as follows. Although, we express the average $\ll \cdots \gg_R$ as equation (4.4), we can rewrite this expression using $\Theta(-T_a(v)S(u))$ as follows.

$$E(R) = \ll \Theta(-T_a(v)S(u)) \gg \tag{4.60}$$
$$F(R) = \ll \Theta(-T_a(v)S(u))\mathrm{sign}(u)u \gg \tag{4.61}$$
$$G(R) = \ll \Theta(-T_a(v)S(u))\mathrm{sign}(u)v \gg \tag{4.62}$$
$$\tag{4.63}$$

Therefore, if the teacher output $T_a(v)$ is inverted with the rate λ, The factor $\Theta(-T_a(v)S(u))$ appearing in the equations (4.61), (4.62) and (4.63) leads to the next expression on average.

$$\Theta(-T_a(v)S(u)) \to (1-\lambda)\Theta(-T_a(v)S(u)) + \lambda\Theta(T_a(v)S(u)) \tag{4.64}$$

Therefore, using this relation, for example, $E(R)$ under the output noise of the teacher can be written as follows.

$$\begin{aligned} \tilde{E}^c(R) &= \lambda \ll \Theta(-T_a(v)S(u)) \gg + (1-\lambda) \ll \Theta(-T_a(v)S(u)) \gg \\ &= (1-\lambda)E(R) + \lambda E^c(R). \end{aligned} \tag{4.65}$$

After simple calculations, we obtain

$$E^c(R) = \int_a^\infty Dv\, H\left(\frac{Rv}{\sqrt{1-R^2}}\right) + \int_0^a Dv\, H\left(\frac{-Rv}{\sqrt{1-R^2}}\right) \tag{4.66}$$
$$F^c(R) = \frac{R}{\sqrt{2\pi}}(1-2\Delta) + \frac{1}{\sqrt{2\pi}} \tag{4.67}$$
$$G^c(R) = \frac{1}{\sqrt{2\pi}}(1-2\Delta) + \frac{R}{\sqrt{2\pi}}. \tag{4.68}$$

Numerical analysis of differential equations

In Figures 4.13 and 4.14 we plot the flows of R and l for the realizable case with noise revel $\lambda = 0.01$ and 0.20 respectively. We also plot the flows of R and l for unrealizable case $a = 2.0$ with same noise revel as the realizable case in Figures 4.15 and 4.16. Naturally the generalization ability deteriorates as the noise level λ approaches 1/2. Even when $a = \infty$ and $\alpha \to \infty$, a λ-dependent fixed point appears and the perfect generalization is impossible. In Figures 4.17 ($a = \infty$) and Figure 4.18 (2.0), we plot generalization errors corresponding to each flow.

Asymptotic analysis of learning curve

We study in this subsection the asymptotic behavior of the learning curve in the limit of small noise level $\lambda \ll 1$. For the realizable case $a = \infty$, we define ε by the relation $R = 1 - \varepsilon$ to get the differential equations

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{\sqrt{2}}{2\pi}\varepsilon^{1/2} - \left(\frac{2\lambda}{\sqrt{2\pi}} + \frac{\varepsilon}{\sqrt{2\pi}}\right)l\right] \tag{4.69}$$

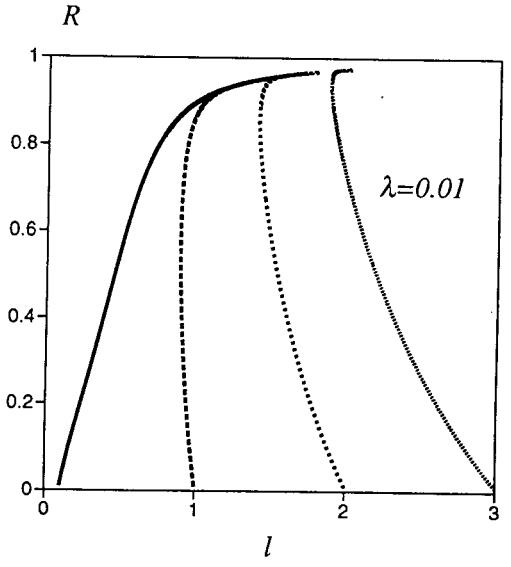

Figure 4.13: Flow of R and l for the learnable case $a = \infty$ with noise level $\lambda = 0.01$. Due to the presence of noise, the flow of R is attracted to a fixed point $R_0 = (1 - 2\lambda)$. Then the generalization stops.

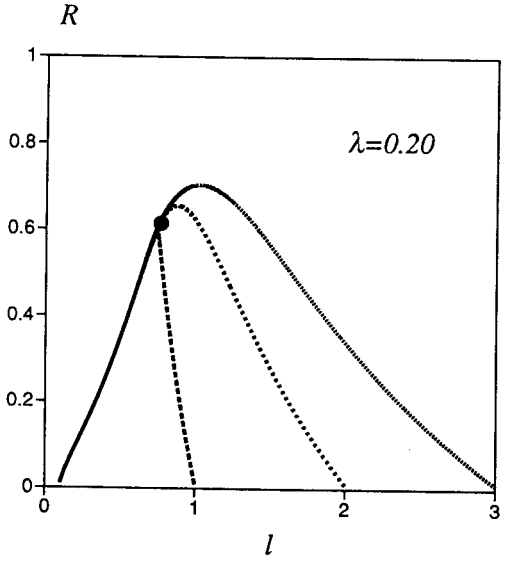

Figure 4.14: Flow of R and l for the learnable case $a = \infty$ with noise level $\lambda = 0.20$. Due to the presence of noise, the flow of R is attracted to a fixed point $R_0 = (1 - 2\lambda)$. Then the generalization stops.

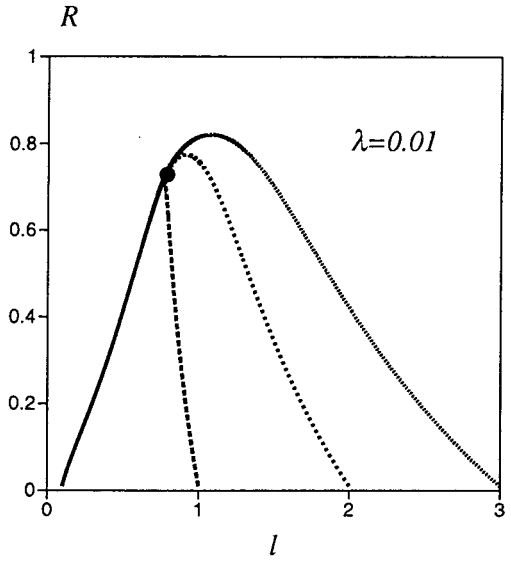

Figure 4.15: Flow of R and l for the unrealizable case $a = 2$ with noise level $\lambda = 0.01$. Due to the presence of noise, the flow of R is attracted to a fixed point $R_0 = (1 - 2\lambda)$. Then the generalization stops.

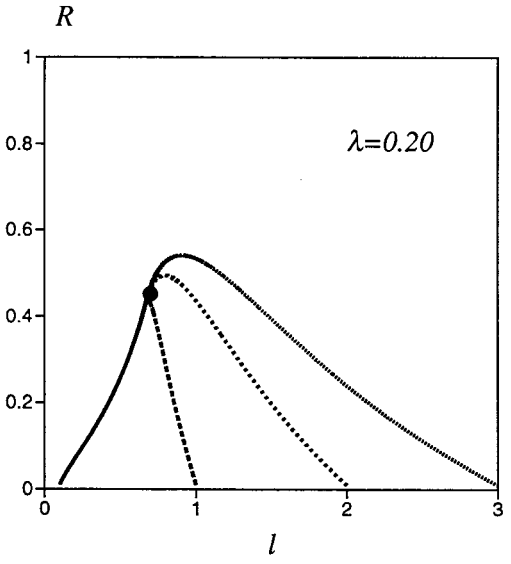

Figure 4.16: Flow of R and l for the unrealizable case $a = 2$ with noise level $\lambda = 0.20$. Due to the presence of noise, the flow of R is attracted to a fixed point $R_0 = (1 - 2\lambda)$. Then the generalization stops.

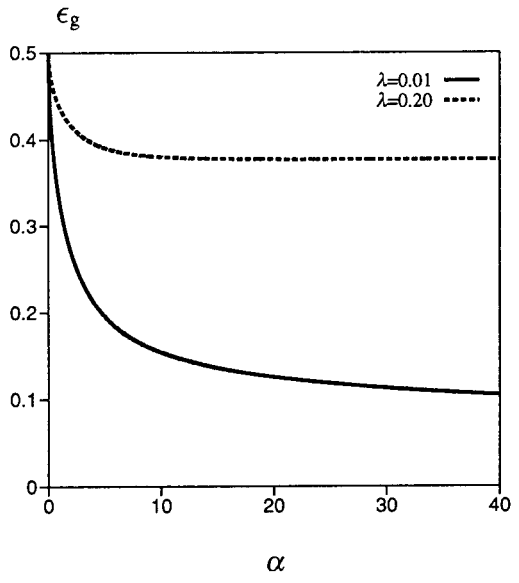

Figure 4.17: Generalization error for the learnable case $a = \infty$ with output noise $\lambda = 0.01$ and $\lambda = 0.2$.

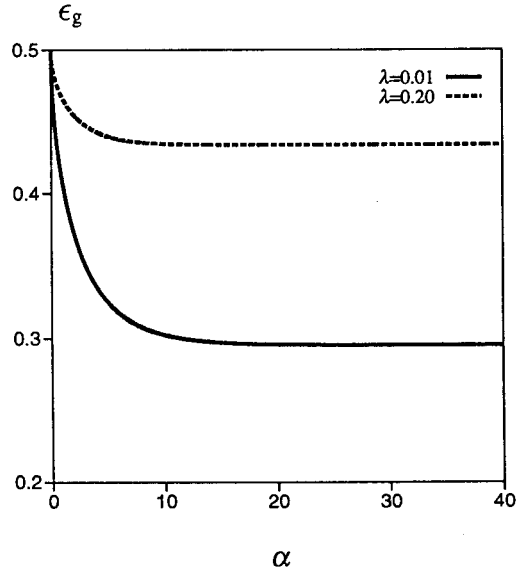

Figure 4.18: Generalization error for the learnable case $a = 2.0$ with output noise $\lambda = 0.01$ and $\lambda = 0.2$.

$$\frac{d\varepsilon}{d\alpha} = \frac{1}{l^2}\left[\frac{\sqrt{2}}{2\pi}\varepsilon^{1/2} - \frac{2\varepsilon}{\sqrt{2\pi}}l\right] \tag{4.70}$$

where only the leading terms with respect to λ have been left. The fixed point (ε_0, l_0) is given as

$$(\varepsilon_0, l_0) = \left(2\lambda, \frac{1}{2\sqrt{2\pi}}\lambda^{-1/2}\right). \tag{4.71}$$

Let us investigate how fast this fixed point is approached. We set

$$\begin{aligned} \varepsilon &= \varepsilon_0(1-\varepsilon) \\ l &= l_0(1-l) \end{aligned} \tag{4.72}$$

and linearize the differential equations around (ε_0, l_0) to obtain

$$\frac{dl}{d\alpha} = -8\lambda^{3/2}l \tag{4.73}$$

$$\frac{d\varepsilon}{d\alpha} = -2\lambda^{1/2}\varepsilon - 8\lambda^{1/2}l. \tag{4.74}$$

From the eigenvalues of the coefficient matrix, we obtain the asymptotic solution

$$\begin{aligned} l &\sim l_0\left[1 + \mathcal{O}(\mathrm{e}^{-8\lambda^{3/2}\alpha})\right] \\ \varepsilon_0 &\sim \varepsilon_0\left[1 + \mathcal{O}(\mathrm{e}^{-8\lambda^{3/2}\alpha})\right]. \end{aligned} \tag{4.75}$$

Therefore, the generalization error ϵ_g converges to a finite value exponentially, $\exp(-8\lambda^{3/2}\alpha)$. The limiting value of the generalization error is

$$E(R = 1 - 2\lambda) = \frac{2}{\pi}\lambda^{1/2}. \tag{4.76}$$

It is important to bear in mind that the above results hold only for small noise level ($\lambda \ll 1$).

According to Biehl *et al* [82], it is useful to distinguish two performance measures of on-line learning, the generalization error ϵ_g and the prediction error ϵ_p. The generalization error ϵ_g is the probability of disagreement between the student and the genuine rule of the teacher as we have discussed. On the other hand, the prediction error ϵ_p is the probability for disagreement between the student and the noisy teacher output for an arbitrary input. In the present case, the prediction error ϵ_p and generalization error ϵ_g satisfy the relation

$$\epsilon_p = \lambda + (1 - 2\lambda)\epsilon_g. \tag{4.77}$$

We next investigate the unrealizable case of large but finite a for small noise level. We can derive the linearized differential equations around the fixed point (R_0, l_0) by replacing E_0, E_1 and so on in subsection 4.2.1 with $\tilde{E}_0(\lambda)$, $\tilde{E}_1(\lambda)$ etc using the same relation as equation (4.14)

$$\begin{aligned} \tilde{E}(\lambda) &= \tilde{E}_0(\lambda) - \tilde{E}_1(\lambda)\varepsilon \\ \tilde{F}(\lambda) &= \tilde{F}_0(\lambda) - \tilde{F}_1(\lambda)\varepsilon \\ \tilde{G}(\lambda) &= \tilde{G}_0(\lambda) - \tilde{G}_1(\lambda)\varepsilon \\ R &= R_0(\lambda) + \varepsilon \\ l &= l_0(\lambda) + l. \end{aligned} \tag{4.78}$$

Then

$$\tilde{E}_0(\lambda) \simeq \lambda + (1-2\lambda)I(a,\lambda) - \frac{1-2\lambda}{4\pi(\lambda-\lambda^2)}\Delta + \mathcal{O}\left(\Delta^{1+1/4(\lambda-\lambda^2)}\right) \tag{4.79}$$

$$\tilde{E}_1(\lambda) \simeq \frac{1-2\lambda}{2\pi\sqrt{\lambda-\lambda^2}}\left(1 - \frac{1-2\lambda}{2(\lambda-\lambda^2)}\Delta\right) + \mathcal{O}\left(\Delta^{1/4(\lambda-\lambda^2)}\right) \tag{4.80}$$

$$\tilde{F}_0(\lambda) \simeq \frac{4}{\sqrt{2\pi}}(\lambda-\lambda^2)+\frac{4}{\sqrt{2\pi}}(1-2\lambda)^2+\mathcal{O}(\Delta^2) \tag{4.81}$$

$$\tilde{F}_1(\lambda) = \frac{1}{\sqrt{2\pi}}(1-2\lambda)-\frac{2(1-2\lambda)}{\sqrt{2\pi}} \tag{4.82}$$

$$\tilde{G}_0(\lambda) = 0 \tag{4.83}$$

$$\tilde{G}_1(\lambda) = -\frac{1}{\sqrt{2\pi}} \tag{4.84}$$

and

$$R_0(\lambda) = (1-2\Delta)(1-2\lambda) \tag{4.85}$$

$$l_0 \simeq L_0(\lambda)+L_1(\lambda)\Delta+\mathcal{O}(\Delta^2) \tag{4.86}$$

where

$$L_0(\lambda)=\frac{\sqrt{2\pi}[\lambda+(1-2\lambda)I(a,\lambda)]}{8(\lambda-\lambda^2)} \tag{4.87}$$

and

$$L_1(\lambda)=\frac{(1-2\lambda)+4\pi(1-\lambda)^2[\lambda+(1-2\lambda)I(a,\lambda)]}{4\pi(\lambda-\lambda^2)} \tag{4.88}$$

with

$$I(a,\lambda)\equiv 2\int_a^\infty \mathrm{D}v\, H\left(-\frac{(1-2\lambda)}{2\sqrt{\lambda-\lambda^2}}v\right)+2\int_0^a \mathrm{D}v\, H\left(\frac{(1-2\lambda)}{2\sqrt{\lambda-\lambda^2}}v\right). \tag{4.89}$$

It is worth noting that as $\Delta(\equiv e^{-a^2/2})$ and λ appear in R_0 in the same form, and therefore the width a of the reversed wedge and the output noise level have the same effect on the generalization ability.

Substituting $\tilde{E}_0$, $\tilde{E}_1$ etc into the differential equations (4.2) and (4.3) for small λ ($\leq 1/2$) and keeping only the leading order terms with respect to λ and Δ, we obtain

$$\frac{dl}{d\alpha} = \left(-8\lambda^{3/2}+8\lambda^{1/2}\Delta\right)l+\frac{\lambda^{-1}}{\sqrt{2\pi}}\Delta\varepsilon \tag{4.90}$$

$$\frac{d\varepsilon}{d\alpha} = \left(16\sqrt{2\pi}\lambda^2-4\sqrt{2\pi}\lambda\Delta\right)l+\left(-2\lambda^{1/2}-\lambda^{-1/2}\Delta\right)\varepsilon \tag{4.91}$$

where $I(a,\lambda)$ appearing in $\tilde{E}_0(\lambda)$ was approximated in the limit of $\lambda\to 0$ by $(2/\pi)\lambda^{1/2}+2H(a)$. In order to compare the realizable and unrealizable cases, it is useful to re-scale the variables

$$\begin{aligned}\varepsilon &\to \varepsilon_0\varepsilon\\ l &\to -l_0 l\end{aligned} \tag{4.92}$$

in equations (4.90) and (4.91). Using this transformation, we rewrite equations (4.90) and (4.91) as

$$\frac{dl}{d\alpha} = \left(-8\lambda^{3/2}+8\lambda^{1/2}\Delta\right)l+2\lambda^{-1/2}\Delta\varepsilon \tag{4.93}$$

$$\frac{d\varepsilon}{d\alpha} = \left(-4\lambda^{1/2}+2\lambda^{-1/2}\Delta\right)l+\left(-2\lambda^{-1/2}-\lambda^{1/2}\Delta\right)\varepsilon. \tag{4.94}$$

The eigenvalues of the matrix composed of coefficients on the right-hand side of the above linearized differential equations are

$$t_\pm=\frac{(-8\lambda^{3/2}-2\lambda^{1/2}\pm\sqrt{(-8\lambda^{3/2}+2\lambda^{1/2})^2-(8\Delta+4\lambda^{-1}\Delta^2)}}{2}. \tag{4.95}$$

In the limit $\Delta\to 0$ and $\lambda\to 0$, we may keep only t_-. Then the generalization error converges to $(2/\pi)\lambda^{1/2}+2H(a)$ exponentially as $\exp(-t_-\alpha)$ for large a and small λ. The prediction error is given by $\epsilon_{\rm p}=\lambda+(1-2\lambda)\epsilon_{\rm g}$.

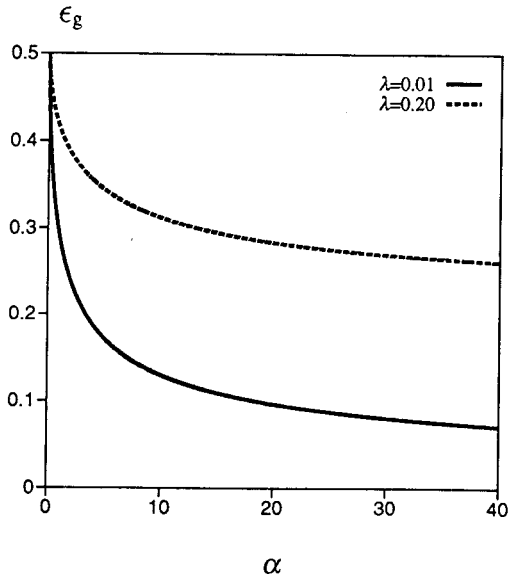

Figure 4.19: Generalization error for the learnable case $a = \infty$ with output noise $\lambda = 0.01$ and 0.20 by the Hebbian learning.

4.3.2 Hebbian learning

Using the same technique as in the perceptron learning problem, we obtain the next differential equations of the order parameters

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{1}{2} + \frac{2R}{\sqrt{2\pi}}(1-2\Delta)(1-2\lambda)l\right] \tag{4.96}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2} + \frac{2}{\sqrt{2\pi}}(1-2\Delta)(1-2\lambda)(1-R^2)l\right]. \tag{4.97}$$

Numerical analysis of differential equation

We plot the generalization error for $a = \infty, 2.0$ and $a = 0.5$ in Figures 4.19, 4.20 and 4.21, respectively, by solving these differential equations numerically. We saw in the previous section that the over-training appears if $a < a_{c1} = \sqrt{2\log 2}$. The student cannot achieve the minimum error $\epsilon_{\min}$ if he learns too much. Figure 4.21 indicates the existence of the number of learning steps α at which $d\epsilon_g/d\alpha = 0$ holds. Beyond this point, over-training appears, and this critical value of α increases as the noise level λ increases. This result may be understood as follows.

For small λ (e.g. $\lambda = 0.01$), ϵ_g once reaches a minimum as a function of α and then begins to increase (over-training). This means from Figure 2.1 that R passes through R_* corresponding to the minimum of $E(R)$. For larger λ (e.g. $\lambda = 0.20$), however, there appears no minimum in ϵ_g as α increases. This implies in terms of Figure 2.1 that R becomes stuck at an intermediate R before it reaches R_*.

Asymptotic analysis of the learning curve

The asymptotic form for the noisy case can be derived simply by replacing $(1-2\Delta)$ in the asymptotic form of the noiseless case with $(1-2\Delta)(1-2\lambda)$. Accordingly, we get the generalization error for $a > a_{c1} = \sqrt{2\log 2}$ as

$$\epsilon_g = \frac{1}{\sqrt{2\pi}(1-2\Delta)(1-2\lambda)}\frac{1}{\sqrt{\alpha}} + 2H(a) \tag{4.98}$$

and for $a < a_{c1}$ as

$$\epsilon_g = \frac{1}{\sqrt{6\pi}(1-2\Delta)(1-2\lambda)}\frac{1}{\sqrt{\alpha}} + 1 - 2H(a). \tag{4.99}$$

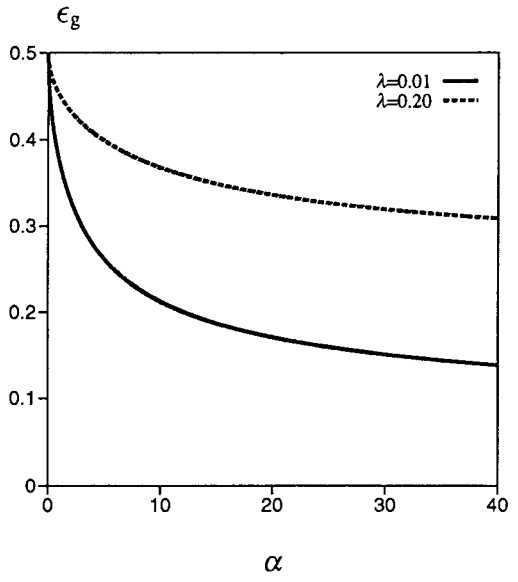

Figure 4.20: Generalization error for the learnable case $a = 2.0$ with output noise $\lambda = 0.01$ and 0.20 by the Hebbian learning.

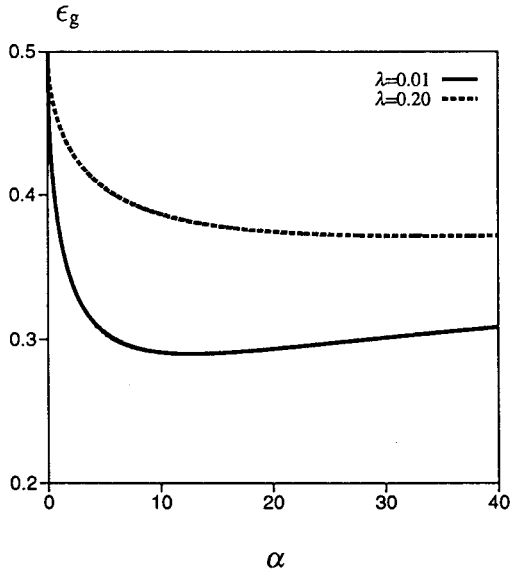

Figure 4.21: Generalization error for the learnable case $a = 0.5$ with output noise $\lambda = 0.01$ and 0.20 by the Hebbian learning.

As we saw in the previous subsection, the prediction error obeys the relation

$$\epsilon_{\rm p} = \lambda + (1-2\lambda)\epsilon_{\rm g}. \tag{4.100}$$

Therefore we get the prediction error for $a > a_{\rm c1}$ as

$$\epsilon_{\rm p} = \frac{1}{\sqrt{2\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + 2(1-2\lambda)H(a) + \lambda \tag{4.101}$$

and for $a < a_{\rm c1}$ as

$$\epsilon_{\rm p} = \frac{1}{\sqrt{6\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + (1-2\lambda)(1-2H(a)) + \lambda. \tag{4.102}$$

From equations (4.98) and (4.99), we see that the width a of the reversed wedge expressed in terms of $\Delta = {\rm e}^{-a^2/2}$ and the output noise λ have the same effect on the asymptotic generalization ability. In the non-monotonic Hopfield model [32, 33, 34, 35, 36, 37] which works as an associative memory, the parameter Δ also plays as the output noise. If we embed patterns by the Hebb rule in the network, the capacity of the network drastically deteriorates for small a. From equations (4.98) and (4.99), we see that the coefficients of $1/\sqrt{\alpha}$ in the generalization error diverges when $\Delta = \lambda = 1/2$

4.4 Optimization of learning rate

We have so far investigated the learning processes with a fixed learning rate. In this section we consider optimization of the learning rate to improve the learning performance. It turns out that the perceptron learning with optimized learning rate achieves the best possible generalization error in the range $a \geq a_{\rm c1}$.

We first introduce the learning rate $g(\alpha)$ in our dynamics. As an example, the learning dynamics for the perceptron algorithm is written using this parameter as

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m - g(\alpha)\,\Theta(-T_a(v)S(u))\,{\rm sign}(u)\,\boldsymbol{x}. \tag{4.103}$$

This optimization procedure is different from the technique of Kinouchi and Caticha [97]. They investigated the on-line dynamics with a general weight function $f(T_a(v), u)$ as

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + f(T_a(v), u)\,T_a(v)\,\boldsymbol{x} \tag{4.104}$$

and chose $f(T_a, u)$ so that it maximizes the increase of R per learning step. [1]

In contrast, our optimization procedure adjusts the parameter $g(\alpha)$ keeping the learning algorithm unchanged. This procedure is somewhat similar to simulated annealing processes with the annealing schedule corresponding to the choice of $g(\alpha)$ [46, 93, 94].

4.4.1 Perceptron learning

Trajectory in the R-l plane

In this section, we show that the trajectories in the R-l plane can be derived explicitly for the optimal learning rate $g_{\rm opt}(\alpha)$. Using the same techniques as in the previous two sections, we can readily obtain the differential equations with the learning rate $g(\alpha)$ as

$$\frac{dl}{d\alpha} = \frac{g(\alpha)^2 E(R)/2 - g(\alpha)F(R)l}{l} \tag{4.105}$$

$$\frac{dR}{d\alpha} = \frac{-RE(R)g(\alpha)^2/2 + g(\alpha)\left[F(R)R - G(R)\right]l}{l^2} \equiv L(g(\alpha)). \tag{4.106}$$

[1] We use this technique for optimizing the function f in the next chapter for the case in which a non-monotonic perceptron as a student learns from a non-monotonic perceptron as a teacher (realizable task).

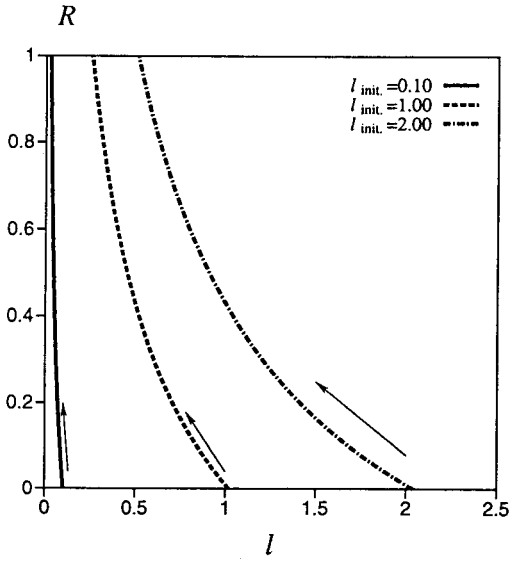

Figure 4.22: The trajectories in the R-l plane with the optimal learning rate by the perceptron learning $a = \infty$ We choose the initial condition as $(R_{\rm init}, l_{\rm init}) = (0.01, 0.10), (0.01, 1.00)$ and $(0.01, 2.00)$. In all cases, the student goes to the state of $R = 1$ after infinite learning steps under any initial condition. As R reaches 1, l decreases.

Now we choose the parameter g to maximize $L(g(\alpha))$ with the aim to accelerate the increase of R

$$g_{\rm opt}(\alpha) = \frac{[F(R)\,R - G(R)]\,l}{RE(R)}. \tag{4.107}$$

Substituting this g into equations (4.105) and (4.106) we obtain

$$\frac{dl}{d\alpha} = -\frac{[F(R)\,R - G(R)]\,[F(R)\,R + G(R)]\,l}{2R^2E(R)} \tag{4.108}$$

$$\frac{dR}{d\alpha} = \frac{[F(R)\,R - G(R)]^2}{2RE(R)} \tag{4.109}$$

It is possible to eliminate α from these equations by taking their ratio

$$\frac{dR}{dl} = -\frac{[F(R)\,R - G(R)]\,R}{[F(R)\,R + G(R)]\,l}. \tag{4.110}$$

Using equations (4.5) and (4.6) we obtain the trajectory in the R-l plane as

$$(1+R)^{-(1+A)/A}(1-R)^{(1-A)/A}R = cl \tag{4.111}$$

where $A = 1 - 2\Delta$ and c is a constant.

In Figures 4.22, 4.23 and 4.24, we plot the above trajectory for $a = \infty$, 2.0 and 0.5, respectively, by adjusting c to reproduce the initial conditions $(R_{\rm init}, l_{\rm init}) = (0.01, 0.10), (0.01, 1.00)$ and $(0.01, 2.00)$. These figures indicate that the student goes to the state of $R = 1$ after infinite learning steps ($\alpha \to \infty$) for any initial condition. However, we may notice that the final value of l strongly depends on the parameter a. If a is small (e.g., 0.5), l increases indefinitely as $\alpha \to \infty$. On the other hand, for larger a, l is seen to decrease as α goes to ∞. We investigate this a-dependence of l in more detail in the next subsection.

We plot the corresponding generalization error in Figures 4.25, 4.26 and 4.27. From these figures, we see that for $a = \infty$ and 2.0, the generalization ability is improved significantly. However, for $a = 0.5$, the generalization ability becomes worse than that for $g = 1$ (the unoptimized case). This may be explained

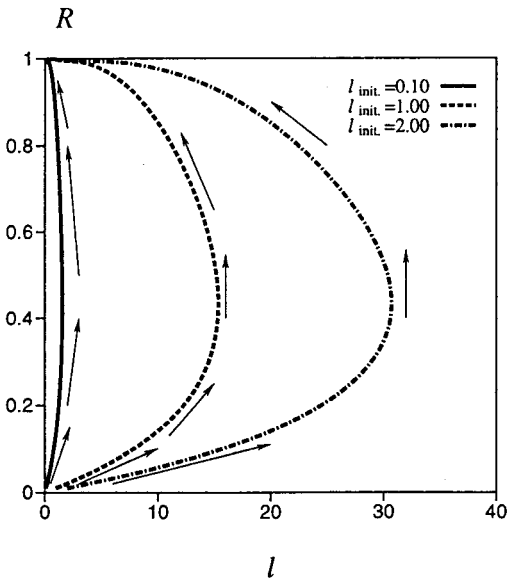

Figure 4.23: The trajectories in the R-l plane with the optimal learning rate by the perceptron learning $a = 2.0$ We choose the initial condition as $(R_{\text{init}}, l_{\text{init}}) = (0.01, 0.10), (0.01, 1.00)$ and $(0.01, 2.00)$. In all cases, the student goes to the state of $R = 1$ after infinite learning steps under any initial condition. As R reaches 1, l decreases.

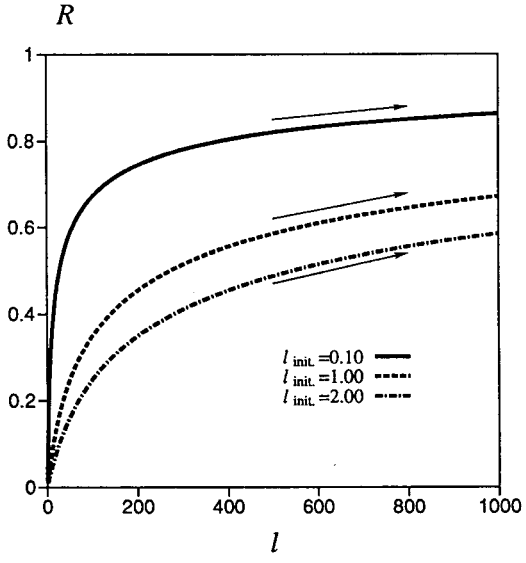

Figure 4.24: The trajectories in the R-l plane with the optimal learning rate by the perceptron learning $a = 0.5$ We choose the initial condition as $(R_{\text{init}}, l_{\text{init}}) = (0.01, 0.10), (0.01, 1.00)$ and $(0.01, 2.00)$. In all cases, the student goes to the state of $R = 1$ after infinite learning steps under any initial condition. As R reaches 1, l increases.

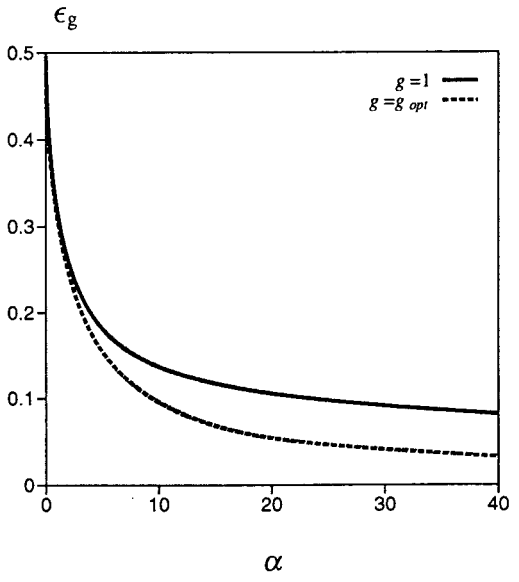

Figure 4.25: Generalization error for with the optimal learning rate $g_{\rm opt}$. For $a = \infty$, $\epsilon_{\rm g}$ goes to the optimal value.

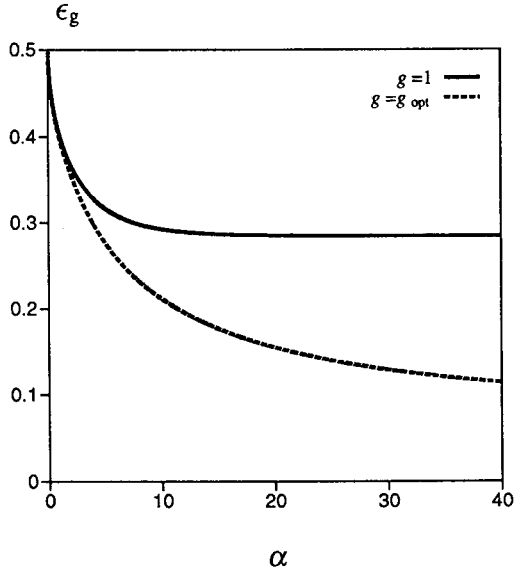

Figure 4.26: Generalization error for with the optimal learning rate $g_{\rm opt}$. For $a = 2$, $\epsilon_{\rm g}$ goes to the optimal value.

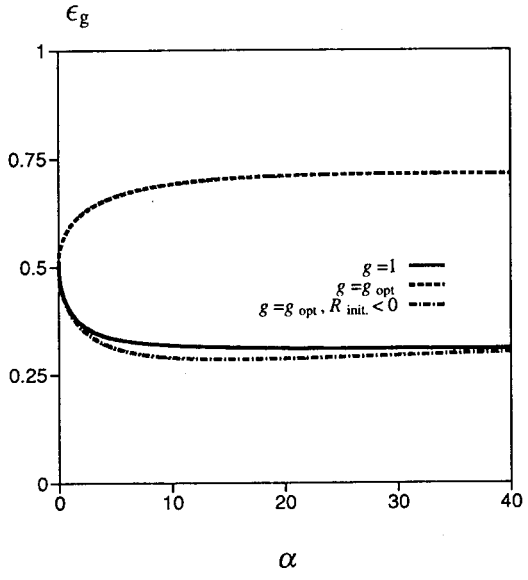

Figure 4.27: Generalization error for with the optimal learning rate $g_{\rm opt}$. For $a = 0.5$, $\epsilon_{\rm g}$ goes to a worse value than that of $g = 1$. If we select a negative value as the initial condition of R for $a = 0.5$, the generalization error converges to $1 - 2H(a)(> 0.5)$.

as follows. In Figure 4.11, we see that the optimal overlap $R_{\rm opt}$, which gives the optimal generalization error $\epsilon_{\rm g}^{\rm opt}$ in the limit $\alpha\to\infty$, is 1 for $a > a_{\rm c2} = 0.80$. However, for $a < a_{\rm c2}$, the optimal overlap is not 1 but $R_{\rm opt} = R_*$. Within the procedure of optimal learning discussed above, the overlap R is forced to increase toward 1 for any positive initial overlap $R > 0$. This can be seen directly by approximating (4.109) around $R = 0$ as

$$\frac{dR}{d\alpha} = \frac{(1-2\Delta)^2}{2\pi}\frac{l^2}{R}. \tag{4.112}$$

On the other hand, if we select a negative R as the initial condition, R reaches -1 and we get the generalization error $1 - 2H(a)$ in the limit of $\alpha\to\infty$. However, this is not the optimal value. Therefore, by this approach the student cannot realize the best possible weight which satisfies $R_{\rm opt} = R_*$.

We note that the above optimal learning rate $g(\alpha) = [F(R)R - G(R)]\,l/RE$ contains the parameter a unknown to the student. Thus this choice of $g(\alpha)$ is not perfectly consistent with the principles of supervised learning. We will propose an improvement on this point in section 4.6 using a parameter-free learning rate. For the moment, we may take the result of the present section as a theoretical estimate of the best possible optimization result.

Asymptotic analysis of the learning curve

Let us first investigate the realizable case $\Delta = 0$ using the scaling $R = 1 - \varepsilon$. In this case, the asymptotic forms of ε l, $\epsilon_{\rm g}$ and g are obtained from the same analysis as in the previous section

$$\begin{aligned} \varepsilon &= \frac{8}{\alpha^2} \\ l &= c\,{\rm e}^{-16/\alpha^2} \end{aligned} \tag{4.113}$$

an

$$\epsilon_{\rm g} = \frac{4}{\pi\alpha} \tag{4.114}$$

$$g(\alpha) = 2\sqrt{2\pi}\,\frac{l}{\alpha} \tag{4.115}$$

$$= 2c\sqrt{2\pi}\frac{e^{-16/\alpha^2}}{\alpha} \tag{4.116}$$

where c is a constant depending on the initial condition. The decay rate to vanishing generalization error is improved from $\alpha^{-1/3}$ for the unoptimized case [81] to α^{-1}. This α^{-1}-law is the same as in the off-line (or batch) learning [91, 95]. We also see that l approaches c as R reaches 1.

We next investigate the unrealizable case $\Delta \neq 0$. The asymptotic forms of ϵ, l and ϵ_g are

$$\begin{aligned} \epsilon &= \frac{2\pi H(a)}{(1-2\Delta)^2}\frac{1}{\alpha} \\ l &= c\,\alpha^{-2\Delta/(1-2\Delta)} \end{aligned} \tag{4.117}$$

$$\epsilon_g = \frac{\sqrt{2}}{\pi}\frac{\sqrt{2\pi H(a)}}{1-2\Delta}\frac{1}{\sqrt{\alpha}} + 2H(a) \tag{4.118}$$

and the optimal learning rate $g_{\rm opt}$ is

$$\begin{aligned} g_{\rm opt}(\alpha) &= \frac{2\pi H(a)}{\sqrt{2H(a)} + \sqrt{2\pi\alpha}H(a)(1-2\Delta)}\frac{l}{\alpha} \\ &= c\frac{2\pi H(a)}{\sqrt{2H(a)} + \sqrt{2\pi\alpha}H(a)(1-2\Delta)}\frac{\alpha^{-2\Delta/(1-2\Delta)}}{\alpha} \\ &\simeq c\frac{\sqrt{2\pi}}{1-2\Delta}\frac{\alpha^{-2\Delta/(1-2\Delta)}}{\alpha}. \end{aligned} \tag{4.119}$$

From the asymptotic form of l, we find that l diverges with α for $a < a_{c1} = \sqrt{2\log 2}$ and goes to zero for $a > a_{c1}$ as observed in the previous subsection. It is interesting that, for a exactly equal to a_{c1}, $g_{\rm opt}$ vanishes as seen from equations (4.5), (4.6) and (4.107) and so does $dl/d\alpha$ and $dR/d\alpha$. The present type of optimization does not make sense in this case.

For $a > a_{c2} = 0.80$, the generalization error converges to the optimal value $2H(a)$ as $\alpha^{-1/2}$. This is the same exponent as that of the Hebbian learning as we saw in the previous section. For $a < a_{c2}$, in order to get the optimal configuration of the student weight vector with $R = R_*$, which gives the optimal generalization error, we must stop the on-line dynamics before the system reaches the state $R = -1$. Accordingly, the method discussed in this section is not useful for the purpose of improvement of generalization ability for $a < a_{c2}$.

4.4.2 Hebbian learning

Trajectory in the R-l plane

The Hebbian learning with learning rate $g(\alpha)$ is

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + g(\alpha)T_a(v)\,\boldsymbol{x}. \tag{4.120}$$

The corresponding differential equations are

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{g^2 E_{\rm H}}{2} + gF_{\rm H}l\right] \tag{4.121}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2}g^2 E_{\rm H} - g(F_{\rm H}R - G_{\rm H})l\right]. \tag{4.122}$$

Using the same technique as in the previous subsection, we find the optimal learning rate for the Hebbian learning $g^{\rm H}_{\rm opt}(\alpha)$ as

$$g^{\rm H}_{\rm opt}(\alpha) = -\frac{[F_{\rm H}(\alpha)R - G_{\rm H}(\alpha)]\,l}{RE_{\rm H}(R)} \tag{4.123}$$

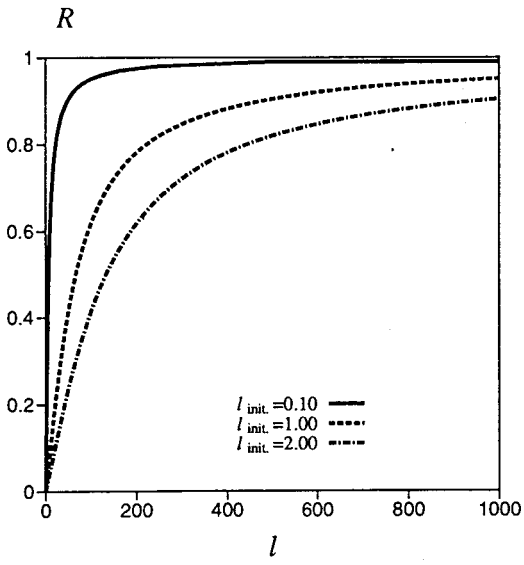

Figure 4.28: The trajectory in the R-l space for the Hebbian learning with the optimal learning rate.

where $E_{\rm H}(\alpha) = 1$, $F_{\rm H}(\alpha) = 2R(1-2\Delta)/\sqrt{2\pi}$ and $G_{\rm H}(\alpha) = 2(1-2\Delta)/\sqrt{2\pi}$. Substituting $g^{\rm H}_{\rm opt}(\alpha)$ into equations (4.121) and (4.122), we obtain

$$\frac{dR}{d\alpha} = \frac{(F_{\rm H}\,R - G_{\rm H})^2}{RE_{\rm H}} \tag{4.124}$$

$$\frac{dl}{d\alpha} = -\frac{(F_{\rm H}\,R - G_{\rm H})(F_{\rm H}\,R + G_{\rm H})l}{2R^2E_{\rm H}}. \tag{4.125}$$

The ratio gives

$$\frac{dR}{dl} = -\frac{(F_{\rm H}R - G_{\rm H})}{(F_{\rm H}R + G_{\rm H})}\frac{R}{l} = \frac{1-R^2}{1+R^2}\frac{R}{l}. \tag{4.126}$$

We thus obtain the R-l trajectory as

$$\frac{R}{(1-R^2)} = c\,l \tag{4.127}$$

where c is a constant determined by the initial condition. It is very interesting that this trajectory is independent of a. We plot this result in Figure 4.28.

Asymptotic analysis of the learning curve

The asymptotic forms of various quantities for $a > a_{\rm c1}$ of the Hebbian learning are

$$\varepsilon = \frac{\pi}{4(1-2\Delta)^2}\frac{1}{\alpha}$$
$$l = c\,\alpha \tag{4.128}$$

and

$$\epsilon_{\rm g} = \frac{1}{\sqrt{2\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + 2H(a) \tag{4.129}$$

$$g(\alpha) = c. \tag{4.130}$$

Accordingly, for $a > a_{c1}$, the asymptotic form of the generalization error is the same as for $g = 1$, the unoptimized case. However, for the parameter region $a < a_{c1}$, the generalization ability of the student deteriorates by introducing the optimal learning rate if we select the initial condition satisfying $R > 0$. To see this, we note from equation (4.124) that $dR/d\alpha$ is approximated around $R = 0$ as G_H^2/R. Therefore if we start the learning dynamics from $R > 0$, the overlap R goes to 1 and the generalization error approaches $2H(a)$ which is not acceptable at all because it exceeds 0.5. On the other hand, for $a < a_{c1}$, the generalization error approaches $1 - 2H(a)$, not the optimal value, as

$$\epsilon_g = \frac{1}{\sqrt{2\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}} + 1 - 2H(a). \tag{4.131}$$

Thus an over-training appears. We must notice that the prefactor of the generalization error changes from $1/\sqrt{6\pi}$ in equation (4.44) to $1/\sqrt{2\pi}$ in equation (4.131) by introducing the optimal learning rate. Therefore the optimization by using the learning rate $g(\alpha)$ is not very useful for the Hebbian learning.

4.4.3 AdaTron learning

In the previous subsection, we saw that the on-line AdaTron learning fails to get the best possible value of the generalization error for the unrealizable case and its residual error $\epsilon_{\min}$ is larger than that of the conventional perceptron learning or Hebbian learning. We show that it is possible to overcome this difficulty using the time dependent learning rate $g(\alpha)$. In order to determine g using the above strategy, we maximize the right hand side of equation (4.47) with respect to $g(\alpha)$ and obtain $g_{\rm opt} = (E_{\rm Ad}R - G_{\rm Ad})/RE_{\rm Ad}$. Using this optimal learning rate, equations (4.46) and (4.47) are rewritten as follows

$$\frac{dl}{d\alpha} = -\frac{(E_{\rm Ad}R - G_{\rm Ad})(E_{\rm Ad}R + G_{\rm Ad})}{2R^2E_{\rm Ad}}l \tag{4.132}$$

$$\frac{dR}{d\alpha} = \frac{(E_{\rm Ad}R - G_{\rm Ad})^2}{2RE_{\rm Ad}}. \tag{4.133}$$

For the realizable case, we obtain the asymptotic form of the generalization error from equations (4.132) and (4.133) by the same relation $R = 1 - \varepsilon$, $\varepsilon \to 0$ as we used for the case of $g = 1$ as

$$\epsilon_g = \frac{4}{3\alpha}. \tag{4.134}$$

This is the same asymptotic behavior as that obtained by optimizing the initial value of l as we saw in subsection 4.2.3.

Next we investigate the unrealizable case. The asymptotic forms of $E_{\rm Ad}$ and $E_{\rm Ad}R - G_{\rm Ad}$ in the limit of $\alpha \to \infty$ are obtained as

$$E_{\rm Ad} \sim 2H(a) + \sqrt{\frac{2}{\pi}}a\Delta \tag{4.135}$$

and

$$E_{\rm Ad}R - G_{\rm Ad} \sim -\frac{4a\varepsilon\Delta}{\sqrt{2\pi}}. \tag{4.136}$$

Then we get the asymptotic solution of equation (4.133) with respect to ε, $R = 1 - \varepsilon$, as

$$\varepsilon = \frac{2\pi H(a) + \sqrt{2\pi}\,a\,\Delta}{4a^2\Delta}\frac{1}{\alpha}. \tag{4.137}$$

As the asymptotic behavior of $E(R)$ is obtained as $E(R) = \epsilon_g = 2H(a) + \sqrt{2\varepsilon}/\pi$ [37], we find the generalization error in the limit of $\alpha \to \infty$ as follows

$$\epsilon_g = 2H(a) + \frac{\sqrt{2}}{\pi}\sqrt{\frac{2\pi H(a) + \sqrt{2\pi}a\Delta}{4a^2\Delta}}\frac{1}{\sqrt{\alpha}}, \tag{4.138}$$

where $2H(a)$ is the best possible value of the generalization error for $a > a_{c2}$. Therefore, our strategy to optimize the learning rate succeeds in training the student to obtain the optimal overlap $R = 1$ for $a > a_{c2}$.

For the perceptron learning, this type of optimization failed to reach the theoretical lower bound of the generalization error for a exactly at $a = a_{c1} = \sqrt{2\log 2}$ in which case the generalization error is $\epsilon_g = 1/2$, equivalent to a random guess because for $a = a_{c1}$ the optimal learning rate vanishes [37]. In contrast, for the AdaTron learning, the optimal learning rate has a non-zero value even at $a = a_{c1}$. In this sense, the on-line AdaTron learning with optimal learning rate is superior to the perceptron learning.

4.5 Optimized learning with output noise

We next investigate the generalization ability of optimized learning under output noise. In this section, we restrict ourselves to the case of the perceptron and Hebbian learnings because the result of the AdaTron learning is almost same as the perceptron learning.

4.5.1 Perceptron learning

The realizable case with output noise was investigated by Barkai et al [81]. They assumed the form of the learning rate as $g(\alpha) \sim \eta\alpha^{-z}$ and optimized η and z to obtain the asymptotic form of the prediction error as

$$\epsilon_p = \lambda + \sqrt{\frac{2\lambda}{\pi\alpha}}. \tag{4.139}$$

We here discuss the optimization of the realizable and unrealizable cases in the presence of output noise.

Trajectory in the R-l plane

Using the same technique as in the previous section, we find the optimal learning rate as $g(\alpha) = (\tilde{F}R - \tilde{G})/R\tilde{E}$ and the trajectory as

$$(1-R)^{-(1+A_\lambda)/A_\lambda}(1+R)^{(1-A_\lambda)/A_\lambda} = cl \tag{4.140}$$

where $A_\lambda \equiv (2\lambda - 1)\Delta$, and c is a constant depending on the initial condition. In Figure 4.29 we plot this trajectory for $a = 2.0$ and the noise levels $\lambda = 0, 0.10$ and 0.20. We also show the corresponding generalization error in Figure 4.30.

Asymptotic analysis of the learning curve

For $a = \infty$ the order parameter R monotonically increases, and hence we use the scaling relation $R = 1-\varepsilon$. We insert this definition into $\tilde{E}$, $\tilde{F}$ etc in equation (4.58) to obtain

$$\begin{aligned} \tilde{F}R - \tilde{G} &= (1-2\lambda)\sqrt{\frac{2}{\pi}}\varepsilon \\ R\tilde{E} &= (1-\varepsilon)\left\{\lambda + (1-2\lambda)\frac{\sqrt{2\varepsilon}}{\pi}\right\}. \end{aligned} \tag{4.141}$$

Substituting these expressions into the differential equations (4.108) and (4.109) and integrating them, we obtain the asymptotic forms of ε, l and the prediction error ϵ_p as

$$\begin{aligned} \varepsilon &= \frac{\pi\lambda}{(1-2\lambda)^2}\frac{1}{\alpha} \\ l &= c\alpha^{-4\lambda/2(1-2\lambda)} \end{aligned} \tag{4.142}$$

$$\epsilon_p = \lambda + \sqrt{\frac{2\lambda}{\pi\alpha}} \tag{4.143}$$

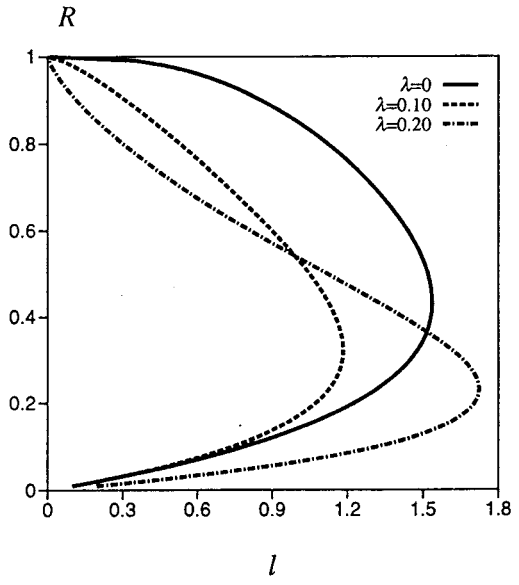

Figure 4.29: The trajectory of R and l of the optimal perceptron learning with output noise is plotted for the cases of $a = 2.0$ and $\lambda = 0, 0.10$ and 0.20.

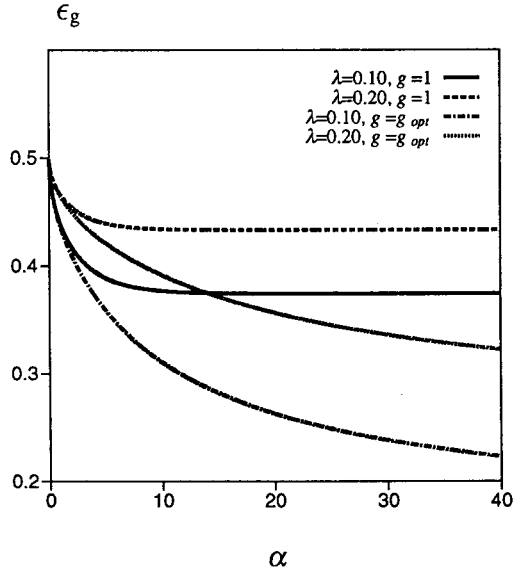

Figure 4.30: The generalization error corresponding to the previous figure.

where the generalization error is obtained by the relation $\epsilon_{\rm p} = \lambda + (1-2\lambda)\epsilon_{\rm g}$. This agrees with the result of Barkai *et al* [81]. The optimal learning rate is given as

$$\begin{aligned} g_{\rm opt}(\alpha) &= c\frac{(1-2\lambda)\sqrt{2/\pi}\,\varepsilon}{(1-\varepsilon)\{\lambda+(1-2\lambda)\sqrt{2\epsilon}/\pi\}}\alpha^{-4\lambda/2(1-2\lambda)} \\ &\simeq c\frac{\sqrt{2\pi}}{1-2\lambda}\frac{\alpha^{-4\lambda/2(1-2\lambda)}}{\alpha}. \end{aligned} \tag{4.144}$$

From this and equation (4.119), we see that the output noise level λ plays the same role as $\Delta = {\rm e}^{-a^2/2}$ plays in the noiseless case. The asymptotic form of l indicates that l goes to 0 for any initial condition.

For the unrealizable case, using the same scaling relation as in the realizable case $R = 1-\varepsilon$, we can derive the asymptotic forms of ϵ, l and the prediction error $\epsilon_{\rm p}$ as

$$\begin{aligned} \varepsilon &= \frac{\pi(2H(a)+\lambda-4H(a)\lambda)}{(1-2\lambda)^2(1-2\Delta)^2}\frac{1}{\sqrt{\alpha}} \\ l &= c\alpha^{-[1-2(1-2\Delta)(1-2\lambda)]/2(1-2\lambda)(1-2\Delta)} \end{aligned} \tag{4.145}$$

and

$$\epsilon_{\rm p} = 2H(a)+\lambda-4H(a)\lambda+\frac{\sqrt{2(2H(a)+\lambda-4H(a)\lambda)}}{\sqrt{\pi}(1-2\Delta)}\frac{1}{\sqrt{\alpha}}. \tag{4.146}$$

The generalization error $\epsilon_{\rm g}$ is obtained by the relation $\epsilon_{\rm p} = \lambda + (1-2\lambda)\epsilon_{\rm g}$. The optimal learning rate is

$$\begin{aligned} g_{\rm opt}(\alpha) &= \frac{\sqrt{2\pi}}{(1-2\lambda)(1-2\Delta)}\frac{l}{\alpha} \\ &= c\frac{\sqrt{2\pi}}{(1-2\lambda)(1-2\Delta)}\frac{\alpha^{-[1-2(1-2\lambda)(1-2\Delta)]/2(1-2\lambda)(1-2\Delta)}}{\alpha}. \end{aligned} \tag{4.147}$$

The functional dependence of $\epsilon_{\rm p}$ and $g_{\rm opt}$ on α is similar to the realizable case in equations (4.143) and (4.144).

4.5.2 Hebbian learning

For the Hebbian learning algorithm, the effect of noise appears in the optimized differential equations in the form $(1-2\lambda)(1-2\Delta)$. Hence all the relevant quantities can be derived simply by replacing $(1-2\Delta)$ with $(1-2\Delta)(1-2\lambda)$. Consequently, we easily see that the *R-l* trajectory, the learning rate and the generalization error all have the same form as those in the noiseless case. As we showed in the noiseless case, the present type of optimization for the Hebbian learning does not work very effectively.

4.6 Optimal learning without unknown parameters

As we mentioned in section 4.5, the generalization error obtained there is the theoretical (not practical) lower bound because the optimal learning rate $g_{\rm opt}$ contains a parameter a unknown to the student. In this section we propose a method to avoid this difficulty for the perceptron and AdaTron learning algorithms.

4.6.1 Perceptron learning

Noiseless case

For the noiseless case we choose the learning rate g as

$$g = \frac{k}{\alpha}l \tag{4.148}$$

which is nothing but the asymptotic form (4.115) of the previous optimized learning rate in section 4.5. Substituting this into equation (4.109) and using the scaling relation $R = 1 - \varepsilon$, we obtain for the realizable case

$$\frac{d\varepsilon}{d\alpha} = \frac{1}{2}\frac{\sqrt{2\varepsilon}}{\pi}\frac{k^2}{\alpha^2} - \sqrt{\frac{2}{\pi}}\varepsilon\frac{k}{\alpha}. \tag{4.149}$$

We assume ε as $\varepsilon = C/\alpha^n$ and determine C and n so that these satisfy equation (4.149). We then find $n = 2$ and

$$C = \frac{k^2}{4\pi(\sqrt{2\pi} - k)^2}. \tag{4.150}$$

The constant k is determined to minimize C. The result is $k = 2\sqrt{2\pi}$ for which $C = 8$. Consequently, we get $\varepsilon = 8/\alpha^2$ and

$$\epsilon_g = \frac{4}{\pi\alpha} \tag{4.151}$$

which agrees with the result of Barkai *et al* [81].

We next investigate the asymptotic generalization error for the unrealizable case. We assume $g(\alpha) = kl/\alpha$ as before and substitute it to equation (4.109) to find

$$-\frac{d\varepsilon}{d\alpha} \simeq -\frac{k^2 H(a)}{\alpha^2} + \sqrt{\frac{2}{\pi}}(1 - 2\Delta)\frac{k\varepsilon}{\alpha}. \tag{4.152}$$

The general solution is

$$\varepsilon = \frac{k^2 H(a)}{bk - 1}\frac{1}{\alpha} + A\left(\frac{k}{\alpha}\right)^{bk} \tag{4.153}$$

where $b \equiv \sqrt{2/\pi}(1 - 2\Delta)$. The first term dominates asymptotically if $bk > 1$. In this case, we have

$$\epsilon_g = 2H(a) + \sqrt{\frac{2k^2 H(a)}{bk - 1}}\frac{1}{\pi\sqrt{\alpha}}. \tag{4.154}$$

The second term on the right-hand side is minimized by choosing

$$k = \frac{\sqrt{2\pi}}{1 - 2\Delta} \tag{4.155}$$

which satisfies $bk > 1$ as required. Equation (4.154) makes sense for $\Delta > 2\sqrt{\log 2}$ if k is chosen as above.

When $bk < 1$, the asymptotic form of the generalization error is

$$\epsilon_g = 2H(a) + \frac{\sqrt{2A}}{\pi}\left(\frac{\sqrt{2\pi}}{\alpha}\right)^{bk/2}. \tag{4.156}$$

This formula is valid for $b > 0$ or $a < a_{c1}$. Similar crossover between two types of asymptotic forms was reported in the problem of one-dimensional decision boundary [18].

Noisy case

Next we investigate the noisy case. If there is noise, unknown parameters for the student are not only the width a of reversed wedge but also the noise level λ. We therefore first assume the learning rate $g = kl/\alpha$. Using (4.148), the differential equation of $\varepsilon = 1 - R$ is

$$-\frac{d\varepsilon}{d\alpha} \simeq -\frac{k^2}{2\alpha^2}(2H(a) + \lambda - 4H(a)\lambda) + \frac{2}{\sqrt{2\pi}}(1 - 2\Delta)(1 - 2\lambda)\frac{k\varepsilon}{\alpha}. \tag{4.157}$$

The general solution is

$$\varepsilon = \frac{(k^2/2)(2H(a) + \lambda - 4H(a)\lambda)}{\left[\sqrt{2/\pi}(1 - 2\Delta)(1 - 2\lambda)k - 1\right]}\frac{1}{\alpha} + A\left(\frac{k}{\alpha}\right)^{\sqrt{2/\pi}(1-2\Delta)(1-2\lambda)k} \tag{4.158}$$

where A is a constant. If we choose k so that it satisfies $\sqrt{2/\pi}(1-2\Delta)(1-2\lambda)k > 1$, the prediction error decays to the minimum value $2H(a)+\lambda-4H(a)\lambda$ as

$$\epsilon_{\rm p} = \epsilon_{\rm min} + \frac{\sqrt{2}}{\pi}\sqrt{\frac{(k^2/2)\,(2H(a)+\lambda-4H(a)\lambda)}{\left[\sqrt{2/\pi}(1-2\Delta)(1-2\lambda)k-1\right]}}\frac{1}{\sqrt{\alpha}}. \tag{4.159}$$

If we set $\Delta = 0$, this agrees with the result of Barkai *et al* [81] for the realizable case. On the other hand, if $\sqrt{2/\pi}(1-2\Delta)(1-2\lambda)k < 1$,

$$\epsilon_{\rm p} = \epsilon_{\rm min} + \frac{\sqrt{2A}}{\pi}\left(\frac{k}{\alpha}\right)^{(1-2\Delta)(1-2\lambda)k/\sqrt{2\pi}} \tag{4.160}$$

which is slower than $\alpha^{-1/2}$.

When the student knows the noise level λ but does not know Δ, we can choose $k = \sqrt{2\pi}/(1-2\lambda)$. Then, the convergence of equation (4.159) is achieved for $a > 2\sqrt{\log 2}$. The convergence of equation (4.160) is observed for $a < 2\sqrt{\log 2}$. When the student knows neither λ nor Δ, the convergence depends upon the value of $2(1-2\lambda)(1-2\Delta)k$.

4.6.2 AdaTron learning

In the previous section, we saw that the AdaTron learning is able to obtain the theoretical lower bound of the generalization error for $a > a_{c2}$ by introducing the optimal learning rate $g_{\rm opt}$. In this subsection, we construct a learning algorithm without the unknown parameter a using the asymptotic form of the optimal learning rate in the AdaTron learning.

Realizable case

For the realizable case, the optimal learning rate is estimated in the limit of $\alpha\to\infty$ as

$$g_{\rm opt} = \frac{E_{\rm Ad}R - G_{\rm Ad}}{RE_{\rm Ad}}l \simeq \frac{3}{2}l. \tag{4.161}$$

This asymptotic form of the optimal learning rate depends on α only through the length l of student's weight vector. We therefore adopt $g(\alpha)$ proportional to l, $g(\alpha) = \eta\, l$, also in the case of the parameter-free optimization and adjust the parameter η so that the student obtains the best generalization ability. Substituting this expression into the differential equation (4.47) for R and using $R = 1-\varepsilon$ with $\varepsilon\to 0$, we get

$$\frac{d\varepsilon}{d\alpha} = -F(\eta)\,\epsilon^{3/2}. \tag{4.162}$$

where we have set

$$F(\eta) \equiv \frac{2\sqrt{2}}{\pi}\eta - \frac{4}{3\sqrt{2\pi}}\eta^2. \tag{4.163}$$

This leads to $\varepsilon = (F(\eta)/2)^{-2}\alpha^{-2}$. Then, the generalization error is obtained from $\epsilon_{\rm g} = \sqrt{2\varepsilon}/\pi$ as

$$\epsilon_{\rm g} = \frac{2\sqrt{2}}{\pi F(\eta)}\frac{1}{\alpha}. \tag{4.164}$$

In order to minimize $\epsilon_{\rm g}$, we maximize $F(\eta)$ with respect to η. The optimal choice of η in this sense is $\eta_{\rm opt} = 3/2$ and we find in such a case

$$\epsilon_{\rm g} = \frac{4}{3\alpha}. \tag{4.165}$$

This is the same asymptotic form as the previous a-dependent result (4.134).

Unrealizable case

Next we consider the unrealizable case. The asymptotic form of the learning rate we derived in the previous section for the unrealizable case is

$$g_{\rm opt} = \frac{E_{\rm Ad}R - G_{\rm Ad}}{RE_{\rm Ad}} \simeq -\frac{4a\varepsilon\Delta/\sqrt{2\pi}}{2H(a)+\sqrt{2/\pi}a\Delta}l = \eta\frac{l}{\alpha}, \tag{4.166}$$

where we used equation (4.137) to obtain the right-most equality and we set the a-dependent prefactor of l as η. Using this learning rate (4.166) and the asymptotic forms of $E_{\rm Ad}(R=1-\varepsilon,\varepsilon\to 0)$ and $G_{\rm Ad}(R=1-\varepsilon,\varepsilon\to 0)$ as $E_{\rm Ad}\sim 2H(a)+\sqrt{2/\pi}a\Delta$ and $G_{\rm Ad}\sim 4a\Delta\varepsilon/\sqrt{2\pi}+E_{\rm Ad}$ in the limit of $\alpha\to\infty$, we obtain the differential equation with respect to ε from equation (4.47) as follows

$$\frac{d\varepsilon}{d\alpha} = \frac{1}{2}\left[2H(a)+\sqrt{\frac{2}{\pi}}a\Delta\right]\frac{\eta^2}{\alpha^2} - \eta\frac{4a}{\sqrt{2\pi}}\Delta\frac{\varepsilon}{\alpha}. \tag{4.167}$$

This differential equation can be solved analytically as

$$\varepsilon = \frac{\eta^2\left(2H(a)+\sqrt{2/\pi}a\Delta\right)}{2\left(4a\Delta\eta/\sqrt{2\pi}-1\right)}\frac{1}{\alpha} + A\left(\frac{\eta}{\alpha}\right)^{4a\Delta\eta/\sqrt{2\pi}}, \tag{4.168}$$

where A is a constant determined by the initial condition. Therefore, if we choose η to satisfy $4a\Delta\eta/\sqrt{2\pi}-1>0$, the generalization error converges to the optimal value $2H(a)$ as

$$\begin{aligned}\epsilon_{\rm g} &= 2H(a)+\frac{\sqrt{2\varepsilon}}{\pi}\\ &= 2H(a)+\frac{\eta}{\pi}\sqrt{\frac{2H(a)+\sqrt{2/\pi}a\Delta}{4a\Delta\eta/\sqrt{2\pi}-1}}\frac{1}{\sqrt{\alpha}}.\end{aligned} \tag{4.169}$$

In order to obtain the best generalization ability, we minimize the prefactor of $1/\sqrt{\alpha}$ in the second term of equation (4.169) and obtain

$$\eta = \sqrt{\frac{\pi}{2}}\frac{\Delta}{a}. \tag{4.170}$$

For this η, the condition $4a\Delta\eta/\sqrt{2\pi}-1>0$ is satisfied. In general, if we take η independent of a, the condition $4a\Delta\eta/\sqrt{2\pi}-1>0$ is not always satisfied. The quantity $b\equiv 4a\Delta/\sqrt{2\pi}$ takes the maximum value $4/\sqrt{2\pi e}$ at $a=1$. Therefore, whatever value of a we choose, we cannot obtain the $\alpha^{-1/2}$ convergence if the product of this maximum value $4/\sqrt{2\pi e}$ and η is not larger than unity. This means that η should satisfy $\eta>\sqrt{2\pi e}/4\simeq 1.033$ for which the first term of equation (4.168) dominate asymptotically, yielding equation (4.169), for a non-vanishing range of a. In contrast, if we choose η to satisfy $b\eta-1<0$, the generalization error is dominated by the second term of equation (4.168) and behaves as

$$\epsilon_{\rm g} = 2H(a)+\frac{\sqrt{2A}}{\pi}\left(\frac{\eta}{\alpha}\right)^{2a\Delta\eta/\sqrt{2\pi}}. \tag{4.171}$$

In this case, the generalization error converges less quickly than (4.169). For example, if we choose $\eta=1$, we find that the condition $b\eta>1$ cannot be satisfied by any a and the generalization error converges as in equation (4.171). If we set $\eta=2$ ($>\sqrt{2\pi e}/4=1.033$) as another example, the asymptotic form of the generalization error is either equation (4.169) or equation (4.171) depending on the value of a.

4.7 Hebbian learning with queries

We have so far assumed that the student is trained using examples drawn from a uniform distribution on the N-dimensional sphere S^N. It is known for the realizable case [96] that selecting training examples out of a limited set sometimes improves the performance of learning. We therefore investigate in the present section how the method of Kinzel and Ruján [96] works for an unrealizable rule.

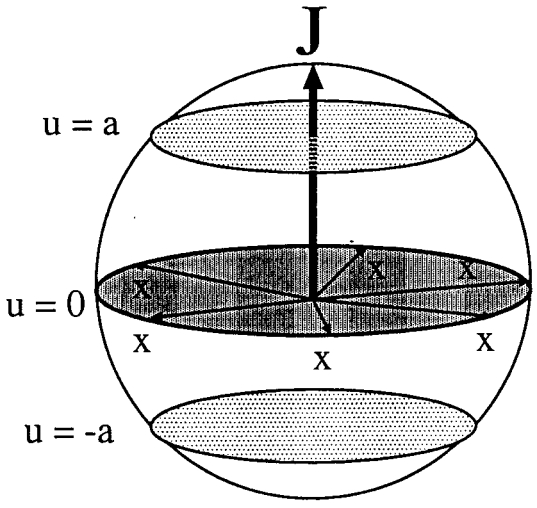

Figure 4.31: Queries lie on the borderline $u = 0$.

4.7.1 Learning with queries under fixed learning rate

The learning dynamics we choose here is nothing but the Hebbian algorithm

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + T_a(v)\boldsymbol{x}. \tag{4.172}$$

In section 4.3, the student was trained by inputs $\boldsymbol{x}$ uniform on S^N. In the present section we follow reference [96] and use selected inputs which lie on the borderline, $\boldsymbol{J}\cdot\boldsymbol{x} = 0$ or $u = 0$, at every dynamical step (see Figure 4.31). The idea behind this choice is that the student is not confident for inputs just on the decision boundary and thus teacher signals for such examples should be more useful than generic inputs.

We use the following conditional distribution, instead of $P_R(u, v)$ in equation (2.5), in order to get the differential equations

$$P_R(v|u = 0) = \sqrt{2\pi}\delta(u)P_R(u, v). \tag{4.173}$$

Using this distribution, we obtain the next differential equations with respect to l and R.

$$\frac{dl^2}{d\alpha} = 1 \tag{4.174}$$

$$\frac{dR}{d\alpha} = \frac{1}{l}\left[\sqrt{\frac{2}{\pi}}\sqrt{1-R^2}\left\{1 - 2\exp\left(-\frac{a^2}{2(1-R^2)}\right)\right\} - \frac{R}{2l}\right]. \tag{4.175}$$

We plotted the generalization error for several values of the width a of reversed wedge, in Figures 4.32 ($a = \infty$), 4.33 ($a = 2.0$) and 4.34 ($a = 1.0$), by numerical integration of the above differential equations. From these figures we see that the generalization ability of student is improved both for realizable and unrealizable cases. In particular for $a = 1.0$, the problem of over-training observed in subsection 4.2 is avoided. In order to investigate the asymptotic form of the generalization error, we solve the differential equations in the limit of $\alpha\to\infty$. Equation (4.174) can be solved easily as

$$l = \sqrt{\alpha}. \tag{4.176}$$

For the realizable case $a\to\infty$, using $R = 1 - \varepsilon$ and $\varepsilon\to 0$, we get

$$-\sqrt{\alpha}\frac{d\varepsilon}{d\alpha} = \frac{2}{\sqrt{\pi}} - \frac{1}{2\sqrt{\alpha}}. \tag{4.177}$$

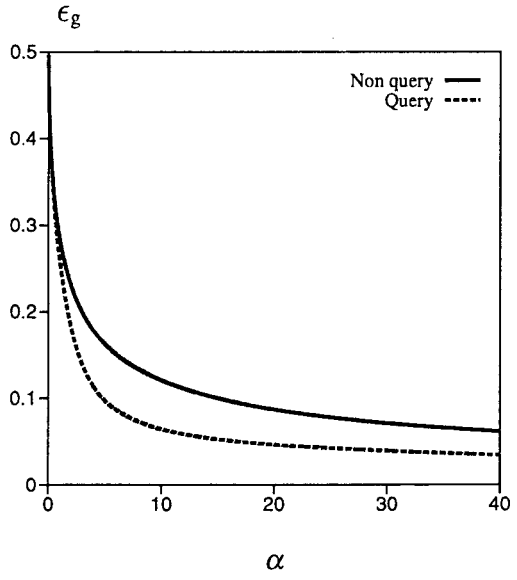

Figure 4.32: Generalization error of the Hebbian learning with queries for $a = \infty$.

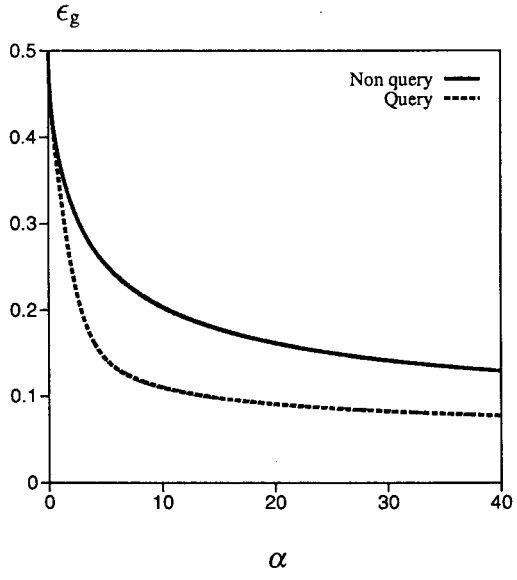

Figure 4.33: Generalization error of the Hebbian learning with queries for $a = 2.0$.

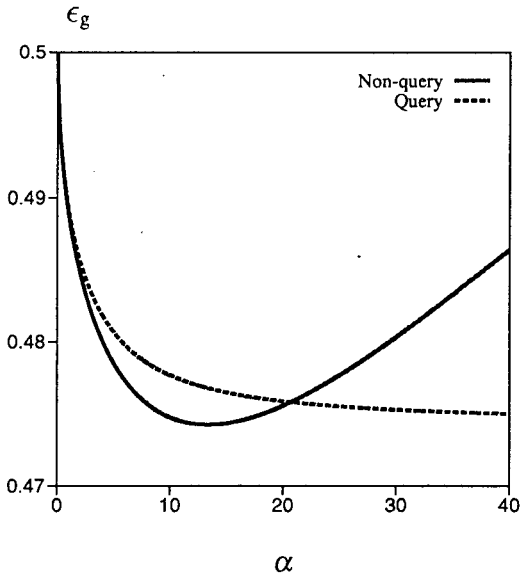

Figure 4.34: Generalization error of the Hebbian learning with queries for $a = 1.0$. Over-training disappeared and the generalization error converges to its optimal value.

This can be solved as $\varepsilon = \pi/(16\alpha)$. The generalization error is

$$\epsilon_g = \frac{1}{2\sqrt{2\pi}} \frac{1}{\sqrt{\alpha}}. \tag{4.178}$$

The numerical prefactor has been reduced by a half compared to the result of the conventional Hebbian learning (4.42).

For finite a, equation (4.175) has fixed points at $R_0 = \pm 1$ and

$$R_1^{(\pm)} = \pm \sqrt{\frac{2\log 2 - a^2}{2\log 2}}. \tag{4.179}$$

The latter fixed point exists only for $a < a_{c1} = \sqrt{2\log 2}$. Thus, if $a > a_{c1}$, $|R|$ eventually approaches 1, and the exponential term in equation (4.175) can be neglected. This implies that the asymptotic analysis for the realizable case applies without modification. The resulting asymptotic form of the generalization error is

$$\epsilon_g = \frac{1}{2\sqrt{2\pi}} \frac{1}{\sqrt{\alpha}} + 2H(a). \tag{4.180}$$

If $a < a_{c1}$, the system is attracted to the fixed point $R_1^{(-)}$ according to the expansion of the right-hand side of equation (4.175) around $R = 0$,

$$\frac{dR}{d\alpha} \simeq \frac{1}{l}\sqrt{\frac{2}{\pi}}(1 - 2\Delta) \tag{4.181}$$

which is negative if $a < a_{c1}$. It is remarkable that $R_1^{(-)}$ coincides with R_* which gives the global minimum of $E(R)$ for $a < a_{c2} = 0.80$. Therefore, for $a < a_{c2}$, the present Hebbian learning with queries achieves the best possible generalization error. In the range $a_{c2} < a < a_{c1}$, $R = R_1^{(-)} = R_*$ is not the global minimum of $E(R)$ but is only a local minimum. However, as seen in Figure 4.34 , over-training has disappeared in this region by introducing queries.

The asymptotic behavior for $a < a_{c1}$ is found to be

$$\epsilon_g = \epsilon_{\rm opt} - \frac{16\log 2\sqrt{2\log 2 - a^2}}{a^2}\left[1 - Q(2, \frac{1}{2}\log 2)\right]$$

$$\times \exp\left[-\frac{8\log 2}{\sqrt{\pi}a}\sqrt{2\log 2 - a^2}\sqrt{\alpha}\right] \tag{4.182}$$

where $Q(x,y)$ is the incomplete gamma function and the asymptotic value is optimal for $a < a_{c1}$:

$$\begin{aligned}\epsilon_{\rm opt} &= E(R_*) \\ &= 2\int_a^\infty Dv\, H\left(\frac{\sqrt{4\log 2 - 2a^2}}{a}v\right) + 2\int_0^a Dv\, H\left(-\frac{\sqrt{4\log 2 - 2a^2}}{a}v\right).\end{aligned} \tag{4.183}$$

4.7.2 Optimized Hebbian learning with queries

We next introduce the parameter g into the Hebbian learning with queries and optimize g so that R goes to 1 as quickly as possible. As discussed in section 4.4, this strategy works only for $a > a_{c2}$ since $R = 1$ is not the optimal value if $a < a_{c2}$. The on-line Hebbian learning is now written as

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + g(\alpha)T_a(v)\boldsymbol{x}. \tag{4.184}$$

Using the distribution $P_R(v|u=0)$, we obtain differential equations

$$2l\frac{dl}{d\alpha} = g^2 \tag{4.185}$$

$$l\frac{dR}{d\alpha} = g\sqrt{\frac{2}{\pi}}\sqrt{1-R^2}\left\{1 - 2\exp\left(-\frac{a^2}{2(1-R^2)}\right)\right\} - g^2\frac{R}{2l}. \tag{4.186}$$

The right-hand side is maximized with respect to g by

$$g = g_{\rm opt} = \frac{l}{R}\sqrt{\frac{2}{\pi}}\sqrt{1-R^2}\left\{1 - 2\exp\left(-\frac{a^2}{1-R^2}\right)\right\}. \tag{4.187}$$

If $a = \infty$, equation (4.186) is then written as

$$\frac{dR}{d\alpha} = \frac{1-R^2}{R\pi} \tag{4.188}$$

with the solution

$$R = \sqrt{1 - c\exp(-\frac{2\alpha}{\pi})} \tag{4.189}$$

where c is a constant. The generalization error decays to zero as

$$\epsilon_{\rm g} = \frac{\sqrt{c}}{\pi}\exp(-\frac{\alpha}{\pi}) \tag{4.190}$$

where c is determined by the initial condition. This exponential decrease for the realizable case is in agreement with reference [97] where the optimization of the type of equation (4.104) was used together with queries. The asymptotic forms of the order parameter l and optimal learning rate $g_{\rm opt}$ are

$$l = c'\sqrt{1 - c\exp(-\frac{2\alpha}{\pi})} \tag{4.191}$$

$$g_{\rm opt}(\alpha) = c'\sqrt{\frac{2c}{\pi}}\exp(-\frac{\alpha}{\pi}) \tag{4.192}$$

where c' is determined by the initial condition.

Next we investigate the case of finite a. Using the same asymptotic analysis as in the realizable case, we obtain the asymptotic form of generalization error $\epsilon_{\rm g}$ as

$$\epsilon_{\rm g} = 2H(a) + \frac{\sqrt{c}}{\pi}\exp(-\frac{\alpha}{\pi}). \tag{4.193}$$

The limiting value $2H(a)$ is the theoretical lower bound for $a > a_{c2} = 0.80$. We therefore have found a method of optimization to achieve the best possible generalization error with a very fast, exponential, asymptotic approach for $a > a_{c2}$. This convergence is faster than the result by Kim and Sompolinsky [84, 85, 86] who used the Gibbs on-line algorithm.

The present method of optimization does not work appropriately for $a < a_{c2}$ because $R = 1$, to which the present method is designed to force the system, is not the best value of R in this range of a.

It is worth investigating whether the exponent of decay changes or not by using a parameter-free optimal learning rate as in section 4.6. If $a > a_{c1}$, there exists only one fixed point $R = 1$. Therefore, the a-dependent term $\exp(-a^2/(1-R^2))$ in equation (4.187) does not affect the asymptotic analysis. We may, therefore, reasonably conclude that the asymptotic form of the generalization error does not change by the optimal learning rate without the unknown parameter a.

4.8 Avoiding over-training by a weight-decay term

We showed in section 4.3 that the over-training appears for the unrealizable case $a < a_{c1}$ by the Hebbian learning. If $a < a_{c1}$, the flow of R goes to -1 for any initial condition passing through the local minimum of $E(R)$ at $R = R_*$. Consequently, the generalization ability of the student decreases as he learns excessively. In order to avoid this difficulty, we must stop the dynamics on the way to the state $R = -1$.

The teacher output is frequently reversed for small a. Thus, as we mentioned in section 4.3, an example presented by the teacher sometime ago may invalidate the information of the most recent example which is most important for the student to generalize. To overcome this difficulty, we may use the on-line dynamics with a weight-decay term or a forgetting term. Recently, Biehl and Schwarze [88] investigated the generalization ability of learning of a time-dependent rule by the Hebbian algorithm with a weight-decay term $-\Lambda \boldsymbol{J}/N$. They optimized Λ so that the fixed point $R_\infty = R_0$ has the maximum value. In their system, as the student and his teacher have the same structure, the perfect generalization is achieved if R goes to 1. Using this optimization, they succeeded to overcome the over-training and obtained a better generalization ability. Our situation is slightly different from their case. We must train the student so that he reaches $R = R_*$ instead of $R = 1$. We nevertheless try their method expecting that our difficulties mentioned above may be avoided.

The on-line dynamics by the Hebbian rule is modified with the weight-decay term as

$$\boldsymbol{J}^{m+1} = (1 - \frac{\Lambda}{N})\boldsymbol{J}^m + T_a(v)\,\boldsymbol{x}. \tag{4.194}$$

The differential equations for R and l are then

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{1}{2} + \frac{2R}{\sqrt{2\pi}}(1-2\Delta)l - \Lambda l^2\right] \tag{4.195}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2} + \frac{2}{\sqrt{2\pi}}(1-2\Delta)(1-R^2)l\right]. \tag{4.196}$$

The fixed point (R_0, l_0) of these equations is

$$R_0 = \frac{2(1-2\Delta)}{\sqrt{\pi\Lambda + 4(1-2\Delta)^2}} \tag{4.197}$$

$$l_0 = \frac{\sqrt{\pi\Lambda + 4(1-2\Delta)^2}}{\sqrt{2\pi}\Lambda}. \tag{4.198}$$

In order to get the optimal value, we choose R_0 so that it agrees with R_* which gives the global minimum of $E(R)$ for $a < a_{c1}$. From this condition, we obtain the optimal $\Lambda_{\rm opt}$ as

$$\Lambda_{\rm opt} = \frac{4a^2(1-2\Delta)^2}{\pi(2\log 2 - a^2)} \tag{4.199}$$

and

$$l_0 = \frac{\sqrt{2\log 2 - a^2}}{2a^2(1-2\Delta)}\sqrt{\pi \log 2}. \tag{4.200}$$

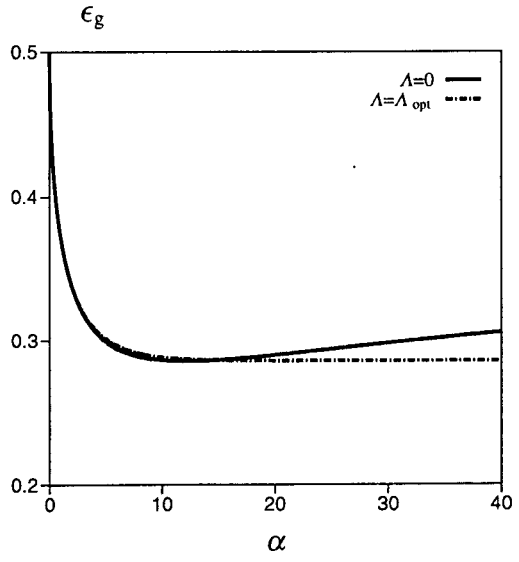

Figure 4.35: Generalization error of the Hebbian learning with a weight-decay term for $a = 0.5$. Over-training disappeared and generalization error converges to its optimal value.

Using this $\Lambda_{\rm opt}$, we solve the differential equations numerically and plot the result in Figure 4.35 for $a = 0.5(< a_{\rm cl})$. We see that the over-training disappears and the generalization error converges to the optimal value.

We next investigate how fast this convergence is achieved. For this purpose, we linearize the differential equations around the fixed point (R_0, l_0):

$$\frac{d\boldsymbol{u}}{d\alpha} = \boldsymbol{A}\,\boldsymbol{u} \tag{4.201}$$

where $\boldsymbol{u} = (l, \varepsilon)$ and matrix elements of $\boldsymbol{A}$ are given as

$$A_{11} = -2a^2(1-2\Delta)^2\left[\frac{\pi(2\log 2 - a^2)+4}{\pi(2\log 2 - a^2)}\right] \tag{4.202}$$

$$A_{12} = \frac{1}{\sqrt{2\pi}}(1-2\Delta) \tag{4.203}$$

$$A_{21} = \frac{8a^6(1-2\Delta)^3}{2\pi\sqrt{2\pi}\log 2\,(2\log 2 - a^2)} \tag{4.204}$$

$$A_{22} = -\left[\frac{1}{2} + \frac{2\log 2 - a^2}{a^2}\right] \tag{4.205}$$

where $A_{11}, A_{12}, A_{22} \gg A_{21}$. Therefore the eigenvalues are $\lambda = A_{11}, A_{22}$. Consequently, we obtain

$$\varepsilon \sim \varepsilon_0\left\{1 + \mathcal{O}\left[\exp(-2a^2(1-2\Delta)^2\left(\frac{\pi(2\log 2 - a^2)+4}{\pi(2\log 2 - a^2)}\right)\alpha)\right]\right\} \tag{4.206}$$

$$l \sim l_0\left\{1 + \mathcal{O}\left[\exp(-2a^2(1-2\Delta)^2\left(\frac{\pi(2\log 2 - a^2)+4}{\pi(2\log 2 - a^2)}\right)\alpha)\right]\right\}. \tag{4.207}$$

We warn here that $\Lambda_{\rm opt}$ in equation (4.199) depends on a which is unknown to the student. Therefore, the result obtained in this section gives the theoretical upper bound of the generalization ability.

4.9 Summary

We have analyzed the problem of on-line learning by the perceptron, Hebbian and AdaTron algorithms. The teacher transfer function is non-monotonic and its non-monotonicity is controlled by the width a of

the reversed wedge. If $a = \infty$, the rule is realizable, but is unrealizable when a is finite. We saw that for the realizable case, the asymptotic decay of the generalization error is proportional to $\alpha^{-1/3}$ for the perceptron learning, to $\alpha^{-1/2}$ for the Hebbian learning and to α^{-1} for the AdaTron learning. For the AdaTron learning, the generalization error decay to zero as $\epsilon_g = 3/(2\alpha)$ if we fix the length of the student wight vector as $l = J/\sqrt{N} = 1/2$ as Biehl and Riegler reported [98]. However, if we allow the time development of the length of the student weight vector, the asymptotic behavior of the generalization error shows dependence on the initial value of l. When the student starts the training process from the optimal length of the weight vector l, we can obtain the generalization error $\epsilon_g = 4/(3\alpha)$ which is a little faster than $3/(2\alpha)$. As the student is able to know the length of its own weight vector in principle, we can get the better generalization ability $\epsilon_g = 4/(3\alpha)$ by a heuristic search of the optimal value of l.

For the unrealizable case, the generalization error decays exponentially to a finite value $E(R_0)$ with $R_0 = 1 - 2\Delta$ in the case of the perceptron learning. For the AdaTron learning, the generalization error also exponentially converges to a non-optimal residual value which is larger than that of the perceptron learning. For the Hebbian learning, the generalization error decays to $2H(a)$, the best possible value, as $1/\sqrt{2\pi(1-2\Delta)^2\alpha}$ for $a > a_{c1}$ and to $1 - 2H(a)$ with $1/\sqrt{6\pi(1-2\Delta)^2\alpha}$ for $a < a_{c1}$. In this latter parameter region $a < a_{c1}$, we observed the phenomenon of over-training.

We also investigated the learning under output noise for the case of the perceptron and Hebbian learning algorithms. For the realizable case of the perceptron algorithm, the order parameters R and l are attracted toward a fixed point (R_0, l_0) asymptotically with an exponential law. As a result, the generalization error decays to a finite value exponentially. On the other hand, for the unrealizable case of the perceptron learning, the generalization error decays exponentially to a finite value $E((1-2\Delta)(1-2\lambda))$. For the Hebbian learning, the generalization error decays to $2H(a)$ in proportion to $1/\sqrt{\alpha}$ for $a > a_{c1}$ and to $1 - 2H(a)$ with also proportionally to $1/\sqrt{\alpha}$ for $a < a_{c1}$.

For both noisy and noiseless cases of the perceptron and Hebbian learning and for noiseless case of the AdaTron learnings, we introduced the learning rate $g(\alpha)$ in the on-line dynamics and optimized it to maximize $dR/d\alpha$. By this treatment we obtained a closed form trajectory of R and l. The generalization ability of the student has been shown to increase for $a > a_{c2} = 0.80$ in the case of the perceptron and AdaTron learning algorithms. For the unrealizable case, the generalization error decays to the best possible value $2H(a)$ in proportion to $1/\sqrt{\alpha}$ and the corresponding prediction error of perceptron learning decays to $2H(a) + \lambda - 4H(a)\lambda$ by a similar law. For the Hebbian learning, both in noisy and noiseless cases, the asymptotic generalization ability did not change by this optimization procedure.

Unfortunately, in the parameter range $a < a_{c2}$, we found it impossible to obtain an optimal performance for the perceptron and AdaTron learnings within our procedure of optimization. The reason is that we improved the dynamics of on-line learning so that the overlap between the student and the teacher approaches 1 or -1 as rapidly as possible by introducing an optimal learning rate $g_{\rm opt}$. Therefore, using this procedure, the student weight vector cannot converge to the optimal solution $\boldsymbol{J}^*$ which satisfies $R = R_*$ for $a < a_{c2}$. This is the same difficulty as in the over-training of the Hebbian learning. In order to avoid this problem, we must stop the dynamics on the way to $R = -1$. As an idea to overcome this difficulty, we investigated the on-line dynamics with a weight-decay term for the Hebbian learning. Using this method, we could eliminate the over-training, and the generalization error converges to the optimal value exponentially. However, our procedure depends on information about the unknown parameter a.

We then introduced a new learning rate independent of the unknown parameter a. We assumed $g(\alpha) = kl/\alpha$ and optimized k so that the generalization error decays to the minimum value as quickly as possible. As a result, for the noiseless unrealizable case of $a > a_{c1}$ the prefactor was somewhat improved although the exponent of decay did not change. Using the same technique, we investigated the noisy case and found that if the student does not have the information about the noise level λ, the prefactor of the generalization error becomes larger than that of the student who knows the teacher noise level.

We also investigated the Hebbian learning with queries. If the student is trained by the Hebbian algorithm using inputs on the decision boundary, his generalization ability is improved except in the range $a_{c2} < a < a_{c1}$. This is a highly non-trivial result because this choice of query works well for the unrealizable case where the student does not know the structure of the teacher. We next introduced the optimal learning rate in the on-line Hebbian learning with queries and obtained very fast convergence of generalization error. For $a > a_{c1}$, the generalization error converges to its optimal value exponentially.

We have observed exponential decays to limiting values in various situations of unrealizable rules. This

fast convergence may originate in the large size of the asymptotic space; if the liming value of R is unity, only a single point in the J-space, $J = B$, is the correct destination of learning dynamics, a very difficult task. If, on the other hand, R approaches $R_0(< 1)$, there are a continuous number of allowed student vectors, and to find one of these should be a relatively easy process, leading to exponential convergence.

æ

Chapter 5

Learning Processes in Non-Monotonic Perceptrons

In this chapter we study the on-line learning process and the generalization ability of this non-monotonic perceptron, namely, a student as a non-monotonic perceptron learns from a teacher as a non-monotonic perceptron, by various learning algorithms.

This chapter is organized as follows. In the next section we introduce our model system and derive the dynamical equations with respect to two order parameters for a general learning algorithm. One is the overlap between the teacher and student weight vectors and the other is the length of the student weight vector. In section 1, we introduce our model system and explain the generic properties of the generalization error. In section 2, we investigate the dynamics of on-line learning in the non-monotonic perceptron for the conventional perceptron learning and Hebbian leaning algorithms. We also investigate the asymptotic form of the differential equations in both small and large α limits and get the asymptotic behavior of the generalization error. In section 3, we investigate the AdaTron learning algorithm and modify the conventional AdaTron algorithm. In this modification procedure, we improve the weight function of the AdaTron learning so as to adopt it to the situation according to the range of a. In section 4, we optimize the learning rate and the general weight function appearing in the on-line dynamics. As the weight function contains the variables unknown for the student, we average these variables over the distribution function, which contains unknown variables for the student, using the Bayes formula. Section 5 contains concluding remarks.

5.1 The model system and dynamical equations

We investigate the generalization ability of the non-monotonic perceptrons for various learning algorithms. The student and teacher perceptron are characterized by their weight vectors, namely $\boldsymbol{J} \in \Re^N$ and $\boldsymbol{B} \in \Re^N$ with $|\boldsymbol{B}| = 1$, respectively. For a binary input signal $\boldsymbol{x} \in \{-1, +1\}^N$, the output is calculated by the non-monotonic transfer function as follows:

$$T_a(v) = \mathrm{sign}\left[v(a-v)(a+v)\right] \tag{5.1}$$

for the teacher and

$$S_a(u) = \mathrm{sign}\left[u(a-u)(a+u)\right] \tag{5.2}$$

for the student, where we define the local fields of the teacher and student as $v \equiv \sqrt{N}(\boldsymbol{B}\cdot\boldsymbol{x})/|\boldsymbol{B}|$ and $u \equiv \sqrt{N}(\boldsymbol{J}\cdot\boldsymbol{x})/|\boldsymbol{J}|$, respectively. The on-line learning dynamics is defined by the following general rule for the change of the student vector under presentation of the mth example;

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + f(T_a(v), u)\boldsymbol{x}. \tag{5.3}$$

Well-known examples are the perceptron learning, $f = -S_a(u)\,\Theta(-T_a(v)S_a(u))$, the Hebbian learning, $f = T_a(v)$, and the AdaTron learning, $f = -u\,\Theta(-T_a(v)S_a(u))$.

We rewrite the update rule, equation (5.3), of $\boldsymbol{J}$ as a set of differential equations introducing the dynamical order parameter describing the overlap between the teacher and student weight vectors $R^m \equiv (\boldsymbol{B}\cdot\boldsymbol{J}^m)/|\boldsymbol{J}$ and another order parameter describing the norm of the student weight vector $l^m \equiv |\boldsymbol{J}^m|/\sqrt{N}$. By taking the overlap of both sides of equation (5.3) with $\boldsymbol{B}$ and by squaring both sides of the same equation, we obtain the dynamical equations in the limit of large m and N keeping $\alpha \equiv m/N$ finite as

$$\frac{dl}{d\alpha} = \frac{1}{2l}\ll f^2(T_a(v),u) + 2f(T_a(v),u)\,ul\gg \tag{5.4}$$

and

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\ll -\frac{R}{2}f^2(T_a(v),u) - (Ru - v)f(T_a(v),u)l\gg. \tag{5.5}$$

Here $\ll\cdots\gg$ denotes the average over the randomness of inputs

$$\ll\cdots\gg \equiv \int\int dudv(\cdots)P_R(u,v) \tag{5.6}$$

with

$$P_R(u,v) \equiv \frac{1}{2\pi\sqrt{1-R^2}}\exp\left[-\frac{(u^2+v^2-2Ruv)}{2(1-R^2)}\right]. \tag{5.7}$$

As we are interested in the typical behavior under our training algorithm, we have averaged both sides of equations (5.4) and (5.5) over all possible instances of examples. The Gaussian distribution (5.7) has been derived from the central limit theorem.

The generalization error, which is the probability of disagreement between the teacher and the trained student, is represented as $\epsilon_g = \ll\Theta(-T_a(v)S_a(u))\gg$. After simple calculations, we obtain the generalization error as

$$\begin{aligned} E(R)\equiv\epsilon_g = {} & 2\int_a^\infty Dv\,H\left(\frac{a+Rv}{\sqrt{1-R^2}}\right) + 2\int_a^\infty Dv\,H\left(\frac{-(a-Rv)}{\sqrt{1-R^2}}\right) \\ & + 2\int_0^a Dv\,H\left(\frac{Rv}{\sqrt{1-R^2}}\right) - 2\int_a^\infty Dv\,H\left(\frac{Rv}{\sqrt{1-R^2}}\right) \\ & - 2\int_0^a Dv\,H\left(\frac{a+Rv}{\sqrt{1-R^2}}\right) + 2\int_0^a Dv\,H\left(\frac{a-Rv}{\sqrt{1-R^2}}\right) \end{aligned} \tag{5.8}$$

where we have set $H(x) = \int_x^\infty Dt$ with $Dt \equiv dt\exp(-t^2/2)/\sqrt{2\pi}$.

We would like to emphasize that the generalization error obtained above (5.8) is independent of the specific learning algorithm. In Figure 5.1, we plot $E(R) = \epsilon_g$ for several values of a. This figure tells us that the student can acquire a perfect generalization ability if he is trained so that R converges to 1 for all values of a. We have confirmed also analytically that $E(R)$ is a monotonically decreasing function of R for any value of a.

5.2 Hebbian and Perceptron learning algorithms

5.2.1 Hebbian learning

We first investigate the performance of the on-line Hebbian learning $f = T_a(v)$. We get the differential equations for l and R as follows

$$\frac{dl}{d\alpha} = \left[\frac{1}{2} + \frac{2R}{\sqrt{2\pi}}(1-2\Delta)l\right]/l \tag{5.9}$$

$$\frac{dR}{d\alpha} = \left[-\frac{R}{2}\frac{2}{\sqrt{2\pi}}(1-2\Delta)(1-R^2)l\right]/l^2. \tag{5.10}$$

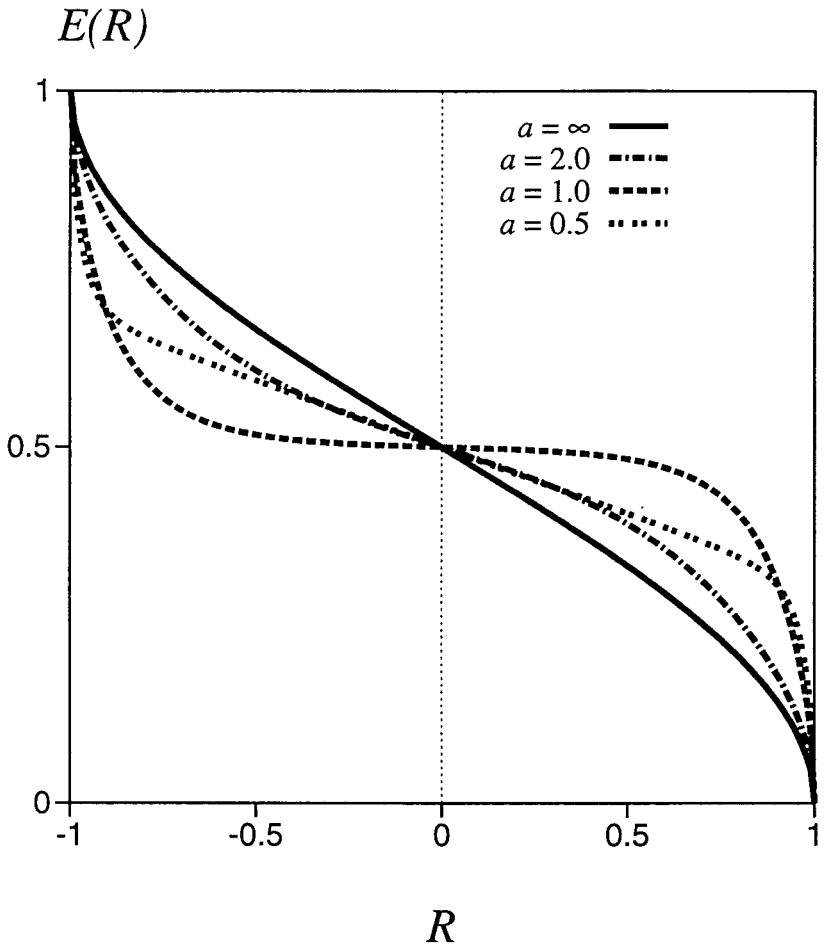

Figure 5.1: Generalization error as a function of the overlap R for the several value of a. The student should be trained so that the overlap goes to 1.

To determine whether or not R increases with α according to a, we approximate the differential equation for R around $R = 0$ as

$$\frac{dR}{d\alpha} = \frac{2}{\sqrt{2\pi}}(1-2\Delta)\frac{1}{l^2}. \tag{5.11}$$

Therefore we use $R = 1-\varepsilon$ for $a > a_c \equiv \sqrt{2\log 2}$ and $R = \varepsilon - 1$ for $a < a_c$. When $a > a_c$, we obtain

$$\epsilon_g = \frac{1}{\sqrt{2\pi}}\frac{1+2\Delta}{1-2\Delta}\frac{1}{\sqrt{\alpha}} \tag{5.12}$$

and

$$l = \sqrt{\frac{2}{\pi}}(1-2\Delta)\alpha. \tag{5.13}$$

On the other hand, for $a < a_c$ we obtain

$$\epsilon_g = 1 + \frac{1}{\sqrt{2\pi}}\frac{1+2\Delta}{1-2\Delta}\frac{1}{\sqrt{\alpha}} \tag{5.14}$$

and

$$l = -\sqrt{\frac{2}{\pi}}(1-2\Delta)\alpha. \tag{5.15}$$

We see that the Hebbian learning algorithms lead to the state $R = -1$ for $a < a_c$.

5.2.2 Perceptron learning

We next investigate the on-line perceptron learning $f = -S_a(u)\,\Theta(-T_a(v)S_a(u))$ by solving the next differential equations numerically;

$$\frac{dl}{d\alpha} = [\frac{1}{2}E(R) - F(R)\,l]/l \tag{5.16}$$

$$\frac{dR}{d\alpha} = [-\frac{1}{2}E(R)R + (F(R)R - G(R))l]/l^2 \tag{5.17}$$

where $F(R) = \ll\Theta(-T_a(v)S_a(u))S_a(u)u\gg$ and $G(R) = \ll\Theta(-T_a(v)S_a(u))S_a(u)v\gg$. Using the distribution (5.7) we can rewrite these functions as

$$F(R) = \frac{(1-R)}{\sqrt{2\pi}}(1-2\Delta) \tag{5.18}$$

$$G(R) = -F(R) \tag{5.19}$$

where $\Delta \equiv \exp(-a^2/2)$. In Figure 5.2 we plot the change of R and l as learning proceeds under various initial conditions for the case of $a = \infty$. We see that the student can reach the perfect generalization state $R = 1$ for any initial condition. The R-l flow in the opposite limit $a = 0$ is shown in Figure 5.3. Apparently, for this case the student reaches the state with the weight vector opposite to the teacher, $R = -1$, after an infinite number of patterns are presented. From equations (5.1) and (5.2), we should notice that the case of $a = 0$ is essentially different from the case of a simple perceptron.

Since the two limiting cases, $a = \infty$ and $a = 0$, follow different types of behavior, it is necessary to check what happens in the intermediate region. For this purpose, we first investigate the asymptotic behavior of the solution of equations (5.16) and (5.17) near $R = \pm 1$ for large α. Using the notation $R = 1-\varepsilon$, $\varepsilon \to 0$, the asymptotic forms of $E(R)$, $F(R)$ and $G(R)$ are found to be

$$E(R) \simeq \frac{\sqrt{2\varepsilon}}{\pi}(1+2\Delta) \tag{5.20}$$

$$F(R) \simeq \frac{\varepsilon}{\sqrt{2\pi}}(1-2\Delta) \tag{5.21}$$

$$G(R) \simeq -\frac{\varepsilon}{\sqrt{2\pi}}(1-2\Delta). \tag{5.22}$$

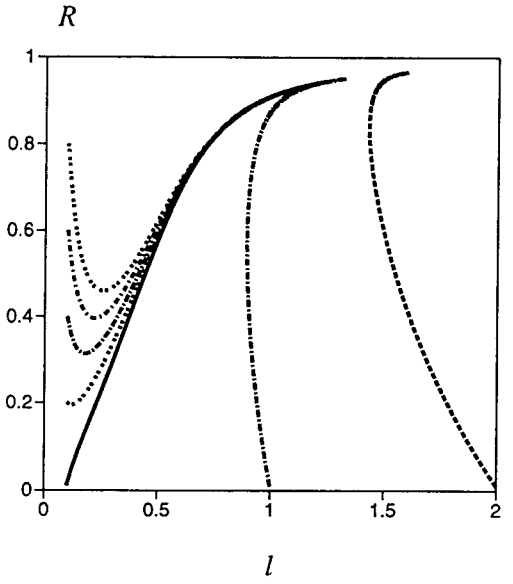

Figure 5.2: Trajectories of the R-l flows for $a = \infty$. All R-l flows converges to the state $R = 1$ after infinite number of examples are presented.

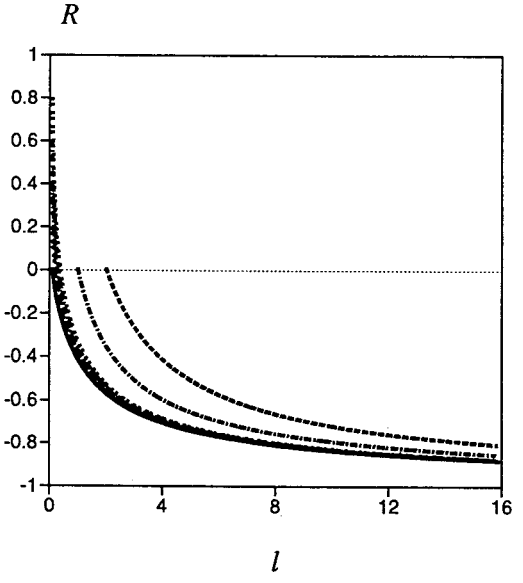

Figure 5.3: Trajectories for the R-l flow for $a = 0$. All R-l flows converges to the state $R = -1$. Therefore, the corresponding generalization error does not converges to the ideal value of zero for this case.

Substituting these expressions into the differential equations (5.16) and (5.17), we obtain

$$\varepsilon = \left[\frac{(1+2\Delta)}{3\sqrt{2}(1-2\Delta)^2}\right]^{2/3} \alpha^{-2/3} \tag{5.23}$$

and

$$l = \frac{1}{2\sqrt{\pi}}\left(\frac{1+2\Delta}{1-2\Delta}\right)\left[\frac{3\sqrt{2}(1-2\Delta)^2}{(1+2\Delta)}\right]^{1/3} \alpha^{1/3}. \tag{5.24}$$

Therefore, the generalization error is obtained from equation (5.20) as

$$\epsilon_g = (1+2\Delta)\frac{\sqrt{2}}{\pi}\left[\frac{(1+2\Delta)}{3\sqrt{2}(1-2\Delta)^2}\right]^{1/3} \alpha^{-1/3}. \tag{5.25}$$

The asymptotic form of l, equation (5.24), shows that Δ should satisfy $2\Delta < 1$ or $a > a_c$. The assumption of $R = 1 - \varepsilon$ with $\varepsilon \to 0$ thus fails if $a < a_c$. This fact can be verified from Eq. (5.17) expanded around $R = 0$ as

$$\frac{dR}{d\alpha} \simeq \frac{2}{\sqrt{2\pi}}(1-2\Delta)\frac{1}{l^2}. \tag{5.26}$$

For $a < a_c$, R decreases with α. Therefore, we use the relation $R = \varepsilon - 1, \varepsilon \to 0$, instead of $R = 1 - \varepsilon$ for $a < a_c$. We then find the asymptotic form of the generalization error as

$$\epsilon_g = 1 + \left[\frac{1+2\Delta}{1-2\Delta}\right]\frac{1}{\sqrt{2\pi\alpha}} \tag{5.27}$$

and l goes to infinity as

$$l = -\frac{2}{\sqrt{2\pi}}(1-2\Delta)\alpha. \tag{5.28}$$

These two results, equations (5.25) and (5.27), confirm the difference in the asymptotic behaviors between the two cases of $a = 0$ and $a = \infty$.

We have found that the Hebbian and the conventional perceptron learning algorithms lead to the state $R = -1$ for $a < a_c = \sqrt{2\log 2}$. This anti-learning effect may be understood as follows. If the student perceptron has learned only one example by the Hebb rule,

$$\boldsymbol{J} = T_{a=0}(v)\boldsymbol{x}. \tag{5.29}$$

Then the output of the student for the same example is

$$\begin{aligned} S_{a=0}(u) &= -\mathrm{sign}(u) \\ &= -\mathrm{sign}\,(\boldsymbol{J}\cdot\boldsymbol{x}) \\ &= -T_{a=0}(v). \end{aligned} \tag{5.30}$$

This relation indicates the anti-learning effect for the $a = 0$ case. Similar analysis holds for the perceptron learning.

5.2.3 Generalized perceptron learning

In this section, we introduce a multiplicative factor $|u|^\gamma$ in front of the perceptron learning function, $f = -|u|^\gamma \Theta(-T_a(v)S_a(u))S_a(u)$, and investigate how the generalization ability depends on the parameter γ. In particular, we are interested in whether or not an optimal value of γ exists. The learning dynamics is therefore

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m - |u|^\gamma S_a(u)\Theta(-T_a(v)S_a(u))\boldsymbol{x}. \tag{5.31}$$

The case of $\gamma = 0$ corresponds to the conventional perceptron learning algorithm. On the other hand, the case of $\gamma = 1$ and $a \to \infty$ corresponds to the conventional AdaTron learning. Using the above learning dynamics, we obtain the differential equations with respect to l and R as

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{E_G(R)}{2} - lF_G(R)\right] \tag{5.32}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2}E_G(R) + (F_G(R)R - G_G(R))l\right], \tag{5.33}$$

where $E_G(R)$, $F_G(R)$ and $G_G(R)$ are represented as

$$E_G(R) \equiv \ll u^{2\gamma}\Theta(-T_a(v)S_a(u))\gg, \tag{5.34}$$

$$F_G(R) \equiv \ll |u|^{\gamma+1}\Theta(-T_a(v)S_a(u))S_a(u)\gg \tag{5.35}$$

$$G_G(R) \equiv \ll |u|^{\gamma}\Theta(-T_a(v)S_a(u))S_a(u)v\gg. \tag{5.36}$$

Let us first investigate the behavior of the R-l flow near $R = 0$. When R is very small, the right-hand side of Eq. (5.33) is found to be a γ-dependent constant:

$$\frac{dR}{d\alpha} = \frac{2^{\frac{1}{2}(\gamma-1)}}{\pi l}\Gamma\left(\frac{\gamma}{2} + \frac{1}{2}\right)(1 - 2\Delta), \tag{5.37}$$

where $\Gamma(x)$ is the gamma function. As the right hand side of equation (5.37) is positive for any γ as long as a satisfies $a > a_c$, R increases around $R = 0$ only for this range of a. Thus the generalized perceptron learning algorithm succeeds in reaching the desired state $R = 1$, not the opposite one $R = -1$, only for $a > a_c$, similarly to the conventional perceptron learning. Therefore, in this section we restrict our analysis to the case of $a > a_c$ and investigate how the learning curve changes according to the value of γ.

Using the notation $R = 1 - \varepsilon$ ($\varepsilon \to 0$), we obtain the asymptotic forms of E_G, F_G and G_G as follows.

$$E_G \simeq c_1\varepsilon^{\gamma+\frac{1}{2}} + c_2\varepsilon^{\frac{1}{2}} \tag{5.38}$$

$$F_G \simeq c_3\varepsilon^{1+\frac{\gamma}{2}} - c_4\varepsilon \tag{5.39}$$

$$G_G \simeq -\frac{c_3}{\gamma+1}\varepsilon^{1+\frac{\gamma}{2}} + c_4\varepsilon \tag{5.40}$$

where $c_1 \equiv 2^{2\gamma+1/2}\Gamma(\gamma+1)/\pi(2\gamma+1)$, $c_2 \equiv 4a^{2\gamma}\Delta/\sqrt{2}\pi$, $c_3 \equiv 2^{\gamma+3/2}\Gamma(\frac{\gamma}{2}+\frac{3}{2})/\pi(\gamma+2)$ and $c_4 \equiv 2\Delta a^{\gamma}/\sqrt{2\pi}$. We first investigate the case of $\Delta \neq 0$ (finite a), namely, $c_2, c_4 \neq 0$. The differential equations (5.32) and (5.33) are rewritten in terms of ε and $\delta = 1/l$ as

$$\frac{d\delta}{d\alpha} = -\frac{\delta^3}{2}\left[c_1\varepsilon^{\gamma+\frac{1}{2}} + c_2\varepsilon^{\frac{1}{2}}\right] + \delta^2\left[c_3\varepsilon^{1+\frac{\gamma}{2}} - c_4\varepsilon\right] \tag{5.41}$$

$$\frac{d\varepsilon}{d\alpha} = \frac{\delta^2}{2}\left[c_1\varepsilon^{\gamma+\frac{1}{2}} + c_2\varepsilon^{\frac{1}{2}}\right] - \delta\left[\left(\frac{2+\gamma}{1+\gamma}\right)c_3\varepsilon^{1+\frac{\gamma}{2}} - 2c_4\varepsilon\right]. \tag{5.42}$$

As $\gamma = 0$ corresponds to the perceptron learning, we now assume $\gamma \neq 0$. When $\gamma > 0$, the terms containing c_1 and c_3 can be neglected in the leading order. Dividing equation (5.41) by equation (5.42), we obtain

$$\frac{d\delta}{d\varepsilon} = \frac{\delta\left[-c_2\delta\varepsilon^{1/2}/2 - c_4\varepsilon\right]}{\left[c_2\delta\varepsilon^{1/2}/2 + 2c_4\varepsilon\right]}. \tag{5.43}$$

If we assume $\delta\varepsilon^{1/2} \gg \varepsilon$ or $\delta\varepsilon^{1/2} \ll \varepsilon$, equation (5.43) is solved as $\delta = \exp(-\varepsilon)$, which is in contradiction to the assumption $|\delta| \ll 1$. Thus, we set

$$\delta = -\frac{4c_4}{c_2}\varepsilon^{1/2} + b\varepsilon^{c} \tag{5.44}$$

and determine b and $c(> 1/2)$. Substituting (5.44) into (5.43), we find $b = 8c_4/c_2$ ($c_2, c_4 > 0$)and $c = 3/2$. The negative value of $\delta = 1/l$ is not acceptable and we conclude that R does not approach 1 when $\gamma > 0$.

Next we investigate the case of $\gamma < 0$. Using the same technique as in the case of $\gamma > 0$, we obtain

$$\varepsilon = \left[\frac{c_1(1+\gamma)(1-\gamma^2)}{6c_3^2(\gamma+2)}\right]^{\frac{2}{3}} \alpha^{-\frac{2}{3}}, \tag{5.45}$$

$$\delta = \frac{2c_3}{c_1}\left(\frac{\gamma+2}{\gamma+1}\right)\varepsilon^{\frac{1}{2}(1-\gamma)} - \frac{4c_3}{c_1(1-\gamma^2)}\varepsilon^{\frac{1}{2}(3-\gamma)} \tag{5.46}$$

and

$$\begin{aligned}\epsilon_g &= \frac{\sqrt{2}}{\pi}(1+2\Delta)\left[\frac{c_1(1+\gamma)(1-\gamma^2)}{6c_3^2(\gamma+2)}\right]^{\frac{1}{3}}\alpha^{-\frac{1}{3}} \\ &\equiv \frac{\sqrt{2}}{\pi}(1+2\Delta)f(\gamma)\alpha^{-\frac{1}{3}}.\end{aligned} \tag{5.47}$$

We notice that γ should satisfy $-1 < \gamma < 0$, because the prefactor of the leading term of δ, namely, $(2c_3/c_1)(\gamma+2)/(\gamma+1)$, must be positive. As the prefactor of the generalization error increases monotonically from $\gamma = -1$ to $\gamma = 0$, we obtain a smaller generalization error for γ closer to -1.

Next we investigate the case of $a \to \infty$, namely $c_2, c_4 = 0$. We first assume $l \to l_0$ in the limit of $\alpha \to \infty$. In this solution, $dl/d\alpha = 0$ should be satisfied asymptotically. Then, from equation (5.41), the two terms $\varepsilon^{\gamma+\frac{1}{2}}$ and $\varepsilon^{1+\frac{\gamma}{2}}$ should be equal to each other, namely, $\varepsilon^{\gamma+\frac{1}{2}} = \varepsilon^{1+\frac{\gamma}{2}}$, which leads to $\gamma = 1$. The learning dynamics (5.31) with $a \to \infty$ and $\gamma = 1$ is nothing but the AdaTron learning which has already been investigated in detail [99]. The result for the generalization error is

$$\epsilon_g = \frac{3}{2\alpha}, \tag{5.48}$$

if we choose l_0 as $l_0 = 1/2$, and

$$\epsilon_g = \frac{4}{3\alpha}. \tag{5.49}$$

if we optimize l_0 to minimize the generalization error.

We next assume $l \to \infty$ as $\alpha \to \infty$. It is straightforward to see that ε has the same asymptotic form as in the case of $\Delta \neq 0$ and $\gamma < 0$. Thus we have

$$\epsilon_g = \frac{\sqrt{2}}{\pi} f_2(\gamma)\alpha^{-\frac{1}{3}}, \tag{5.50}$$

where $f_2(\gamma)$ is defined as

$$f_2(\gamma) = \left[\frac{\pi(1+\gamma)(1-\gamma^2)\Gamma(\gamma+1)}{6\cdot 2^{5/2}\Gamma^2(\frac{\gamma}{2}+\frac{1}{2})}\right]^{\frac{1}{3}} \tag{5.51}$$

and γ can take any value within $-1 < \gamma < 0$.

From the above analysis, we conclude that the student can get the generalization ability α^{-1} if and only if $a \to \infty$ and $\gamma = 1$ (AdaTron). For other cases the generalization error behaves as $\alpha^{-1/3}$, the same functional form as in the case of the conventional perceptron learning, as long as the student can obtain a vanishing residual error. Therefore the learning curve has universality in the sense that it does not depend on the detailed value of the parameter γ.

5.3 AdaTron learning algorithm

5.3.1 AdaTron learning

In this subsection, we investigate the generalization performance of the conventional AdaTron learning $f = -u\Theta(-T_a(v)S_a(u))$ [98]. The differential equations for l and R are given as follows:

$$\frac{dl}{d\alpha} = \left(\frac{1-2l}{2l}\right)E_{\rm Ad}(R) \tag{5.52}$$

$$\frac{dR}{d\alpha} = -\frac{1}{l}\left[\left(\frac{1-2l}{2l}\right)E_{\rm Ad}(R)R + G_{\rm Ad}(R)\right] \tag{5.53}$$

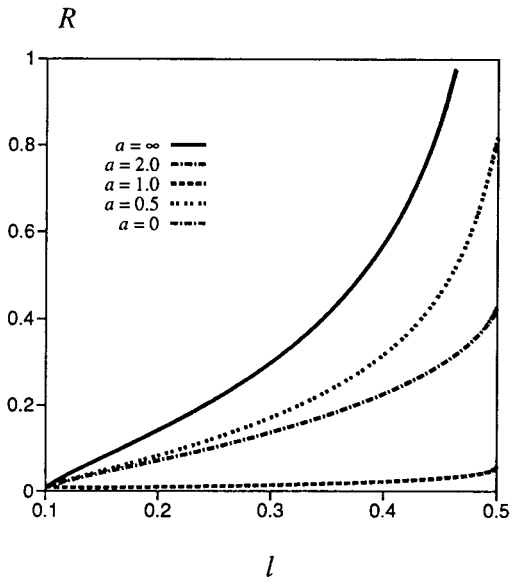

Figure 5.4: Trajectories for the conventional AdaTron learning. Except for the case $a = \infty$, the trajectories converge to the state $l = 1/2$.

where $E_{\rm Ad}(R) = \ll u^2\Theta(-T_a(v)S_a(u))\gg$ and $G_{\rm Ad}(R) = \ll uv\Theta(-T_a(v)S_a(u))\gg$. After simple calculations, we obtain

$$\begin{aligned} E_{\rm Ad}(R) &= 2\left(\int_a^\infty + \int_{-a}^0\right) Du\, u^2\, H\left(\frac{a+Ru}{\sqrt{1-R^2}}\right) \\ &+ 2\left(\int_0^a + \int_{-\infty}^{-a}\right) Du\, u^2 \left[H\left(\frac{Ru}{\sqrt{1-R^2}}\right) - H\left(\frac{a+Ru}{\sqrt{1-R^2}}\right)\right] \end{aligned} \tag{5.54}$$

and

$$\begin{aligned} G_{\rm Ad}(R) &= E_{\rm Ad}(R)R \\ &+ \frac{4Ra\Delta}{\sqrt{2\pi}}(1-R^2)\left[H\left(\frac{a(1+R)}{\sqrt{1-R^2}}\right) - H\left(\frac{aR}{\sqrt{1-R^2}}\right) - H\left(\frac{a(1-R)}{\sqrt{1-R^2}}\right) + \frac{1}{2}\right] \\ &+ \frac{2(1-R^2)^{\frac{3}{2}}}{\pi}\left[\Delta\exp\left[-\frac{a^2R^2}{2(1-R^2)}\right] - \Delta\exp\left[-\frac{a^2(1+R)^2}{2(1-R^2)}\right] - \Delta\exp\left[-\frac{a^2(1-R)^2}{2(1-R^2)}\right] - \frac{1}{2}\right] \end{aligned} \tag{5.55}$$

At first, we check the behavior of R around $R = 0$. Evaluating the differential equation (5.53) around $R = 0$, we obtain

$$\frac{dR}{d\alpha} = \frac{4}{l\pi}\left(\Delta - \frac{1}{2}\right)^2 \tag{5.56}$$

From this result we find that for any value of a, the flow of R increases around $R = 0$. In Figure 5.4, we display the flows in the R-l plane for several values of a by numerical integration of equation (5.53). This figure indicates that the overlap R increases monotonically, but R does not reach the state $R = 1$ if a is finite. This means that the differential equation (5.53) with respect to R has a non-trivial fixed point $R = R_0(< 1)$ if $a < \infty$, which is the solution of the non-linear equation $G_{\rm Ad}(R) = 0$ since $l = 1/2$ is clearly the fixed point from equation (5.52) and Figure 5.4. Therefore, we conclude that for $a = \infty$ and $a = 0$, we obtain the generalization error as $\epsilon_g \sim \alpha^{-1}$, but the generalization error converges to a finite value exponentially for finite a.

5.3.2 Modified AdaTron learning

In the previous subsection, we found that the on-line AdaTron learning fails to obtain the zero residual error for finite a. In this subsection, we modify the AdaTron learning as $f = \Theta(-T_a(v)S_a(u))h(u)l$ with

$$h(u) = \begin{cases} a-u & (u > \frac{a}{2}) \\ -u & (-\frac{a}{2} < u < \frac{a}{2}) \\ -a-u & (u < -\frac{a}{2}) \end{cases} \tag{5.57}$$

and see if the generalization ability of our non-monotonic system is improved. The motivation for the above choice comes from the optimization of the learning algorithm to be mentioned in the next section. Details of derivation of equation (5.57) are found in Appendix B. Then the differential equation with respect to R is obtained as follows.

$$\frac{dR}{d\alpha} = -\frac{R^2}{2}E_{\rm MA}(R) - RF_{\rm MA}(R) + G_{\rm MA}(R) \tag{5.58}$$

where $E_{\rm MA}(R) = \ll h^2(u)\Theta(-T_a(v)S_a(u))\gg$, $F_{\rm MA}(R) = \ll uh(u)\Theta(-T_a(v)S_a(u))\gg$ and $G_{\rm MA}(R) = \ll vh(u)\Theta(-T_a(v)S_a(u))\gg$. To see the asymptotic behavior of the generalization error, we evaluate the leading-order contribution as R approaches 1, $R = 1-\varepsilon$, as

$$E_{\rm MA} \sim \frac{2\sqrt{2}}{\pi}(1+2\Delta)\varepsilon^{\frac{3}{2}} \tag{5.59}$$

$$F_{\rm MA} \sim -\frac{2\sqrt{2}}{\pi}\left(1+2(1-a^2)\Delta\right)\varepsilon^{\frac{3}{2}} \tag{5.60}$$

$$G_{\rm MA} \sim \frac{4\sqrt{2}a^2\Delta}{\pi}\varepsilon^{\frac{3}{2}}. \tag{5.61}$$

Substituting these expressions into the differential equation (5.58), we obtain $\varepsilon^{1/2} = \sqrt{2}\pi/(1+2\Delta)\alpha^{-1}$ and the generalization error as

$$\epsilon_g = \frac{\sqrt{2}(1+2\Delta)}{\pi}\varepsilon^{\frac{1}{2}} = \frac{2}{\alpha}. \tag{5.62}$$

We should notice that the above result is independent of a and the generalization ability of the student is improved by this modification for all finite a.

5.4 Optimized learning

5.4.1 Optimization of the learning rate

In the present subsection, we improve the conventional perceptron learning by introducing a time-dependent learning late [90, 99]. We consider the next on-line dynamics;

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m - g(\alpha)\,\Theta(-T_a(v)S_a(u))S_a(u)\boldsymbol{x}. \tag{5.63}$$

Using the same technique as in the previous section, we can derive the differential equations with respect to l and R as follows.

$$\frac{dl}{d\alpha} = \frac{1}{l}\left[\frac{1}{2}g(\alpha)^2E(R) - g(\alpha)F(R)l\right] \tag{5.64}$$

$$\frac{dR}{d\alpha} = \frac{1}{l^2}\left[-\frac{R}{2}E(R)g(\alpha)^2 + g(\alpha)(F(R)R - G(R))l\right] \equiv L(g(\alpha)). \tag{5.65}$$

The optimal learning rate $g_{\rm opt}(\alpha)$ is determined so as to maximize $L(g(\alpha))$ to accelerate the increase of R. We then find

$$g_{\rm opt} = \frac{[F(R)R - G(R)]l}{RE(R)}. \tag{5.66}$$

Substituting this expression into the above differential equations, we obtain

$$\frac{dl}{dR} = -\frac{[F(R)R - G(R)][F(R)R + G(R)]l}{2R^2E(R)} \tag{5.67}$$

$$\frac{dR}{dl} = \frac{[F(R)R - G(R)]^2}{2RE(R)}. \tag{5.68}$$

We can obtain the asymptotic form of $\varepsilon\,(=1-R)$, l and ϵ_g with the same technique of analysis as in the previous section;

$$\varepsilon = 4\left[\frac{2\sqrt{2}(1+2\Delta)}{(1-2\Delta)^2}\right]^2 \alpha^{-2}, \tag{5.69}$$

$$l = \exp\left[-16\left(\frac{1+2\Delta}{(1-2\Delta)^2}\right)^4 \alpha^{-4}\right], \tag{5.70}$$

and

$$\epsilon_g = \frac{\sqrt{2}}{\pi}(1+2\Delta)\left[\frac{2\sqrt{2}(1+2\Delta)}{(1-2\Delta)^2}\right]\alpha^{-1}. \tag{5.71}$$

Therefore, the generalization ability has been improved from $\alpha^{-1/3}$ for $g = 1$ to α^{-1}. The optimal learning rate $g_{\rm opt}(\alpha)$ behaves asymptotically as

$$g_{\rm opt} = \frac{2\sqrt{2\pi}}{(1-2\Delta)}\alpha^{-1}\exp\left(-16\left(\frac{1+2\Delta}{(1-2\Delta)^2}\right)^4 \alpha^{-4}\right). \tag{5.72}$$

The factor $F(R)R - G(R)$ of $g_{\rm opt}$ appearing in equation (5.66) is calculated by substituting $F(R)$ and $G(R)$ in Eqs. (5.18) and (5.19) as $F(R)R - G(R) = (1-R^2)(1-2\Delta)/\sqrt{2\pi}$. Thus, at $a = a_c = \sqrt{2\log 2}$, the optimal learning rate vanishes. Therefore our formulation does not work at $a = a_c$.

As the optimal learning rate $g_{\rm opt}$ changes the sign at $a = a_c$, from the arguments in subsection 5.2, we can see why the optimal learning rate can eliminate the anti-learning.

In relation to this phenomenon at $a = \sqrt{2\log 2}$, Van den Broeck [100, 101] recently investigated the same reversed-wedge perceptron which learns in the unsupervised mode from the distribution

$$P(v) = 2\frac{\exp(-\frac{v^2}{2})}{\sqrt{2\pi}}\left[\Theta(v-a) + \Theta(v+a)\Theta(-v)\right] \tag{5.73}$$

with $v = \sqrt{N}(\boldsymbol{B}\cdot\boldsymbol{x})/|\boldsymbol{B}|$. For small α, he found $R(\alpha) \sim \sqrt{\alpha < v >^2}$ for the optimal on-line learning, where $< \cdots >$ denotes the average over the distribution (5.73). Then he showed that at $a = \sqrt{2\log 2}$, the distribution (5.73) leads to $< v >= 0$ and consequently $R(\alpha) \equiv 0$. From this result, he concluded that as long as $< v >= 0$ holds, any kind of on-line learning necessarily fails and the corresponding learning curve has a plateau. It seems that a similar mechanism may lead to a failure of the optimal learning at $a = \sqrt{2\log 2}$ in our model.

5.4.2 Optimization of the weight function using the Bayes formula

In this subsection we try another optimization procedure by Kinouchi and Caticha [92]. We choose the optimal weight function $f(T_a(v), u)$ by differentiating the right hand side of equation (5.5) with the aim to accelerate the increase of R

$$f^* = \frac{l}{R}(v - Ru). \tag{5.74}$$

It is important to remember that f^* contains some unknown information for the student, namely, the local field of the teacher v. Therefore, we should average f^* over a suitable distribution to erase v from f^*. For this purpose, we transform the variables u and v to u and z

$$v = z\sqrt{1-R^2} + Ru. \tag{5.75}$$

Then, the connected Gaussian distribution $P_R(u,v)$ is rewritten as

$$P_R(u,v) = \frac{1}{2\pi\sqrt{1-R^2}}\exp(-\frac{u^2}{2})\exp(-\frac{z^2}{2}). \tag{5.76}$$

We then obtain

$$< f^* >= \frac{\sqrt{1-R^2}}{R} l < z > \tag{5.77}$$

where $< \cdots >$ stands for the averaging over the variable v. Substituting this into the differential equation (5.5), we find

$$\frac{dR}{d\alpha} = \frac{(1-R^2)}{2R} \ll < z >^2 \gg. \tag{5.78}$$

Let us now calculate $< z >$. For this purpose, we use the distribution $P(z|y,u)$. This quantity means the posterior probability of z when y and u are given, where we have set $y \equiv T_a(v)$. This conditional probability is rewritten by the Bayes formula

$$P(z|y,u) = \frac{P(z)P(y|u,z)}{\int dz\, P(z)P(y|u,z)}, \tag{5.79}$$

from which we can calculate $< z >$ as

$$\begin{aligned} < z > &= \int dz\, z\, P(z|y,u) \\ &= \frac{\int dz\, z\, P(z)\, P(y|u,z)}{\int dz\, P(z)\, P(y|u,z)} \\ &= \frac{\int z\, Dz\, P(y|u,z)}{\int Dz\, P(y|u,z)}. \end{aligned} \tag{5.80}$$

Here $P(y|u,z)$ is given as

$$\begin{aligned} P(y|u,z) &= y\,\Theta(z\sqrt{1-R^2}+Ru) \\ &- y\,\Theta(z\sqrt{1-R^2}+Ru-a) \\ &+ y\,\Theta(-z\sqrt{1-R^2}-Ru-a) \\ &+ \frac{1}{2}(1-y) \end{aligned} \tag{5.81}$$

from the distribution $y = T_a(v)$. Then, the denominator of equation (5.79) is calculated as

$$\begin{aligned} \int Dz\, P(y|u,z) &= y\int Dz\,\Theta(z\sqrt{1-R^2}+Ru) \\ &- y\int Dz\,\Theta(z\sqrt{1-R^2}+Ru-a) \\ &+ y\int Dz\,\Theta(-z\sqrt{1-R^2}-Ru-a) + \frac{1}{2}(1-y) \\ &\equiv \Omega(y|u), \end{aligned} \tag{5.82}$$

where $\Omega(y|u)$ means the posterior probability of y when the local field of the student u is given. As we treat the binary output teacher, we obtain from equation (5.82)

$$\Omega(\pm 1|u) = H(\mp\frac{Ru}{\sqrt{1-R^2}}) \mp H(\frac{a-Ru}{\sqrt{1-R^2}}) \pm H(\frac{a+Ru}{\sqrt{1-R^2}}). \tag{5.83}$$

In Figures 5.5 ($R = 0.5$) and 5.6 ($R = 0.9$), we plot $\Omega(+1|u)$ for the cases of $a = 4.0, 2.0, 1.0$ and $a = 0.5$. From these figures, we find that for any a $\Omega(+1|u)$ seems to reach $(T_a(u)+1)/2$ as R goes to $+1$. Using

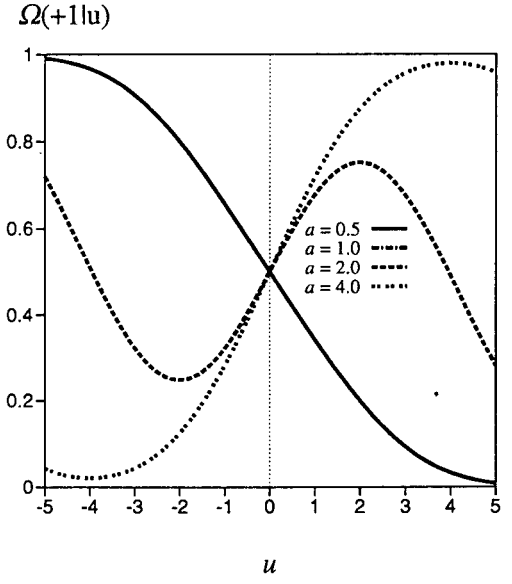

Figure 5.5: Shape of $\Omega(+1|u)$ for $R = 0.5$.

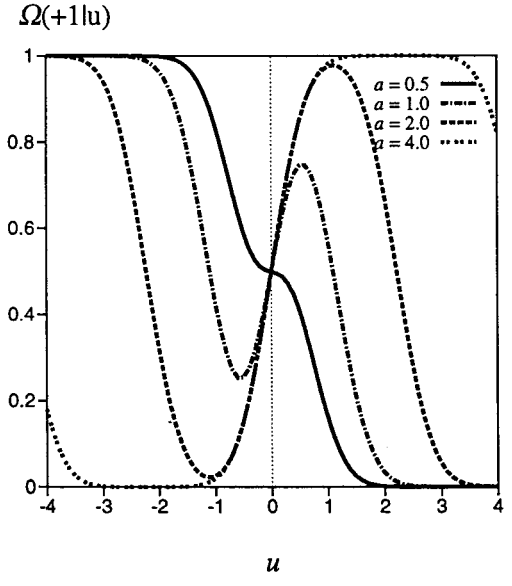

Figure 5.6: Shape of $\Omega(+1|u)$ for $R = 0.9$. Wee see that for any a, $\Omega(+1|u)$ seems to reach $(T_a(u) + 1)/2$ as R goes to $+1$.

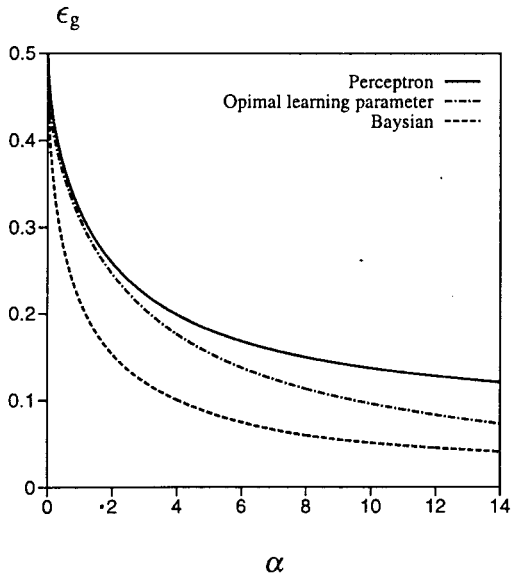

Figure 5.7: Learning curves of perceptron, optimized perceptron and Baysian optimization algorithms for $a = \infty$. The Baysian optimization algorithm is the best among three.

the same technique, we can calculate $\int Dz\, z\, P(y|u,z)$ and obtain

$$\int Dz\, z\, P(y|u,z) = \frac{\sqrt{1-R^2}}{R}\frac{\partial}{\partial u}\Omega(y|u). \tag{5.84}$$

Substituting this into the right hand side of $dR/d\alpha$, equation (5.78), we obtain

$$\frac{dR}{d\alpha} = \ll -\frac{(1-R^2)^2}{2R^3}\left\{\frac{\partial}{\partial u}\log\Omega(y|u)\right\}^2 + \frac{(1-R^2)^{3/2}z}{R^2}\frac{\partial}{\partial u}\log\Omega(y|u)\gg, \tag{5.85}$$

where $\ll\cdots\gg$ stands for the averaging over the distribution $P(y,u) = \int Dz\, P(y|u,z)P(u)P(z)$. Performing this average, we finally obtain

$$\frac{dR}{d\alpha} = \frac{(1-R^2)}{4\pi R}\int_{-\infty}^{\infty} Du\, \Xi_a(R,u) \tag{5.86}$$

where

$$\begin{aligned}\Xi_a(R,a) &\equiv \left[\exp(-\frac{A_1^2}{2}) - \exp(-\frac{A_2^2}{2}) - \exp(-\frac{A_3^2}{2})\right]^2 \\ &\times \left[\frac{1}{H(-A_1) - H(A_2) + H(A_3)} + \frac{1}{H(A_1) + H(A_2) - H(A_3)}\right]\end{aligned} \tag{5.87}$$

and $A_1 \equiv Ru/\sqrt{1-R^2}$, $A_2 \equiv (a-Ru)/\sqrt{1-R^2}$, $A_3 \equiv (a+Ru)/\sqrt{1-R^2}$. We plot the generalization error by numerically solving equations (5.16), (5.17), (5.67), (5.68), and (5.86) for the cases of $a = \infty$ in Figure 5.7 and $a = 1.0$ in Figure 5.8. From these figures, we see that for the both cases of $a = \infty$ and $a < \infty$, the generalization error calculated by the Bayes formula converges more quickly to zero than by the optimal learning rate $g_{\rm opt}(\alpha)$.

Recently, Simmonetti and Caticha [102] introduced the on-line learning algorithm for the non-overlapping parity machine with a general number of nodes K. In their method, the weight vector of the student in each hidden unit is trained by the method in Ref. [92]. In order to average over the internal fields of teacher in the differential equation with respect to the specific hidden unit k of the student, they need the conditional probability which depends not only on the internal field of the unit k but also on the internal

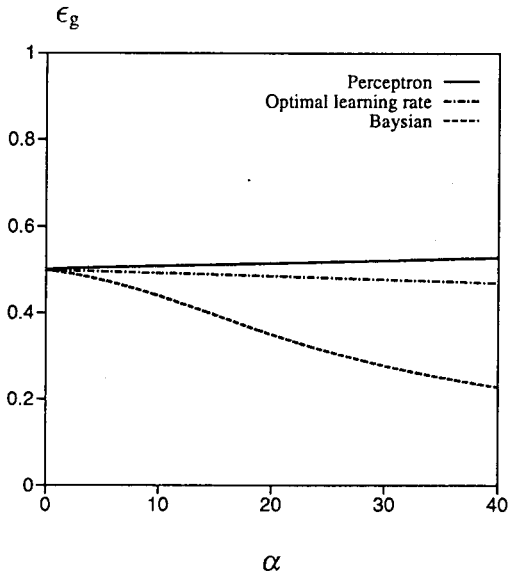

Figure 5.8: Learning curves of perceptron, optimized perceptron and Baysian optimization algorithms for $a = 1.0$. The Baysian optimization algorithm is the best among three.

field of the other units ($i \neq k$). This fact shows that their optimal algorithm is non-local. In our problem, the input-output relation of the machine can be mapped to those of a single layer reversed-wedge perceptron. Therefore, it is not necessary for us to use the information about all units and our optimizing procedure leads to a local algorithm.

In order to investigate the performance of the Bayes optimization, we have calculated the asymptotic form of the generalization error from equation (5.86) and the result is

$$\varepsilon^{\frac{1}{2}} = \frac{2}{(1+2\Delta)C\alpha} \tag{5.88}$$

for $\varepsilon = 1 - R$, where

$$C \equiv \frac{1}{\pi^{3/2}} \int_{-\infty}^{\infty} dt \frac{\exp(-t^2)}{H(t)}. \tag{5.89}$$

The generalization error is then given by equation (5.20) as

$$\epsilon_g = \frac{2\sqrt{2\pi}}{\int_{-\infty}^{\infty} dt \exp(-t^2)/H(t)} \frac{1}{\alpha} \sim 0.883 \frac{1}{\alpha}. \tag{5.90}$$

This asymptotic form of the generalization error agrees with the result of Kinouchi and Caticha [92]. We notice that this form is independent of the width of the reversed wedge a.

We next mention the physical meaning of $\Xi_a(R, u)$ appearing in the differential equation (5.86). As the rate of increase $dR/d\alpha$ is proportional to $\Xi_a(R, u)$, this quantity is regarded as the distribution of the gain which determines the increase of R. Therefore, $\Xi_a(R, u)$ yields important information about the strategy to make queries. A query means to restrict the input signal to the student, u, to some subspace. Kinzel and Ruján suggested that if the student learns by the Hebbian learning algorithm from restricted inputs, namely, inputs lying on the subspace $u = 0$, the prefactor of the generalization error becomes a half [96]. In the present formulation (5.86), a query-making can be incorporated by inserting appropriate delta functions in the integrand. The learning process is clearly accelerated by choosing the peak position of $\Xi_a(R, u)$ as the location of these delta functions. We plot the distribution $\Xi_a(R, u)$ for $a = 2.0$ in Figure 5.9, $a = 1.0$ in Figure 5.10 and $a = 0.8$ in Figure 5.11. From these figures, we learn that for large a ($= 2.0$), the most effective example lies on the decision boundary ($u = 0$) at the initial training stage (small R). However, as the student learns, two different peaks appear symmetrically and

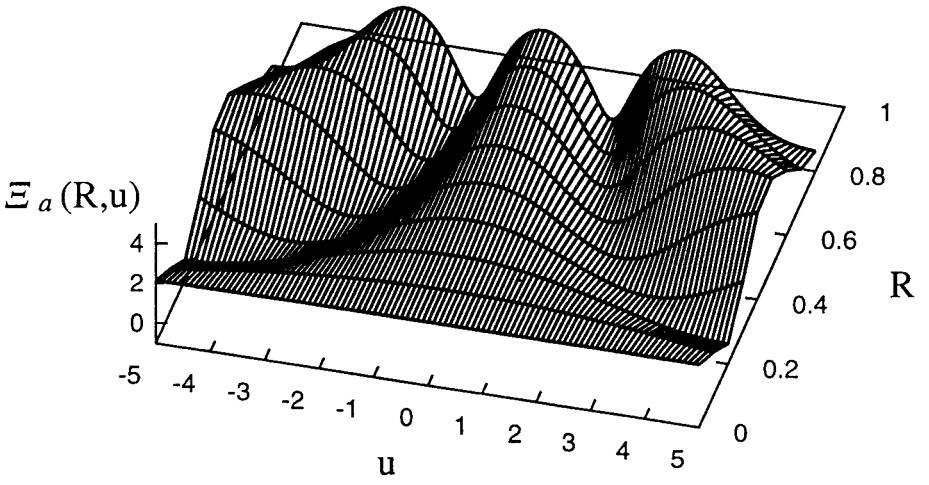

Figure 5.9: Distributions of the gain $\Xi_a(R,u)$ for $a = 2.0$.

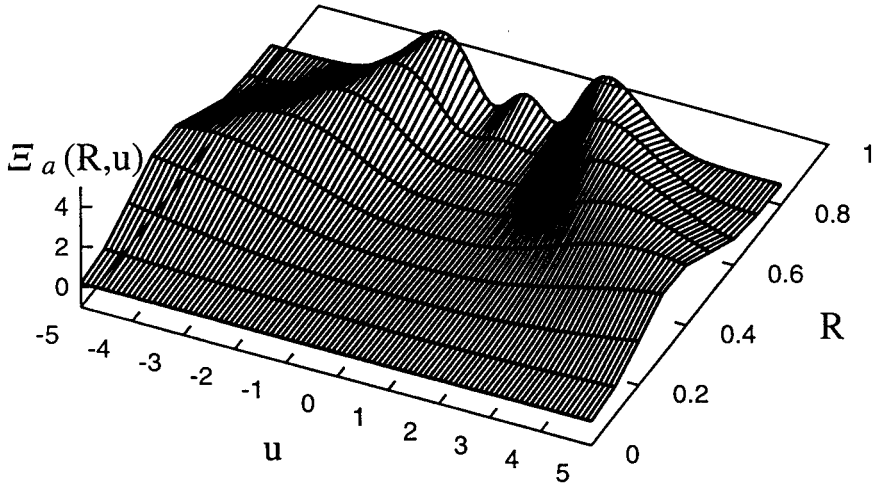

Figure 5.10: Distributions of the gain $\Xi_a(R,u)$ for $a = 1.0$.

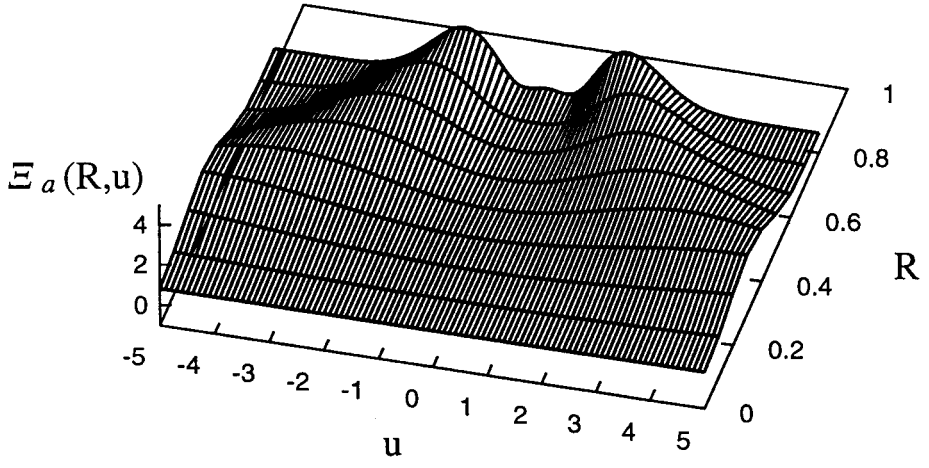

Figure 5.11: Distributions of the gain $\Xi_a(R,u)$ for $a = 0.8$.

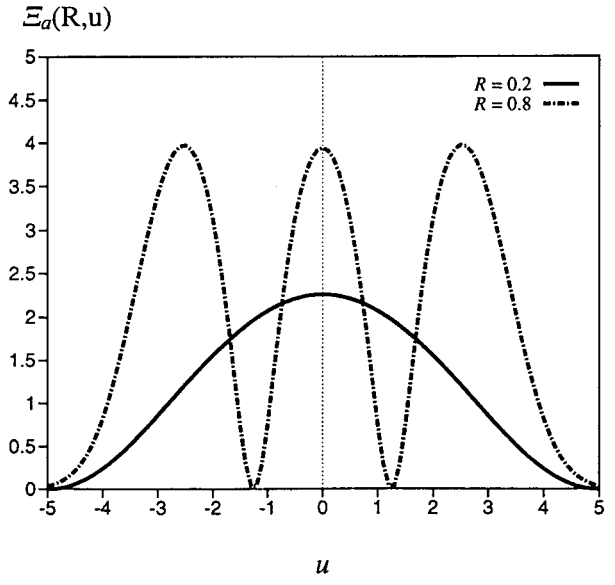

Figure 5.12: Cross-sections of $\Xi_a(R,u)$ at $R = 0.2$ and $R = 0.8$ for $a = 2.0$.

in the final stage of training, the distribution has three peaks around $u = 0$ and $u = \pm a$. On the other hand, for small a ($= 0.8$), the most effective examples lie at the tails ($u = \pm\infty$) for the initial stage. In the final stage, the distribution has two peaks around $u = \pm a$. Therefore it is desirable to change the location of queries adaptively. In Figures 5.12 and Figure 5.13, figure 5.14, we plot the cross-sections of Figures 5.9, Figure 5.10 and Figure 5.11 at $R = 0.2$ and $= 0.8$ respectively.

5.5 Summary

We have investigated the generalization abilities of a non-monotonic perceptron, which may also be regarded as a multilayer neural network, a parity machine, in the on-line mode. We first showed that the conventional perceptron and Hebbian learning algorithms lead to the perfect learning $R = 1$ only when $a > a_c = \sqrt{2\log 2}$. The same algorithms yield the opposite state $R = -1$ in the other case $a < a_c$. These algorithms have originally been designed having the simple perceptron ($a = \infty$) in mind, and thus are natural to give the opposite result for the reversed-output system ($a \sim 0$). In contrast, the conventional AdaTron learning algorithm failed to obtain the zero residual error for all finite values of a. For the unlearnable situation (where the structures of the teacher and student are different), Inoue and Nishimori reported that the AdaTron learning converges to the largest residual error among the three algorithms [99]. It is interesting that the AdaTron learning algorithm is not useful even for the learnable situation.

In order to overcome this difficulty, we introduced several modified versions of the conventional learning rules. We first introduced the time-dependent learning rate into the on-line perceptron learning and optimize it. As a result, the generalization error converges to zero in proportion to α^{-1} except at $a = \sqrt{2\log 2}$ where the learning rate becomes identically zero. We next improved the conventional AdaTron learning by modifying the weight function so that it changes according to the value of the internal potential u of the student. By this modification, the generalization ability of the student dramatically improved and the generalization error converges to zero with an a-independent form, $2\alpha^{-1}$.

We also investigated a different type of optimization: We first optimized the weight function $f(T_a(v), u)$ appearing in the on-line dynamics, not the rate g. Then, as the function f contains the unknown variable v, we averaged it over the distribution of v using the well-known technique of the Bayes statistics. This optimization procedure also provided other useful information for the student, namely, the distribution of most effective examples. Kinzel and Ruján [96] reported that for the situation in which a simple perceptron learns from a simple perceptron (the $a = \infty$ case), the Hebbian learning with selected ex-

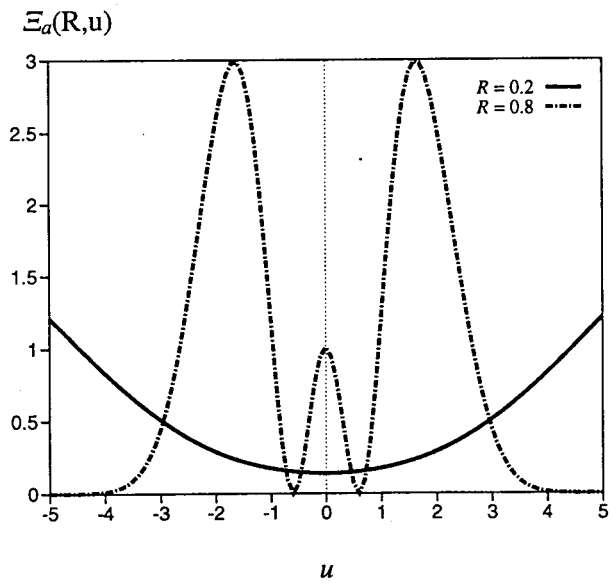

Figure 5.13: Cross-sections of $\Xi_a(R,u)$ at $R = 0.2$ and $R = 0.8$ for $a = 1.0$.

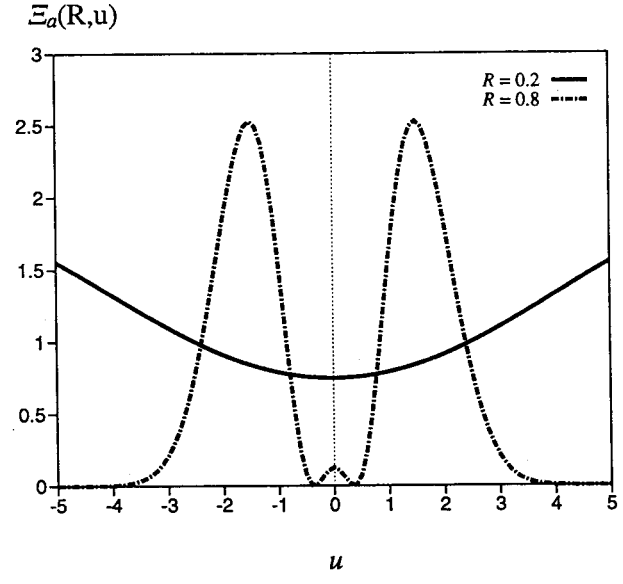

Figure 5.14: Cross-sections of $\Xi_a(R,u)$ at $R = 0.2$ and $R = 0.8$ for $a = 0.8$.

amples ($u = 0$) leads to faster convergence of the generalization error than the conventional Hebbian learning. However, we have found that for finite values of a, the most effective examples lie not only on the boundary $u = 0$ but also on $u = \pm a$. Furthermore, we could learn that for small values of a and at the initial stage of learning (R small), the most effective examples lie on the tails ($u = \pm\infty$). As the learning proceeds, the most effective examples change the locations to $u = \pm a$. This information is useful for effective query constructions adaptively at each stage of learning.

æ

Chapter 6

Optimization by Simulated Annealing

6.1 Annealing schedule

Simulated annealing has been a powerful tool for combinatorial optimization problems [9, 103, 104]. To find the minimum of a cost function, one introduces a stochastic process similar to Monte Carlo simulations in statistical mechanics with a control parameter corresponding to the temperature to allow the system to escape from local minima. By gradually decreasing the temperature one searches for increasingly narrower regions in the phase space closer to the optimal state, eventually reaching the optimal state itself in the infinite-time limit.

A very important factor in such processes is the annealing schedule, or the rate of decrease of temperature. If one lowers the temperature too quickly, the system may end up in one of the local minima. On the other hand, a very slow decrease of temperature would surely bring the system to the true minimum. However, such a slow process is not practically useful. One therefore has to determine carefully how fast to decrease the temperature in simulated annealings. On this problem, Geman and Geman [46] proved that the decrease of temperature as $T = \text{const}/\log t$, with the proportionality constant roughly of the order of the system size, guarantees convergence to the optimal state for a wide class of combinatorial optimization problems. This inverse-log law is still too slow for most practical purposes. Nevertheless this result serves as a mathematical background of empirical investigations by numerical methods.

There have been a few proposals to accelerate the annealing schedule by modifying the transition probabilities used in the conventional simulated annealing. Szu and Hartly [105] pointed out for a problem defined in a continuum space that occasional non-local samplings significantly improve the performance, leading to an annealing schedule inversely proportional to time $T = \text{const}/t$. This non-local sampling corresponds to modification of the generation probability (or, more precisely, the neighbourhood) to be defined later. Tsallis and Stariolo [54] proposed to modify the acceptance probability generalizing the usual Boltzmann form in addition to the generation probability (which they call the visiting distribution). Numerical investigations show faster convergence to the optimal state by annealing processes using their generalized transition probability or its modifications [54, 106]. Szu and Hartley and Tsallis and Stariolo proved that the modified *generation* probability assures convergence to the optimal state under a power-low decrease of the temperature as a function of time. However there has been no mathematically rigorous argument on the convergence under the generalized *acceptance* probability of Tsallis and Stariolo.

We prove in the present chapter that the inhomogeneous Markov process generated by the generalized transition (acceptance) probability of Tsallis and Stariolo actually satisfies the property of weak ergodicity under an annealing schedule inversely proportional to the power of time. Rigorously speaking, weak ergodicity (which roughly means independence of the probability distribution from the initial condition) itself does not immediately guarantee the convergence to the optimal state. Nevertheless our result is expected to be close enough to this final goal because the probability distribution would depend upon the initial condition if the annealing schedule is not appropriately chosen.

Various definitions are given in the next section. The proof of our main theorem appears in section

6.3. An example of fast convergence by the generalized transition probability is discussed in section 6.4 for a parameter range not covered by the theorem in section 6.3. In section 6.5 we investigate if we may further generalize the transition probability in the case of the simple model discussed in section 6.4. The final section is devoted to discussions on the significance of our result.

6.2 Inhomogeneous Markov chain

Let us first list up various definitions to fix notations. We consider a problem of combinatorial optimization with the space of states denoted by $\mathcal{S}$. The *cost function* E is a real single-valued function on $\mathcal{S}$. The goal of a combinatorial optimization problem is to find the minimum (or minima) of the cost function. For this purpose we introduce the process of simulated annealing using the Markov chain generated by the *transition probability* from state $x(\in \mathcal{S})$ to state $y(\in \mathcal{S})$ at time step t:

$$G(x,y;t) = \begin{cases} P(x,y)A(x,y;T(t)) & (x \neq y) \\ 1 - \sum_{z(\neq x)} P(x,z)A(x,z;T(t)) & (x = y) \end{cases}, \tag{6.1}$$

where $P(x,y)$ is the *generation probability*

$$P(x,y) \begin{cases} > 0 & (y \in \mathcal{S}_x) \\ = 0 & (\text{otherwise}) \end{cases} \tag{6.2}$$

with $\mathcal{S}_x$ the *neighbourhood* of x (the set of states that can be reached by a single step from x), and $A(x,y;T)$ is the *acceptance probability*. In the case of the generalized transition probability, the acceptance probability is given as [54]

$$\begin{aligned} A(x,y;T) &= \min\{1, u(x,y;T)\} \\ u(x,y;T) &= \left(1 + (q-1)\frac{E(y) - E(x)}{T}\right)^{1/(1-q)} \end{aligned} \tag{6.3}$$

where q is a real parameter. For technical reasons we have to restrict ourselves to the region $q > 1$ in this and the next sections. This acceptance probability reduces to the usual Boltzmann form in the limit $q \to 1$. The present Markov chain is *inhomogeneous*, i.e., the transition probability (6.1) depends on the time step t.

We choose the *annealing schedule*, or the t-dependence of the parameter T (the *temperature*), as

$$T(t) = \frac{b}{(t+2)^c} \quad (b, c > 0, t = 0, 1, 2, \cdots). \tag{6.4}$$

To analyze the inhomogeneous Markov chain generated by the above transition probability, we introduce the *transition matrix* $G(t)$ with the element

$$[G(t)]_{x,y} = G(x,y;t). \tag{6.5}$$

Let us write the set of probability distributions on $\mathcal{S}$ as $\mathcal{P}$. A probability distribution $p(\in \mathcal{P})$ may be regarded as a vector with the component $[p]_x = p(x)(x \in \mathcal{S})$. The probability distribution at time step t, starting from an initial distribution $p_0(\in \mathcal{P})$ at time s, is

$$p(s,t) = p_0 G(s,t) = p_0 G(s) G(s+1) \cdots G(t-1). \tag{6.6}$$

The *coefficient of ergodicity* is defined as

$$\alpha(G) = 1 - \min\{\sum_{z \in \mathcal{S}} \min\{G(x,z), G(y,z)\} | x, y \in \mathcal{S}\}. \tag{6.7}$$

We shall prove in the next section the property of *weak ergodicity* for the present Markov chain, which means that the probability distribution function after sufficiently long time becomes independent of the initial condition:

$$\forall s \geq 0 : \lim_{t \to \infty} \sup\{\|p_1(s,t) - p_2(s,t)\| \mid p_{01}, p_{02} \in \mathcal{P}\} = 0 \tag{6.8}$$

where $p_1(s,t)$ and $p_2(s,t)$ are the probability distributions with different initial conditions p_{01} and p_{02}:

$$p_1(s,t) = p_{01}G(s,t) \tag{6.9}$$
$$p_2(s,t) = p_{02}G(s,t). \tag{6.10}$$

The norm is defined by

$$\|p\| = \sum_{x \in \mathcal{S}} |p(x)|. \tag{6.11}$$

Although we focus our attention on weak ergodicity in the present chapter, it may be useful as a reference to recall the definition of *strong ergodicity*:

$$\exists r \in \mathcal{P}, \forall s \geq 0 : \lim_{t \to \infty} \sup\{\|p(s,t) - r\| \mid p_0 \in \mathcal{P}\} = 0. \tag{6.12}$$

The following theorems give criteria for weak and strong ergodicity [103, 104]:

Theorem 1 (Condition for weak ergodicity) *An inhomogeneous Markov chain is weakly ergodic if and only if there exists a strictly increasing sequence of positive numbers*

$$t_0 < t_1 < \cdots < t_i < t_{i+1} < \cdots$$

such that

$$\sum_{i=0}^{\infty} (1 - \alpha(G(t_i, t_{i+1}))) = \infty. \tag{6.13}$$

Theorem 2 (Condition for strong ergodicity) *An inhomogeneous Markov chain is strongly ergodic if it satisfies the following conditions:*

1. *it is weakly ergodic*
2. *there exists* $p_t \in \mathcal{P}(\forall t \geq 0)$ *such that* $p_t = p_t G(t)$
3. p_t *satisfies*

$$\sum_{t=0}^{\infty} \|p_t - p_{t+1}\| < \infty. \tag{6.14}$$

6.3 Weak ergodicity

We prove in the present section that the condition in Theorem 1 is satisfied by the present inhomogeneous Markov chain generated by the generalized transition probability. The argument closely follows that for the conventional Boltzmann-type transition probability [46, 103, 104]. We need the following Lemma for this purpose.

Lemma 1 (Lower bound on the transition probability) *The elements of the transition matrix satisfy the following bounds. For off-diagonal elements,*

$$P(x,y) > 0 \Rightarrow \forall t \geq 0 : G(x,y;t) \geq w \left(1 + \frac{(q-1)L}{T(t)}\right)^{1/(1-q)}, \tag{6.15}$$

and for diagonal elements,

$$\forall x \in \mathcal{S} - \mathcal{S}_m, \exists t_1 > 0, \forall t \geq t_1 : G(x,x;t) \geq w \left(1 + \frac{(q-1)L}{T(t)}\right)^{1/(1-q)} \tag{6.16}$$

where $\mathcal{S}_m$ is the set of locally maximum states

$$\mathcal{S}_m = \{x | x \in \mathcal{S}, \forall y \in \mathcal{S}_x : E(y) \leq E(x)\}, \tag{6.17}$$

L denotes the maximum change of the cost function by a single step

$$L = \max\{|E(x) - E(y)| \mid P(x,y) > 0\} \tag{6.18}$$

and w is the minimum value of $P(x,y)$

$$w = \min\{P(x,y) \mid P(x,y) > 0, x, y \in \mathcal{S}\}. \tag{6.19}$$

Proof.

First we prove (6.15). When $E(y) - E(x) > 0$, we have $u(x,y;T(t)) \leq 1$ and thus

$$\begin{aligned} G(x,y;t) &= P(x,y)A(x,y:T(t)) \\ &\geq w \min\{1, u(x,y;T(t))\} \\ &= w\, u(x,y;T(t)) \\ &\geq w \left(1 + \frac{(q-1)L}{T(t)}\right)^{1/(1-q)}. \end{aligned} \tag{6.20}$$

If $E(y) - E(x) \leq 0$, $u(x,y;T(t)) \geq 1$ and therefore

$$\begin{aligned} G(x,y;t) &\geq w \min\{1, u(x,y;T(t))\} \\ &= w \\ &\geq w \left(1 + \frac{(q-1)L}{T(t)}\right)^{1/(1-q)}. \end{aligned} \tag{6.21}$$

We next prove (6.16). Since $x \in \mathcal{S} - \mathcal{S}_m$, there exists a state $z \in \mathcal{S}_x$ satisfying $E(z) - E(x) > 0$. For such a state z,

$$\lim_{t \to \infty} u(x,z;T(t)) = 0 \tag{6.22}$$

and consequently

$$\lim_{t \to \infty} \min\{1, u(x,z;T(t))\} = 0. \tag{6.23}$$

Then $\min\{1, u(x,z;T(t))\}$ can be made arbitrarily small for sufficiently large t. More precisely, there exists $t_1 > 0$ and $0 < \epsilon < 1$ such that

$$\forall t \geq t_1 : \min\{1, u(x,z;T(t))\} < \epsilon. \tag{6.24}$$

We therefore have

$$\sum_{y \in \mathcal{S}} P(x,y)A(x,y;T(t)) = P(x,z)\min\{1, u(x,z;T(t))\} + \sum_{y \in \mathcal{S} - \{z\}} P(x,y)\min\{1, u(x,y;T(t))\}$$

$$\begin{aligned} &< P(x,z)\epsilon + \sum_{y\in\mathcal{S}-\{z\}} P(x,y) \\ &= -(1-\epsilon)P(x,z) + 1. \end{aligned} \tag{6.25}$$

The diagonal element of (6.1) thus satisfies

$$\begin{aligned} G(x,x;t) &\geq (1-\epsilon)P(x,z) \\ &\geq w\left(1+\frac{(q-1)L}{T(t)}\right)^{1/(1-q)} \end{aligned} \tag{6.26}$$

where we have used that the last factor can be made arbitrarily small for sufficiently large t. ∎

We use the following notations in the proof of weak ergodicity. The minimum number of state transitions to reach y from x (or *vice versa*) is written as $d(x,y)$. One can then reach any state from x within $k(x)$ steps:

$$k(x) = \max\{d(x,y)|y\in\mathcal{S}\}. \tag{6.27}$$

The minimum of $k(x)$ for $x\in\mathcal{S}-\mathcal{S}_m$ is denoted as R, and the state giving this minimum value is x^*:

$$R = \min\{k(x)|x\in\mathcal{S}-\mathcal{S}_m\} \tag{6.28}$$

$$x^* = \arg\min\{k(x)|x\in\mathcal{S}-\mathcal{S}_m\}. \tag{6.29}$$

Theorem 3 (Weak ergodicity) *The inhomogeneous Markov chain defined in Section 6.2 is weakly ergodic if* $0 < c \leq (q-1)/R$.

Proof.

Consider a transition from state x to x^*. According to the definition (6.6) of the double-time transition matrix, we have

$$G(x,x^*;t-R,t) = \sum_{x_1,\cdots,x_{R-1}} G(x,x_1;t-R)G(x_1,x_2;t-R+1)\cdots G(x_{R-1},x^*;t-1). \tag{6.30}$$

From the definitions of x^* and R, there exists at least one sequence of transitions to reach x^* from x within R steps such that

$$x\neq x_1\neq x_2\neq\cdots\neq x_k = x_{k+1}\cdots = x_R = x^*. \tag{6.31}$$

If we keep only such a sequence in the summation of (6.30) and use Lemma 1,

$$\begin{aligned} G(x,x^*;t-R,t) &\geq G(x,x_1;t-R)G(x_1,x_2;t-R+1)\cdots G(x_{R-1},x_R;t-1) \\ &\geq \prod_{k=1}^{R} w\left(1+\frac{(q-1)L}{T(t-R+k-1)}\right)^{1/(1-q)} \\ &\geq w^R\left(1+\frac{(q-1)L}{T(t-1)}\right)^{R/(1-q)}. \end{aligned} \tag{6.32}$$

Then the coefficient of ergodicity satisfies

$$\begin{aligned} \alpha(G(t-R,t)) &= 1-\min\{\sum_{z\in\mathcal{S}}\min\{G(x,z;t-R,t),G(y,z;t-R,t)\}|x,y\in\mathcal{S}\} \\ &\leq 1-\min\{\min\{G(x,x^*;t-R,t),G(y,x^*;t-R,t)\}|x,y\in\mathcal{S}\} \\ &\leq 1-w^R\left(1+\frac{(q-1)L}{T(t-1)}\right)^{R/(1-q)}. \end{aligned} \tag{6.33}$$

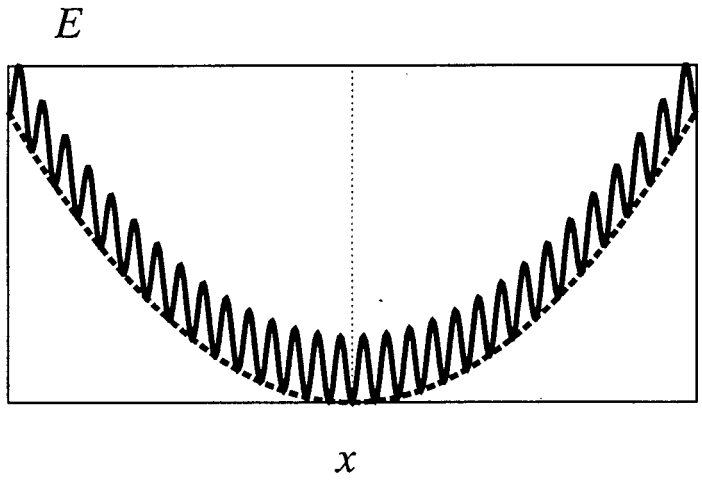

Figure 6.1: Energy landscape of the parabola potential.

We now use the annealing schedule (6.4). There exists a non-negative integer k_0 such that the following inequalities hold for all $k \geq k_0$:

$$\begin{aligned} 1-\alpha(G(kR-R,kR)) &\geq w^R\left(1+\frac{(q-1)L(kR+1)^c}{b}\right)^{R/(1-q)} \\ &\geq w^R\left(\frac{2(q-1)LR^c}{b}\left(k+\frac{1}{R}\right)^c\right)^{R/(1-q)} . \end{aligned} \tag{6.34}$$

It is clear from (6.34) that the summation

$$\sum_{k=0}^{\infty}(1-\alpha(G(kR-R,kR))) \geq \sum_{k=k_0}^{\infty}(1-\alpha(G(kR-R,kR))) \tag{6.35}$$

diverges if c satisfies $0 < c \leq (q-1)/R$. This proves weak ergodicity according to Theorem 1. ∎

Remark. The arguments developed in Sections 6.2 and 6.3 break down for $q < 1$. For instance, the acceptance probability (6.3) becomes imaginary for sufficiently small T if $E(y) - E(x) > 0$ and $q < 1$. Nevertheless Theorem 3 does not exclude the possibility that the present Markov chain is weakly ergodic for $q < 1$ or that it is strongly ergodic for arbitrary q.

6.4 Example for $q < 1$

It is instructive to investigate a simple solvable model with the parameter $q < 1$ because the general analysis in the previous section excluded this range of q for technical reasons. The one-dimensional model discussed by Shinomoto and Kabashima [93] is particularly suited for this purpose.

They considered the thermal diffusion process of an object in a one-dimensional space. The object is located on one of the discrete positions $x = ai$, with i an integer, and is under the potential $E(x) = x^2/2$ (see Figure 6.1).

Hoppings to neighboring positions $i+1$ and $i-1$ take place if thermal fluctuations allow the object to climb over the barriers with height B for the process $i \to i-1$ and height $B+\Delta_i$ for $i \to i+1$ (see Figure 6.2), where Δ_i is the difference of the potentials at neighbouring locations $\Delta_i = E(a(i+1)) - E(ai)$. By adaptively optimizing the temperature at each give time, they found that the energy $y = \langle E(x) \rangle$ (the expectation value of the potential) decreases as $y \sim B/\log t$. The optimum annealing schedule $T_{\rm opt}(t)$ was shown to have this same asymptotic behavior as a function of t. We show in the present section that the generalized transition probability with $q = 1/2$ leads to a much faster convergence of the energy.

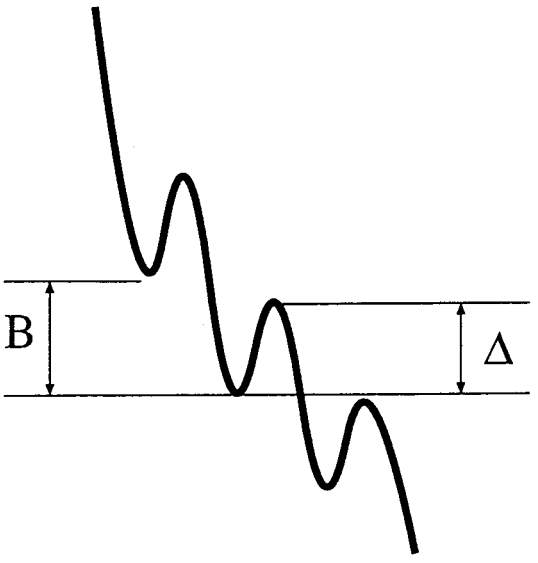

Figure 6.2: Local structure of the parabola potential.

The problem is defined by the master equation describing the time evolution of the probability P_i that the object is at the ith position at time t:

$$\begin{aligned}\frac{dP_i}{dt} &= \left(1+(q-1)\frac{B}{T}\right)^{1/(1-q)} P_{i+1} + \left(1+(q-1)\frac{B+\Delta_{i-1}}{T}\right)^{1/(1-q)} P_{i-1} \\ &- \left(1+(q-1)\frac{B+\Delta_i}{T}\right)^{1/(1-q)} P_i - \left(1+(q-1)\frac{B}{T}\right)^{1/(1-q)} P_i. \end{aligned} \tag{6.36}$$

It is straightforward to show that this master equation reduces to the following Fokker-Planck equation in the continuum limit $a \to 0$

$$\frac{\partial P}{\partial t} = \gamma(T)\frac{\partial}{\partial x}(xP) + D(T)\frac{\partial^2 P}{\partial x^2} \tag{6.37}$$

where

$$\gamma(T) = \frac{1}{T}\left(1+(q-1)\frac{B}{T}\right)^{q/(1-q)} \tag{6.38}$$

$$D(T) = \left(1+(q-1)\frac{B}{T}\right)^{1/(1-q)}. \tag{6.39}$$

We have rescaled the time unit by $1/a^2$ as in [93]. In Appendix F, we explain the derivation of the Fokker-Plank equation in detail.

Our aim is to find the fastest possible asymptotic decrease of the expectation value of the potential defined by

$$y = \int dx E(x) P(x,t) \tag{6.40}$$

by adaptively changing T as a function of time. Differentiating both sides of the definition (6.40) and using the Fokker-Planck equation (6.37), we obtain the following equation describing the time evolution of y:

$$\frac{dy}{dt} = -2\gamma(T)y + D(T). \tag{6.41}$$

The temperature is adaptively optimized by extremizing this right hand side with respect to T, yielding

$$\begin{aligned} T_{\text{opt}} &= \frac{2yB + (1-q)B^2}{2y + B} \\ &= (1-q)B + 2qy + O(y^2). \end{aligned} \tag{6.42}$$

The evolution equation (6.41) then has the form

$$\frac{dy}{dt} = -\frac{2}{B}\left(\frac{2q}{1-q}\right)^{q/(1-q)} y^{1/(1-q)}. \tag{6.43}$$

The solution is

$$y = B^{q/(1-q)}\left(\frac{1-q}{2q}\right)^{1/q} t^{-(1-q)/q}. \tag{6.44}$$

The optimum annealing schedule (6.42) is now

$$T_{\text{opt}} \sim (1-q)B + \text{const} \cdot t^{-(1-q)/q}. \tag{6.45}$$

The asymptotic behavior of the average position can be calculated in the same way. The result is

$$\begin{aligned} \langle x \rangle &= \int dx\, x P(x,t) \\ &\sim \text{const} \cdot t^{-1/2q}. \end{aligned} \tag{6.46}$$

It is useful to restrict of the value of q to avoid unphysical behavior of the generalized transition probability in the present one-dimensional problem. One of the transition probabilities in the master equation (6.36)

$$\left(1 + (q-1)\frac{B + \Delta_{i-1}}{T}\right)^{1/(1-q)} \tag{6.47}$$

reduces for $T = (1-q)B + O(y)$ to

$$\left(\frac{a^2 - 2ax}{2B} + O(y)\right)^{1/(1-q)}. \tag{6.48}$$

This quantity must be a small positive number for any x and sufficiently small (but fixed) a. This requirement is satisfied if

$$q = 1 - \frac{1}{2n} \quad (n = 1, 2, \cdots). \tag{6.49}$$

Consistency of the other transition probabilities in (6.36) is also guaranteed under (6.49).

The fastest decrease of the energy is achieved when $q = 1/2$. With this value of q,

$$y \sim \frac{B}{4} t^{-1} \tag{6.50}$$

$$T_{\text{opt}} \sim \frac{B}{2} + \frac{B}{4} t^{-1} \tag{6.51}$$

$$\langle x \rangle \sim \text{const} \cdot t^{-1}. \tag{6.52}$$

It may be useful to remark that the non-vanishing value $(1-q)B$ of the temperature (6.45) in the infinite-time limit does not cause troubles. What is required is not an asymptotically vanishing value of the temperature but that the probability distribution does not change with time in the infinite-time limit. This condition is satisfied if $T = (1-q)B$ as is apparent from (6.37) with (6.38) and (6.39).

The results (6.50) and (6.51) show asymptotic relaxations proportional to t^{-1} which is much faster than those for the conventional transition probability, $B/\log t$ [93].

6.5 More general transition probability

A natural question may arise on how far the arguments in the previous sections depend on the specific form of the acceptance probability (6.3). We investigate this problem for the one-dimensional model treated in the preceding section.

The master equation is now generalized to

$$\frac{dP_i}{dt} = f\left(\frac{B}{T}\right) P_{i+1} + f\left(\frac{B+\Delta_{i-1}}{T}\right) P_{i-1} - f\left(\frac{B+\Delta_i}{T}\right) P_i - f\left(\frac{B}{T}\right) P_i. \tag{6.53}$$

The same Fokker-Planck equation (6.37) is derived in the limit $a \to 0$ with the following parameters

$$\gamma(T) = -\frac{1}{T^2} f'\left(\frac{B}{T}\right) \tag{6.54}$$

$$D(T) = f\left(\frac{B}{T}\right). \tag{6.55}$$

The expectation value of the potential obeys the same evolution equation as in (6.41):

$$\frac{dy}{dt} = -2\gamma(T)y + D(T) = \frac{2y}{T} f'\left(\frac{B}{T}\right) + f\left(\frac{B}{T}\right) \equiv \mathcal{L}\left(\frac{1}{T}\right). \tag{6.56}$$

Minimization of $\mathcal{L}(v)(v = 1/T)$ with respect to T for given y leads to

$$2vyBf''(Bv) + (2y + B)f'(Bv) = 0. \tag{6.57}$$

The solution of this equation for v gives the optimal annealing schedule

$$\frac{1}{T_{\text{opt}}} = v = g(y). \tag{6.58}$$

Assuming analyticity of $g(y)$ as $y \to 0$, we write (6.58) as

$$v = c_1 + c_2 y + O(y^2). \tag{6.59}$$

It is required that the system stops its time evolution as $y \to 0$ and $v \to c_1$. We then have $f(Bc_1) = 0$ from (6.56) assuming c_1 is finite. (This condition of $c_1 < \infty$ is not satisfied by the conventional acceptance probability in which $1/v = T \to 0(c_1 \to \infty)$ as $y \to 0$.) It is also necessary that the minimization condition (6.57) is satisfied in the same limit, leading to $f'(Bc_1) = 0$. These two conditions on f and f' are satisfied if $f(Bv)(= f(Bc_1 + Bc_2 y))$ and its derivative behave for small y

$$f(Bv) \sim c_3 y^k, \quad f'(Bv) \sim -c_4 y^{k-1}. \tag{6.60}$$

Here $k > 1$ and $c_3, c_4 > 0$. The minus sign in front of c_4 comes from the observation that an increase of the inverse temperature $v = 1/T$ means a decrease of the energy y and therefore the differentiations by v and y should be done with the opposite sign (*i.e.* $c_2 < 0$).

The evolution equation (6.56) then has a form

$$\frac{dy}{dt} = -c_5 y^k \tag{6.61}$$

with positive c_5 if $2c_1 c_4 > c_3$. This equation is solved as

$$y = (c_5(k-1)t)^{-1/(k-1)}. \tag{6.62}$$

This shows a power decay of the expectation value of the cost function.

It is useful to set a restriction on k as in the preceding section for q. The following acceptance probability for $y \to 0$ should be positive for any x:

$$f\left(\frac{B+\Delta_{i-1}}{T}\right) \sim f(Bc_1 + Bc_1\Delta_{i-1}) \sim \left(\frac{a^2}{2} - ax\right)^k, \tag{6.63}$$

where we have used (6.60). This requirement is satisfied if k is a positive even number $k = 2n$. The energy (6.62) then decays as

$$y \sim t^{-1}, t^{-1/3}, t^{-1/5}, \cdots, \tag{6.64}$$

the same formula as in the preceding section. In fact the argument in section 6.4 is recovered if we choose

$$f(v) = (1 + (q-1)v)^{1/(1-q)}. \tag{6.65}$$

In this way the fast decrease of the energy has been obtained for a very general acceptance probability distribution function satisfying certain analyticity conditions.

6.6 Discussion

We have proved weak ergodicity of the inhomogeneous Markov process generated by the generalized transition probability under certain conditions on the parameters. For technical reasons we were unable to prove strong ergodicity, or more strongly, convergence to the optimal distribution function. We could not show that the condition (6.14) of Theorem 2 is satisfied by the present inhomogenous Markov chain. However, it is unlikely that the property of convergence to the optimal state does not hold because weak ergodicity alone already means that the state of the system asymptotically becomes independent of the initial condition; it is most likely that such an asymptotic state is the optimal one as mentioned in section 6.1.

It is appropriate to comment on the computational complexity here. The time t_1 necessary for the temperature (6.4) to reach a small specified value δ is obtained by solving the relation $b/t_1^c \sim \delta$ $(c = (q-1)/R)$ for t_1:

$$t_1 \sim \exp\left(\frac{k_1 N}{q-1} \log \frac{b}{\delta}\right). \tag{6.66}$$

Here we have set $R = k_1 N$ with N the system size because R defined in (6.28) is roughly of this order of magnitude in many cases. For example, in the problem of spin glasses, one can reach any spin configuration by flipping at most N spins. The corresponding time for the conventional simulated annealing is

$$t_2 \sim \exp\left(\frac{k_2 N}{\delta}\right) \tag{6.67}$$

which has been obtained from $k_2 N/\log t_2 \sim \delta$. A comparison of (6.66) with (6.67) reveals that the coefficient of N in the exponent has been reduced from $1/\delta$ to $\log 1/\delta$ by using the generalized transition probability. In this sense, $t_1 \ll t_2$. Since we have proved Theorem 3 under very general conditions on the system (which would include problems with NP completeness), it is not possible to reach a low temperature state in polynomial time. The best we could archive is an improvement of the coefficient in the exponent.

One should be careful that the rapid decrease of the temperature does not immediately mean a rapid decrease of the cost function. This aspect can be checked by comparing the acceptance probability (6.3) at $T = \delta$

$$u_1(t)(T = \delta) = \left(\frac{\delta}{(q-1)\Delta E}\right)^{1/(q-1)} \tag{6.68}$$

with the corresponding one for the conventional transition

$$u_2(T = \delta) = \sim \exp(-\Delta E/\delta). \tag{6.69}$$

Since $u_1(t_1) \gg u_2(t_2)$ if $\Delta E/\delta \gg 1$, we see that the generalized transition probability at a given temperature has a larger value to include transitions into states with high values of the cost function than in the case of the conventional one at the same temperature. Thus expectation value of the cost function may be larger under the generalized transition probability than under the conventional Boltzmann form at the same temperature if one waits sufficiently long until thermal equilibrium is reached. This phenomenon has actually been observed in a numerical investigation under a slightly different (but essentially similar)

situation [107]. Therefore, if the expectation value of the cost function is observed in a numerical simulation to indeed decrease rapidly under the generalized transition probability , it would be not only for the rapid decrease of the temperature but also because the relaxation time is shorter. The conventional transition probability may give a larger possibility for the system to stay longer in local minima with high values of the cost function. A mathematical analysis of this property of quick relaxation by the generalized transition probability is beyond the scope of the present thesis. However, one may naively expect it to happen from the larger probability to climb over high barrier as discussed above.

It should be remarked that Theorem 3 with the annealing schedule (6.4) does not give a practically useful prescription of simulated annealing. In actual numerical simulations one rarely use such annealing schedules as (6.4) obtained from worst-case estimates. Even exponentially fast decreases of the temperature often give satisfactory results in the conventional and generalized methods (see [106] and references in [54]). The significance of Theorem 3 is that convergence (in the sense of weak ergodicity) has anyhow been proved with the annealing schedule (6.4) under the generalized transition (acceptance) probability where only empirical numerical investigations have been carried out without mathematical guarantee of convergence under any annealing schedule. æ

Chapter 7

Summary and Concluding Remarks

In this thesis we investigated two topics about the keywords, *learning* and *optimization* from a statistical mechanical point of view. In this chapter, we summarize our results.

Firstly we investigated the generalization abilities of a simple perceptron in the context of supervised learning. We took up the realistic learning situation in which structures of the teacher and student are different. Especially, we investigated the situation in which a simple perceptron learns a task of the non-monotonic (or reversed-wedge) perceptron. As the teacher machine has greater abilities than the student, this task is unrealizable for the student. Difficulties of learning for the student continuously change by controlling the width of the reversed wedge a. The case of $a \to \infty$ is special and for this case the task of the teacher becomes realizable.

For this case of structural mismatch in supervised learning, we first investigated the generalization abilities of the student in the off-line (or batch) learning scenario in chapter 3. In this learning mode, we chose the Gardner-Derrida cost function which means the number of disagreements between the training sets and student outputs. From this cost function, we calculated the free energy using the standard statistical mechanical procedure with the replica trick. This free energy corresponds to the so-called Gardner volume if the number of examples is smaller than the critical capacity of the student. We estimated the saddle point of the free energy using the replica symmetric ansatz and obtained the saddle point equation for all values of a. Results are summarized as follows.

For the large a region, namely, the case in which a satisfies $a > a_{c0} \sim 1.53$, the generalization error decreases monotonically to its theoretical lower bound and the asymptotic form of the generalization error behaves as $\epsilon_g \sim \epsilon_{\min} + \alpha^{-1}$. For the parameter region $a_{c0} > a > a_{c1} \sim 0.8$, we found that the generalization error shows a discontinuous phase transition at $\alpha \sim \alpha_{th} = 14.7$ and at this point the student machine suddenly obtains a high generalization ability. For this parameter region, we also investigated the asymptotic behavior of the generalization error and found that $\epsilon_g \sim \epsilon_{\min} + \alpha^{-1}$, where $\epsilon_{\min}$ is the best possible value for this parameter region. Next we investigated the case of $a_{c1} > a > a_{c2} = \sqrt{2\log 2}$. In this parameter region, the discontinuous phase transition from the poor generalization phase to the good generalization phase was again observed. However, we found that the spinodal point α_{sp} becomes infinity in contrast to the previous region, which means that the quasi-stable solution beyond α_{th} persists even in the limit $\alpha \to \infty$. In addition, the asymptotic form of the generalization error for this parameter region is very singular from a statistics point of view. We found that $\epsilon_g \sim \epsilon_{\min} + \alpha^{-2/3}$. This exponent of learning curve is very similar to the learning of blurred boundary in low dimension [18].

We next investigated the small a case, namely, $a_{c2} > a > 0$. For this case, the first order phase transition does not occur and the generalization error asymptotically converges to the optimal value as $\epsilon_g \sim \epsilon_{\min} + \alpha^{-2/3}$. In order to obtain the information about the generalization ability by solving the saddle point equations, we assumed that several order parameters appearing in the free energy are independent of the replica index, namely, the replica symmetric ansatz. Therefore, it is necessary for us to check the validity of the ansatz. We checked this point from different two points of view. We first calculated the AT line for our model system and found that the replica symmetric solution is stable if the number of examples α is smaller than the critical capacity α_c and unstable if $\alpha > \alpha_c$. Therefore,

for the asymptotic region, the replica symmetric solution is unstable. However, we should investigate to what extent our approximation is reliable. For this purpose, we checked the exponent of the asymptotic decay of the learning curves by computer simulation for the two-dimensional case and concluded that our approximation is very good in the sense that the exponent may be unchanged even if we calculated the replica symmetry breaking solution for our model system. We would like to emphasize that the off-line learning algorithm with the Gardner-Derrida cost function leads the student to the convergence to the best possible value of the generalization error for all values of parameter a.

In chapter 4, we investigated the same learning system as in chapter 3 in the context of the on-line learning scenario.

In on-line learning mode, the processes of learning can be represented by a coupled differential equations with respect to the macroscopic order parameters l and R, namely, the norm of the student and the overlap between the teacher and student. We fist investigated the well-known three learning algorithms, namely, perceptron, Hebbian and AdaTron learning algorithms. For these algorithms, if a task of the teacher is realizable, namely, $a \to \infty$, the performance of each learning strategy has been investigated by several authors. The results can be summarized as follows.

In the asymptotic region, learning curves for the perceptron, Hebbian and AdaTron learning algorithms are $\epsilon_g \sim \alpha^{-1/3}$, $\epsilon_g \sim \alpha^{-1/2}$ and $\epsilon_g \sim \alpha^{-1}$ respectively. Among these results, we found that the generalization error for the AdaTron learning algorithm has faster convergence to zero than the previous result $\epsilon_g \sim (3/2)\alpha^{-1}$ which was reported by Biehl and Riegler [98]. If we choose the initial length of the student weight optimally, the generalization error converges to zero as $\epsilon_g \sim (4/3)\alpha^{-1}$. It is important for us to bear in mind that in this on-line learning mode, there was no phase transition as we saw in the off-line learning scenario.

As the case of finite a is highly non-trivial, we investigated the behaviors of learning curves in detail. At first, for the perceptron learning algorithm, we found that the flows of R and l converge to one of the local minima and the student fails to obtain the theoretical lower bound of the generalization error for all values of the parameter a. For the Hebbian learning, the generalization error converges to the best possible value for the case of $a > a_{c1} \sim 0.80$, but, if the parameter a is smaller than a_{c1}, an over-training occurs and the generalization error converges to worse value which is larger than 0.5 (the case of random guess).

For the AdaTron learning algorithm, similarly to the perceptron learning algorithm, flows of the order parameters are trapped in a local minimum and the corresponding generalization error can not converge to the best possible value for all values of parameter a. For both perceptron and AdaTron learning algorithms, convergence of the generalization error to the non-optimal value is exponentially fast, and as soon as the parameter a goes to infinity, the relaxation time diverges and exponential decays slow down to the power law. In addition, the residual error for the AdaTron learning algorithm is larger than that of the perceptron learning algorithm for all values of the width of the reversed wedge a. We concluded that the accuracy of the on-line learning is worse than the off-line learning for finite a cases although the on-line learning can save the training time in comparison with the off-line learning.

Nevertheless, we showed that this worse accuracy can be overcome by improving these conventional learning algorithms. We first introduced a time dependent learning rate into the learning dynamics and optimize it so as to maximize the rate of increase of overlap R at every time step. It is obvious that this strategy works well for the case of a larger than a_{c1} because the optimal state $R_{\rm opt}$ for this region is $+1$. For this parameter range, the generalization error of the perceptron and AdaTron learning algorithms converges to the best possible value as $\sim \alpha^{-1}$. Here we should notice that the case $a = a_{c2} = \sqrt{2\log 2}$ is special and the learning rate vanishes for this parameter value. On the other hand, for the Hebbian learning, the speed of convergence did not change by the optimized time dependent learning rate. Although the optimal time dependent learning rate depends on the unknown parameters for the student, for example, a or R, we also succeeded in constructing unknown parameter-free optimization using the information of the asymptotic form of the learning late.

For the parameter region $a < a_{c1}$, we introduced a weight decay term into the Hebbian learning and optimized it. As a result, the generalization error converged to the best possible value.

We also investigated to what extent the query works well on our model system. In the context of the learning with query, the student trained by the Hebbian learning algorithm requires inputs lying on the

border line $u = 0$. For this learning scenario, we found that the corresponding learning curve converges to the optimal value as $\sim \alpha^{-1/2}$ for the case of $a > a_{c2} = \sqrt{2\log 2}$ and decays to the best possible value exponentially for the case of $a < a_{c1} = 0.8$. In addition, if we introduce the time dependent learning rate into the Hebbian learning algorithm with query and optimize it, the convergence rate of the learning curves becomes exponential for the parameter range $a > a_{c2}$.

In conclusion, in chapter 4, we found that if we improve the on-line learning algorithm by various ways, we can obtain the same speed of the convergence to the best possible value as the off-line learning.

In chapter 5, we investigated the learning process of a non-monotonic perceptron. In this learning system, a non-monotonic perceptron learns from the same type of non-monotonic perceptron. Therefore, this learning problem is s realizable case.

Intuitively, it seems that the generalization error converges to zero as long as we choose the conventional learning algorithm like a perceptron learning algorithm. However, we found that for the parameter region $a < a_{c2} = \sqrt{2\log 2}$, the perceptron and Hebbian learning algorithms lead to a worse generalization than the result of a random guess $\epsilon_g = 0.5$ contrary to our intuition. We called this curious phenomenon the *anti-learning*. On the other hand, the AdaTron learning failed to obtain the zero residual error for all finite a. In order to prevent the student from converging to the non-zero value of the generalization error, we introduced several learning algorithms. Among these modifications, the Baysian on-line learning showed the best performance. In this Baysian optimization context, we introduced the learning function $f(T_a(v), u)$, instead of a learning rate g, into the on-line learning dynamics and optimized it so as to accelerate the increase of the order parameter R at every time step. As this learning function contains the unknown variable v for the student machine, we averaged it over the distribution of v using the well-known Baysian formula. Using this Baysian optimization strategy, the generalization error converged to zero as $\epsilon_g \sim 0.883\,\alpha^{-1}$. We should note that this asymptotic form of the generalization error is independent of the width of the reversed wedge a and this exponent is a half of the bound for the off-line learning which was obtained by Opper and Haussler [75]. In addition, using this procedure based on Baysian statistics, we could obtain the useful information about the optimal queries.

In the analysis of off-line learning, we assumed that the stochastic process based on Metropolis samplings leads to the distribution with a delta peak around the optimal weight $\boldsymbol{J}_*$ if we cool down the system sufficiently slowly. Essentially we need such a stochastic process in order to prevent the system from being trapped in one of the local minima when the cost function has many local minima. However, it is not so clear whether the transition probability of Gibbs-Boltzmann type is the best of all candidates as the transition probability.

In addition, it seems a very important problem how the temperature of the system should be cooled down if we take up a new kind of transition probability. This problem is quite general and interesting not only for the problem of machine learning but for a lot of combinational optimization problems. In order to make this problem clear, we investigated the possibilities of different kind of transition probability, we especially chose the transition probability based on Tsallis statistics [53, 54] and discussed the optimal annealing schedule for the transition probability. We could proved weak ergodicity of the inhomogeneous Markov process generated by the generalized transition probability under certain conditions on the parameters. Although the weak ergodicity does not mean the convergence to the optimal distribution, it is possible to say that the final state is independent of the initial state and the most likely the asymptotic state is the optimal one if we use the optimal annealing schedule $T \sim t^{-c}$. We also investigated the convergence of the generalized annealing for the case of the one-dimensional parabola potential and we concluded that the convergence to the global minimum of the potential as $y \sim t^{-1}$ if one cools the system according to the schedule $T \sim t^{-1}$. From these examinations, we concluded that it may be possible to obtain the faster convergence to the ground state than the conventional annealing based on the Gibbs-Boltzmann statistics.

Our simulated annealing method based on the new type of the transition probability may be applied to various combinational optimization problems.

Most of our results in this thesis is rigorous or exact. Of course, it is hard for us to compare the results for our simple model with the actual information processing in the complicated real brain. However, some of our results are interesting from the experimental psychological point of view. For example, the first order phase transition (from the poor generalization to the exedent generalization) which we observed in the off-line learning scenario seems to have analogy with a *flash* in our real life. We also considered that the time complexity is one of the important problems in the real brain and investigated the performance of the on-line learning for a typical structural mismatch case and compared the results with those of the off-line learning. For example, we revealed that the Hebbian learning rule, which has been considered as a candidates of the simple learning rules in the real brain and used in the practical situations such as robotics, fails to obtain the theoretical lower bound of the generalization error and the over-training phenomena appear. Therefore, some sophisticated strategies should be needed and we proposed the several candidates for this purpose. The learning strategies which we proposed worked well on our model system and some of them succeeded in obtaining the exponential decays, which is much faster than the off-line learning, to the best possible value of the generalization error. Our model system is much simpler than the real brain, however, when we consider the more complicated network and its learning strategies, it should be important for us to take into account the limitation of the power for a single neuronal unit, as long as we have a conjecture that high performance of the real brain results from the collective behavior of many simple elements. Therefore, we should regard our results as a first step to understand the learning way in the real brain.

On the other hand, the optimization method which depends on the thermal fluctuation as the generalized transition probability may be applicable to the various problems.

We believe that our work may contribute to some mathematical foundations in the cross-disciplinary fields including the brain science, information science and physics.

Appendix A

Evaluation of $\ll \ln Z(\beta) \gg$

In this appendix, we derive eqs. (3.10) and (3.26) by the replica method. Under the replica symmetric ansatz (3.13) and (3.14), $\ll Z^n(\beta) \gg_{\xi^P}$ is evaluated by the saddle point method with respect to R and q as

$$\begin{aligned}
\ll Z^n(\beta) \gg_{\xi^P} &= \mathrm{ext}_{\{R,q\}} \Big\{ \int \prod_{a=0}^{n} d\boldsymbol{J}_a \prod_{a=1}^{n} \delta(\boldsymbol{J}_a \cdot \boldsymbol{J}_a - N) \prod_{a=1}^{n} \delta(\boldsymbol{B} \cdot \boldsymbol{J}_a - NR) \prod_{a>b} \delta(\boldsymbol{J}_a \cdot \boldsymbol{J}_b - Nq) \\
&\times \ll \prod_{a=1}^{n} \prod_{\mu=1}^{P} [e^{-\beta} + (1 - e^{-\beta})\, \Theta(y_\mu \cdot u_{a\mu})] \gg_{\xi^P} \Big\} \\
&= \mathrm{ext}_{\{R,q\}} \{ A_0(R, q : n) \times A_1(R, q, \beta : n) \}, \qquad \text{(A.1)}
\end{aligned}$$

$$u_{a\mu} \equiv \frac{\boldsymbol{J}_a \cdot \boldsymbol{x}_\mu}{\sqrt{N}}, \qquad \text{(A.2)}$$

$$y_\mu = T_a(v_\mu), \qquad \text{(A.3)}$$

$$v_\mu = \frac{\boldsymbol{B} \cdot \boldsymbol{x}_\mu}{\sqrt{N}}. \qquad \text{(A.4)}$$

In the last line of eq. (A.1), we defined $A_0(R, q : n)$ and $A_1(R, q, \beta : n)$ as

$$\begin{aligned}
A_0(R, q : n) &\equiv \int \prod_{a=0}^{n} d\boldsymbol{J}_a \prod_{a=1}^{n} \delta(\boldsymbol{J}_a \cdot \boldsymbol{J}_a - N) \prod_{a=1}^{n} \delta(\boldsymbol{B} \cdot \boldsymbol{J}_a - NR) \\
&\times \prod_{a>b} \delta(\boldsymbol{J}_a \cdot \boldsymbol{J}_b - Nq), \qquad \text{(A.5)}
\end{aligned}$$

and

$$A_1(R, q, \beta : n) \equiv \ll \prod_{a=1}^{n} \prod_{\mu=1}^{P} [e^{-\beta} + (1 - e^{-\beta}) \Theta(y_\mu \cdot u_{a\mu})] \gg_{\xi^P}, \qquad \text{(A.6)}$$

respectively. We first evaluate $A_0(R, q : n)$. In order to evaluate A_0, we rewrite the delta function appearing in A_0 by the integral form as

$$\begin{aligned}
A_0(R, q : n) &= \int \prod_{j,a} dJ_j^a \exp \left[i \left\{ F \left(\sum_{a<b} J_j^a J_b^j - qN \right) + G \left(\sum_a J_j^a B_j - NR \right) + E \left(\sum_a (J_j^a)^2 - N \right) \right\} \right] \\
&= \int \prod_{j,a} dJ_j^a \exp \left[i \left\{ F \sum_{a<b} J_j^a J_j^b + G \sum_a J_j^a B_j + E \sum_a (J_j^a)^2 \right\} \right] \\
&\times \exp \left[N \left(-\frac{in(n-1)qF}{2} - inGR - inE \right) \right]. \qquad \text{(A.7)}
\end{aligned}$$

By diagonalizing the $n \times n$ matrix $F\sum_{a<b} J_j^a J_j^b + G\sum_a J_j^a B_j + E\sum_a (J_j^a)^2$, we can actually calculate the Gaussian integral with respect to $\boldsymbol{J}^a$. After using the saddle point method with respect to E, F, and G, $A_0(R, q : n)$ is evaluated as

$$A_0(R, q : n) = \left[(1 + n\frac{q - R^2}{1 - q}) \times (1 - q)^n\right]^{N/2}, \tag{A.8}$$

except for a numerical factor [75]. Next, we evaluate $A_1(R, q, \beta : n)$. Since it is assumed that each $\boldsymbol{x}_\mu$ ($\mu = 1, 2 \cdots P$) is drawn independently and iteratively from an identical distribution, the average with respect to ξ^P in equation (A.6) is replaced by the product of the averages

$$\ll \prod_{a=1}^{n} [e^{-\beta} + (1 - e^{-\beta})\Theta(y_\mu \cdot u_{a\mu})] \gg_{(\boldsymbol{x}_\mu, y_\mu)}, \tag{A.9}$$

where $\mu = 1, \cdots P$. These averages are independent of index μ and therefore we drop μ in the evaluation of equation (A.9) from now on. For an input $\boldsymbol{x}$, the "probability" that $y = 1$ is returned by the non-monotonic teacher is

$$\begin{aligned} P(y = +1|\boldsymbol{x}) &= \Theta\left(-\frac{\boldsymbol{B} \cdot \boldsymbol{x}}{\sqrt{N}} - a\right) + \Theta\left(\frac{\boldsymbol{B} \cdot \boldsymbol{x}}{\sqrt{N}}\right) - \Theta\left(\frac{\boldsymbol{B} \cdot \boldsymbol{x}}{\sqrt{N}} - a\right) \\ &= \Theta(-v - a) + \Theta(v) - \Theta(v - a) \\ &= 1 - P(y = -1|\boldsymbol{x}) = P(y = -1| - \boldsymbol{x}). \end{aligned} \tag{A.10}$$

By taking the average with respect to y first in equation (A.9) using eq. (A.10) and taking the symmetry between $\boldsymbol{x}$ and $-\boldsymbol{x}$ into account, we obtain

$$\ll 2[\Theta(-v - a) + \Theta(v) - \Theta(v - a)] \prod_{a=1}^{n} [e^{-\beta} + (1 - e^{-\beta})\Theta(u_a)] \gg_{\boldsymbol{x}}. \tag{A.11}$$

It should be remarked that under the RS ansatz (3.13) and (3.14), v, u_1, $\cdots$ u_n become a set of joint Gaussian random variables which satisfy the condition

$$< u_a \cdot u_b >= (1 - q)\,\delta_{ab} + q \qquad \text{for } a, b = 1 \cdots, n, \tag{A.12}$$

$$< v \cdot u_a >= R \qquad \text{for } a = 1 \cdots, n, \tag{A.13}$$

$$< v^2 >= 1 \tag{A.14}$$

when $\boldsymbol{x}$ is uniformly drawn from S^N. Here, $< \cdots >$ stands for the statistical average of $\cdots$. These joint Gaussian random variables are represented explicitly by $n + 2$ independent Gaussian random variables z_a ($a = 0, 1, \cdots, n$) and t which satisfy the condition

$$< z_a \cdot z_b >= \delta_{ab} \qquad \text{for } a, b = 0, 1 \cdots, n, \tag{A.15}$$

$$< t \cdot z_a >= 0 \qquad \text{for } a = 0, 1 \cdots, n, \tag{A.16}$$

$$< t^2 >= 1 \tag{A.17}$$

as

$$u_a = \sqrt{1 - q}\, z_a + \sqrt{q} t \qquad \text{for } a = 1 \cdots, n, \tag{A.18}$$

$$v = \sqrt{1 - \frac{R^2}{q}}\, z_0 + \frac{R}{\sqrt{q}} t. \tag{A.19}$$

Substituting equations (A.18) and (A.19) into equation (A.11) and taking the average with respect to z_a ($a = 0, 1, \cdots, n$) and t instead of $\boldsymbol{J}$, we obtain

$$\ll \prod_{a=1}^{n} [e^{-\beta} + (1 - e^{-\beta})\Theta(y \cdot u_a)] \gg_{(\boldsymbol{x}, y)}$$

$$
\begin{aligned}
&= 2\int Dt\left[\int Dz_0\left[\Theta\left(-\sqrt{1-\frac{R^2}{q}}z_0-\frac{R}{\sqrt{q}}t-a\right)+\Theta\left(\sqrt{1-\frac{R^2}{q}}z_0+\frac{R}{\sqrt{q}}t\right)\right.\right.\\
&\left.\left.-\Theta\left(\sqrt{1-\frac{R^2}{q}}z_0+\frac{R}{\sqrt{q}}t-a\right)\right]\right]\\
&\times\prod_{a=1}^{n}\int Dz_a[e^{-\beta}+(1-e^{-\beta})\,\Theta(\sqrt{1-q}z_a+\sqrt{q}t)]\\
&= 2\int Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)\{\Xi_\beta(q:t)\}^n
\end{aligned}
\tag{A.20}
$$

which means

$$
A_1(R,q,\beta:n)=\left[2\int Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)\{\Xi_\beta(q:t)\}^n\right]^P. \tag{A.21}
$$

For small n, equations (A.8) and (A.21) is expanded with respect to n as

$$
A_0(q,R:n)\sim 1+n\times N\times\left[\frac{1}{2}\ln(1-q)+\frac{q-R^2}{2(1-q)}\right]+O(n^2), \tag{A.22}
$$

and

$$
A_1(q,R,\beta:n)\sim 1+n\times 2P\times\left[\int Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)\ln\Xi_\beta(q:t)\}\right]+O(n^2), \tag{A.23}
$$

respectively. From these, we obtain

$$
\begin{aligned}
\frac{\ll\ln Z(\beta)\gg_{\xi^P}}{N}=\mathrm{ext}_{\{R,q\}}\Bigg\{ &\quad 2\alpha\int Dt\,\Omega\left(\frac{R}{\sqrt{q}}:t\right)\ln\Xi_\beta(q:t)+\\
&\frac{1}{2}\ln(1-q)+\frac{q-R^2}{2(1-q)}\Bigg\},
\end{aligned}
\tag{A.24}
$$

which yields equations (3.10) and (3.26).

Appendix B

Stability of the RS solution

Here, we show that our RS solution is unstable for $\alpha > \alpha_c$. The Almeida-Thouless (AT) stability for the RS solution is judged by the following quantity for any temperature [62],

$$\Lambda_3 = \alpha\lambda_3\tilde{\lambda_3} - 1, \tag{B.1}$$

where

$$\lambda_3 \equiv \frac{2}{q^2}\int Dt\,\Omega\left(\frac{R}{\sqrt{q}} : t\right)\left[\frac{\partial^2}{\partial t^2}\ln \Xi_\beta(q:t)\right]^2, \tag{B.2}$$

$$\tilde{\lambda_3} \equiv (1-q)^2. \tag{B.3}$$

From eq. (3.31), we obtain the following relation for the RS solution

$$\ln \Xi_\beta(q:t) \sim \begin{cases} -\beta, & \text{for } t < -\sqrt{2x} \\ -\frac{\beta}{2x}t^2, & \text{for } -\sqrt{2x} < t < 0 \\ 0, & \text{for } 0 < t \end{cases} \tag{B.4}$$

where $x = \beta(1-q)$, when β is large and $1-q$ is small. This yields the relation

$$\left[\frac{\partial^2}{\partial t^2}\ln \Xi_\beta(q:t)\right]^2 \sim \beta\left[\sqrt{\frac{2}{x}}\delta(t+\sqrt{2x}) - \frac{1}{x}\Big(\Theta(t+\sqrt{2x}) - \Theta(t)\Big)\right], \tag{B.5}$$

which means that Λ_3 is calculated as

$$\Lambda_3 = 2\alpha x^2\int \Omega(R:t)\left[\sqrt{\frac{2}{x}}\delta(t+\sqrt{2x}) - \frac{1}{x}\Big(\Theta(t+\sqrt{2x}) - \Theta(t)\Big)\right]^2 - 1. \tag{B.6}$$

in the limit $\beta \to \infty$. This diverges to infinity unless x is infinite because the right hand side of eq. (B.6) includes a term such as $\delta^2(t+\sqrt{2x})$ in the integral. For $\alpha > \alpha_c$, x of our RS solution is finite, which means that this solution is thermodynamically unstable.

We should mention that Λ_3 becomes 0 at $\alpha = \alpha_c$. Namely, the RS solution loses the AT stability *just at* α_c. This is explained as follows. By partially integrating the left hand side of equation (3.23), we obtain

$$2\alpha_c\int_{-\infty}^{0} Dt\,[\Omega(R_c:t) + t\,\Omega_t(R_c:t)] = 1 - {R_c}^2. \tag{B.7}$$

Adding equation (3.22) to this equation, we obtain the following relation at $\alpha = \alpha_c$

$$2\alpha_c\int_{-\infty}^{0} Dt\,\Omega(R_c:t) = 1. \tag{B.8}$$

Note that $x = \infty$ at $\alpha = \alpha_c$. Then, the right hand side of equation (B.6) becomes

$$2\alpha_c \int_{-\infty}^{0} Dt\, \Omega(R_c : t) - 1. \tag{B.9}$$

From equation (B.8), this means that $\Lambda_3 = 0$ at $\alpha = \alpha_c$.

Appendix C

Derivation of the differential equations of the on-line dynamics

In this appendix, we explain the derivation of differential equations with respect to l and R from the on-line learning dynamics which we discussed in chapter 4. In order to treat about the general weight function, we use $f(T_a(v), u)$ as the weight function. We should notice that the conventional perceptron, Hebbian and AdaTron learnings are recovered if we put

$$f(T_a(v), u) = -\Theta(-T_a(v)S(u))S(u) \tag{C.1}$$
$$f(T_a(v), u) = T_a(v) \tag{C.2}$$
$$f(T_a(v), u) = -u\Theta(-T_a(v)S(v)) \tag{C.3}$$

For this generalized weight function, the dynamics of the on-line learning is described as follows

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + f(T_a(v), u)\boldsymbol{x} \tag{C.4}$$

We first derive the differential equation with respect to l. Squaring the dynamics (C.4), we obtain

$$|\boldsymbol{J}^{m+1}|^2 = |\boldsymbol{J}^m|^2 + f(T_a(v), u)(\boldsymbol{J}^m \cdot \boldsymbol{x}) + f(T_a(v), u)^2. \tag{C.5}$$

Using the notations $l^m = |\boldsymbol{J}^m|/\sqrt{N}$ and $u_m = \sqrt{N}(\boldsymbol{J}^m \cdot \boldsymbol{x})/|\boldsymbol{J}^m|$, we rewrite equation (C.5) as

$$N(l^{m+1})^2 - N(l^m)^2 = \ll f(T_a(v), u)u_m l^m \gg + \ll f(T_a(v), u)^2 \gg \tag{C.6}$$

where $\ll \cdots \gg$ means the average over the distribution (4.4). We next rewrite the above equation (C.6) using the relations $(l^{m+1})^2 - (l^m)^2 = (l^{m+1} + l^m)(l^{m+1} - l^m) \simeq 2ldl$ and $1/N \simeq d\alpha$ in the limit of $N \to \infty$ as

$$\frac{dl}{d\alpha} = \frac{1}{2l}\left[\ll f(T_a(v), u)u \gg l + \ll f(T_a(v), u)^2 \gg\right] \tag{C.7}$$

Next we derive the differential equation with respect to R. Making the product between $\boldsymbol{B}$ and each side of the on-line dynamics (C.4), we obtain

$$\boldsymbol{J}^{m+1} \cdot \boldsymbol{B} = \boldsymbol{J}^m \cdot \boldsymbol{B} + f(T_a(v), u)\boldsymbol{B} \cdot \boldsymbol{x} \tag{C.8}$$

Using the relation $\boldsymbol{B} \cdot \boldsymbol{J}^m = Nl^m R^m$ and $v = \boldsymbol{B} \cdot \boldsymbol{x}$, we can rewrite the above equation (C.8) as follows

$$Nl^{m+1}R^{m+1} = Nl^m R^m + \ll f(T_a(v), u)v \gg \tag{C.9}$$

By the relations $l^{m+1}R^{m+1} - l^m R^m \simeq Rdl + ldR$ and $1/N \simeq d\alpha$, the above equation (C.9) is rewritten as

$$R\frac{dl}{d\alpha} + l\frac{dR}{d\alpha} = \ll f(T_a(v), u)v \gg. \tag{C.10}$$

Substituting the result (C.7), we finally obtain the following differential equation with respect to R

$$\frac{dR}{d\alpha} = \frac{1}{l}\left[\ll f(T_a(v), u) \gg - \frac{R}{2l}\left[\ll f(T_a(v), u)u \gg l + \ll f(T_a(v), u)^2 \gg\right]\right]. \tag{C.11}$$

Appendix D

Integrals

In this appendix, we listed calculations of the integrals appearing in the differential equations of the on-line learning. We first consider the structural mismatch case, namely, a simple perceptron learns from a non-monotonic perceptron.

1. $\ll uT_a(v) \gg$

$$\begin{aligned} \ll uT_a(v) \gg &= \int_{-\infty}^{\infty} du\, u \int_{-\infty}^{-a} dv\, P_R(u,v) + \int_{-\infty}^{\infty} du\, u \int_{0}^{a} dv\, P_R(u,v) \\ &- \int_{-\infty}^{\infty} du\, u \int_{-a}^{0} dv\, P_R(u,v) - \int_{-\infty}^{\infty} du\, u \int_{a}^{\infty} dv\, P_R(u,v) \\ &= \frac{2R}{\sqrt{2\pi}}(1-2\Delta) \end{aligned} \tag{D.1}$$

where $P_R(u,v)$ is the distribution we introduced in (4.4).

Similar for

$$\ll vT_a(v) \gg = \frac{2}{\sqrt{2\pi}}(1-2\Delta) \tag{D.2}$$

2. $\ll uS(u)\Theta(-uT_a(v)) \gg$
See Figure (D.1) for the integral regions.

$$\begin{aligned} I_1 &= -\int_{-\infty}^{0} du \int_{-\infty}^{-a} dv\, \frac{u}{2\pi\sigma} \exp\left[-\frac{u^2+v^2-2Ruv}{2\sigma^2}\right] \quad (\sigma^2 \equiv 1-R^2) \\ &= -\int_{-\infty}^{-a} dv \int_{-\infty}^{-\frac{Rv}{\sigma}} \frac{du}{2\pi}(-\sigma u + Rv) \mathrm{e}^{-\frac{u^2+v^2}{2}} \\ &= \int_{-\infty}^{-a} dv \int_{\frac{Rv}{\sigma}}^{\infty} \frac{du}{2\pi}(\sigma u - Rv) \mathrm{e}^{-\frac{u^2+v^2}{2}} \\ &= -\frac{R}{2\sqrt{2\pi}} \int_{-\infty}^{-a} v dv \mathrm{e}^{-\frac{v^2}{2}} \operatorname{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right) + \frac{\sigma}{2\pi} \int_{-\infty}^{-a} dv\, \mathrm{e}^{-\frac{v^2}{2\sigma^2}} \end{aligned} \tag{D.3}$$

The first integral in the above expression is

$$\begin{aligned} &- \frac{R}{2\sqrt{2\pi}} \left\{ \left[-\mathrm{e}^{-\frac{v^2}{2}} \operatorname{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right) \right]_{-\infty}^{-a} - \frac{2}{\sqrt{\pi}} \int_{-\infty}^{-a} dv\, \mathrm{e}^{-\frac{v^2}{2}} \frac{R}{\sqrt{2\pi}} \mathrm{e}^{-\frac{R^2v^2}{2\sigma^2}} \right\} \\ &= \frac{R}{2\sqrt{2\pi}} \mathrm{e}^{-\frac{a^2}{2}} \operatorname{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) + \frac{R^2}{2\pi\sigma} \int_{-\infty}^{-a} dv\, \mathrm{e}^{-\frac{v^2}{2\sigma^2}}. \end{aligned} \tag{D.4}$$

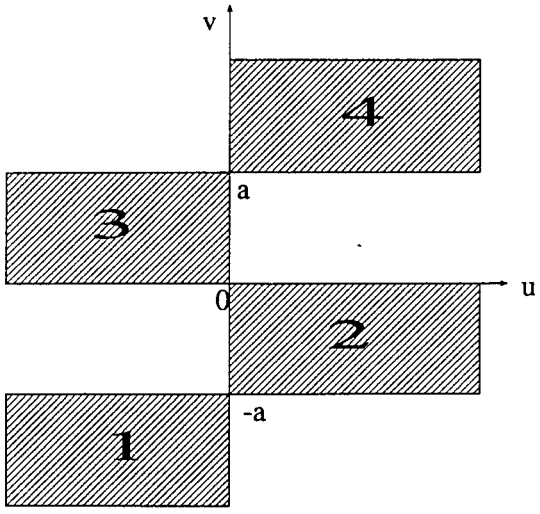

Figure D.1: Integral regions for the mismatch case.

Therefore, the integral in the region 1 is

$$I_1 = \frac{R}{2\sqrt{2\pi}} e^{-\frac{a^2}{2}} \operatorname{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) + \frac{1}{2\pi\sigma}\int_{-\infty}^{-a} dv\, e^{-\frac{v^2}{2\sigma^2}} \tag{D.5}$$

Similarly,

$$I_2 = -\frac{R}{2\sqrt{2\pi}}\left\{1 - e^{-\frac{a^2}{2}} \operatorname{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right)\right\} + \frac{1}{2\pi\sigma}\int_{0}^{a} dv\, e^{-\frac{v^2}{2\sigma^2}} \tag{D.6}$$

$$I_3 = -\frac{R}{2\sqrt{2\pi}}\left\{1 - e^{-\frac{a^2}{2}} \operatorname{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right)\right\} + \frac{1}{2\pi\sigma}\int_{-a}^{0} dv\, e^{-\frac{v^2}{2\sigma^2}} \tag{D.7}$$

$$I_4 = -\frac{R}{2\sqrt{2\pi}} e^{-\frac{a^2}{2}} \operatorname{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right) + \frac{1}{2\pi\sigma}\int_{a}^{\infty} dv\, e^{-\frac{v^2}{2\sigma^2}} \tag{D.8}$$

$$\tag{D.9}$$

The total makes

$$-\frac{R}{\sqrt{2\pi}}(1-2\Delta) + \frac{1}{\sqrt{2\pi}}. \tag{D.10}$$

3. $\ll vS(u)\Theta(-uT_a(v))\gg$

$$\begin{aligned} I_1 &= \int_{-\infty}^{0} du \int_{-\infty}^{-a} v dv \frac{-1}{2\pi\sigma} e^{-\frac{(u-Rv)^2}{2\sigma^2}} e^{-\frac{v^2}{2}} \\ &= -\int_{-\infty}^{-a} \frac{v dv}{\sqrt{2\pi}} e^{-\frac{v^2}{2}} \frac{1}{2} \operatorname{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right) \\ &= -\frac{1}{2\sqrt{2\pi}}\left\{\left[e^{-\frac{v^2}{2}} \operatorname{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right)\right]_{-\infty}^{-a} - \int_{-\infty}^{-a} dv e^{-\frac{v^2}{2}} \frac{2R}{\sqrt{2\pi}\sigma} \exp\left(-\frac{R^2v^2}{2\sigma^2}\right)\right\} \\ &= \frac{1}{2\sqrt{2\pi}} e^{-\frac{a^2}{2}} \operatorname{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) + \frac{R}{2\pi\sigma}\int_{-\infty}^{-a} dv e^{-\frac{v^2}{2\sigma^2}} \end{aligned} \tag{D.11}$$

Similarly,

$$I_2 = \frac{1}{2\sqrt{2\pi}}\left\{e^{-\frac{a^2}{2}} \operatorname{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right) - 1\right\} + \frac{R}{2\pi\sigma}\int_{0}^{a} dv e^{-\frac{v^2}{2\sigma^2}}. \tag{D.12}$$

From symmetry of the integrand under the change of sign os u and v, we find $I_3 = I_2$ and $I_4 = I_1$. Summing up those results, we have the final value

$$-\frac{1}{\sqrt{2\pi}}\left(1-2\mathrm{e}^{-\frac{a^2}{2}}\right)+\frac{R}{\sqrt{2\pi}}. \tag{D.13}$$

4. $\ll u^2\Theta(-uT_a(v))\gg$

$$\begin{aligned} I_1 &= \frac{1}{2\pi\sigma}\int_{-\infty}^{0}u^2du\int_{-\infty}^{-a}dv\exp\left[-\frac{(v-Ru)^2}{2\sigma^2}\right]\mathrm{e}^{-\frac{u^2}{2}} \\ &= \frac{1}{2\sqrt{2\pi}}\int_{-\infty}^{0}u^2du\mathrm{e}^{-\frac{u^2}{2}}\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) \end{aligned} \tag{D.14}$$

$$I_2 = \frac{1}{2\sqrt{2\pi}}\int_{0}^{\infty}u^2du\mathrm{e}^{-\frac{u^2}{2}}\left\{\mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right)-\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)\right\} \tag{D.15}$$

$$I_3 = I_2 \tag{D.16}$$

$$I_4 = I_1 \tag{D.17}$$

The total is

$$\frac{1}{\sqrt{2\pi}}\int_{0}^{\infty}u^2du\,\mathrm{e}^{-\frac{u^2}{2}}\left\{\mathrm{erfc}\left(\frac{a-Ru}{\sqrt{2}\sigma}\right)+\mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right)-\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)\right\}. \tag{D.18}$$

5. $\ll uv\Theta(-uT_a(v))\gg$

$$\begin{aligned} I_1 &= \frac{1}{2\pi\sigma}\int_{-\infty}^{0}udu\int_{-\infty}^{-a}vdv\exp\left(-\frac{(v-Ru)^2}{2\sigma^2}\right)\mathrm{e}^{-\frac{u^2}{2}} \\ &= \frac{1}{2\pi}\int_{-\infty}^{0}udue^{-\frac{u^2}{2}}\int_{-\infty}^{-\frac{a+Ru}{\sigma}}(\sigma t+Ru)\mathrm{e}^{-\frac{t^2}{2}}dt \\ &= \frac{1}{2\pi}\int_{-\infty}^{0}udue^{-\frac{u^2}{2}}\left\{\sigma\left[-\mathrm{e}^{-\frac{t^2}{2}}\right]_{-\infty}^{-\frac{a+Ru}{\sigma}}+Ru\int_{\frac{a+Ru}{\sqrt{2}\sigma}}^{\infty}\sqrt{2}dt\mathrm{e}^{-t^2}\right\} \\ &= -\frac{\sigma}{2\pi}\int_{-\infty}^{0}udu\exp\left(-\frac{a^2+u^2+2au}{2\sigma^2}\right) \\ &+ \frac{R}{2\sqrt{2\pi}}\int_{-\infty}^{0}u^2du\mathrm{e}^{-\frac{u^2}{2}}\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) \\ &= -\frac{\sigma}{2\pi}\mathrm{e}^{-\frac{a^2}{2\sigma^2}}\int_{-\infty}^{0}udu\exp\left(-\frac{(u+aR)^2}{2\sigma^2}+\frac{a^2R^2}{2\sigma^2}\right) \\ &+ \frac{R}{2\sqrt{2\pi}}\int_{0}^{\infty}u^2du\mathrm{e}^{-\frac{u^2}{2}}\mathrm{erfc}\left(\frac{a-Ru}{\sqrt{2}\sigma}\right). \end{aligned} \tag{D.19}$$

The first term in the above expression is

$$\begin{aligned} &-\frac{\sigma}{2\pi}\mathrm{e}^{-\frac{a^2}{2}}\int_{-\infty}^{aR}(t-aR)dt\mathrm{e}^{-\frac{t^2}{2\sigma^2}} \\ &= -\frac{\sigma}{2\pi}\mathrm{e}^{-\frac{a^2}{2}}\left\{\left[-\sigma^2\mathrm{e}^{-\frac{t^2}{2\sigma^2}}\right]_{-\infty}^{aR}-aR\int_{-\frac{aR}{\sqrt{2}\sigma}}^{\infty}\sqrt{2}\sigma dt\mathrm{e}^{-t^2}\right\} \\ &= \frac{\sigma^3}{2\pi}\mathrm{e}^{-\frac{a^2}{2\sigma^2}}+\frac{aR\sigma^2}{2\sqrt{2\pi}}\mathrm{e}^{-\frac{a^2}{2}}\mathrm{erfc}\left(-\frac{aR}{\sqrt{2}\sigma}\right). \end{aligned} \tag{D.20}$$

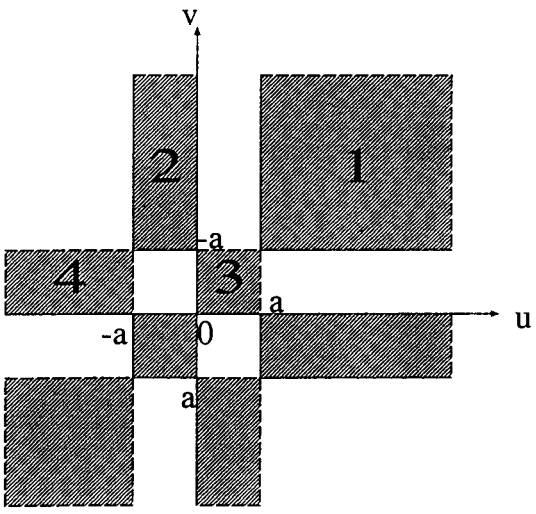

Figure D.2: Integral regions for the realizable case.

Similarly,

$$I_2 = \frac{\sigma^3}{2\pi}\left(e^{-\frac{a^2}{2\sigma^2}} - 1\right) - \frac{aR\sigma^2}{2\sqrt{2\pi}}e^{-\frac{a^2}{2}}\mathrm{erfc}\left(\frac{aR}{\sqrt{2}\sigma}\right) \tag{D.21}$$

$$I_3 = I_2 \tag{D.22}$$

$$I_4 = I_1 \tag{D.23}$$

The total is

$$\begin{aligned} &\frac{\sigma^3}{\pi}\left(2e^{-\frac{a^2}{2\sigma^2}} - 1\right) + \frac{2aR\sigma^2}{\sqrt{2\pi}}e^{-\frac{a^2}{2}}\left(1 - \mathrm{erfc}\left(\frac{aR}{\sqrt{2}\sigma}\right)\right) \\ + &\frac{R}{\sqrt{2\pi}}\int_0^\infty u^2 du e^{-\frac{u^2}{2}}\left\{\mathrm{erfc}\left(\frac{a-Ru}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) + \mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right)\right\} \end{aligned} \tag{D.24}$$

Next we consider the case in which a non-monotonic perceptron learns from a non-monotonic perceptron.

1. The asymptotic form of the generalization error $E(R)$ for the case of $R = 1 - \varepsilon$ and $\varepsilon \to 0$.

$$\begin{aligned} E(R) &= \int_a^\infty \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a+Rv}{\sqrt{2}\sigma}\right) + \int_a^\infty \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(-\frac{(a-Rv)}{\sqrt{2}\sigma}\right) \\ &+ \int_0^a \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right) - \int_a^\infty \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(\frac{Rv}{\sqrt{2}\sigma}\right) \\ &- \int_0^a \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a+Rv}{\sqrt{2}\sigma}\right) + \int_0^a \frac{dv}{\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a-Rv}{\sqrt{2}\sigma}\right) \\ &\equiv E_1 + E_2 + E_3 + E_4 + E_5 + E_6. \end{aligned} \tag{D.25}$$

The first term E_1 can be rewritten by the change of variable $(a + Rv)/\sqrt{2}\sigma \sim (a + v)/2\sqrt{\varepsilon} = t$ as

$$\begin{aligned} E_1 &= \sqrt{\frac{2\varepsilon}{\pi}}\int_{\frac{a}{\sqrt{\varepsilon}}}^\infty dt\, e^{-\frac{1}{2}(a+2\sqrt{\varepsilon}t)^2}\mathrm{erfc}(t) \\ &\sim \sqrt{\frac{2\varepsilon}{\pi}}e^{-\frac{a^2}{2}}\int_{\frac{a}{\varepsilon}}^\infty dt(1 - 2a\sqrt{\varepsilon}t)\mathrm{erfc}(t) \end{aligned}$$

$$= \sqrt{\frac{2\varepsilon}{\pi}}\mathrm{e}^{-\frac{a^2}{2}}\left\{\int_{\frac{a}{\sqrt{\varepsilon}}}^{\infty} dt(t)'\mathrm{erfc}(t) - 2a\sqrt{\varepsilon}\int_{\frac{a}{\varepsilon}}^{\infty} dt\,(\frac{t^2}{2})'\mathrm{erfc}(t)\right\} = 0 \tag{D.26}$$

Similarly,

$$E_2 = \frac{\sqrt{2\varepsilon}}{\pi}\mathrm{e}^{-\frac{a^2}{2}}(1+a\sqrt{\varepsilon}) \tag{D.27}$$

$$E_3 = 0 \tag{D.28}$$

$$E_4 = \frac{\sqrt{2\varepsilon}}{\pi} \tag{D.29}$$

$$E_5 = 0 \tag{D.30}$$

$$E_6 = \frac{\sqrt{2\varepsilon}}{\pi}\mathrm{e}^{-\frac{a^2}{2}}(1-a\sqrt{\varepsilon}) \tag{D.31}$$

The total makes

$$E(\varepsilon) \sim \frac{\sqrt{2\varepsilon}}{\pi}(1+2\Delta). \tag{D.32}$$

2. $\ll uS_a(u)\Theta(-uT_a(v))\gg$
See. Figure D.2 for the integration regions.

$$\begin{aligned} I_1 &= \int_{-\infty}^{-a}\frac{dv}{\sqrt{2\pi}}\mathrm{e}^{-\frac{v^2}{2}}\int_{\frac{a-Rv}{\sigma}}^{\infty}(\sigma u+Rv)\mathrm{e}^{-\frac{u^2}{2}} \\ &= \frac{\sigma}{\sqrt{2\pi}}\int_{-\infty}^{-a}\frac{dv}{\sqrt{2\pi}}\mathrm{e}^{-\frac{v^2}{2}}[-\mathrm{e}^{-\frac{v^2}{2}}]_{\frac{a-Rv}{\sigma}}^{\infty} + \frac{R}{2\sqrt{2\pi}}\int_{-\infty}^{-a}\frac{vdv}{\sqrt{2\pi}}\mathrm{e}^{-\frac{v^2}{2}}\mathrm{erfc}\left(\frac{a-Rv}{\sqrt{2}\sigma}\right) \\ &= \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sigma}a\right) + \frac{R}{2\sqrt{2\pi}}\int_{-\infty}^{-a}\frac{dv}{\sqrt{2\pi}}(-\mathrm{e}^{-\frac{v^2}{2}})'\mathrm{erfc}\left(\frac{a-Rv}{\sqrt{2}\sigma}\right) \\ &= \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right) - \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right) + \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right). \end{aligned} \tag{D.33}$$

Using the same technique as the above calculation,

$$\begin{aligned} I_2 &= \frac{\sigma^2}{2\sqrt{2\pi}} - \frac{\sigma^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) - \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{1+R}{\sqrt{2}\sigma}a\right) \\ &+ \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) - \frac{R}{2\sqrt{2\pi}} + \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right) \\ &+ \frac{R^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(-\frac{a}{\sqrt{2}\sigma}\right) - \frac{R^2}{2\sqrt{2\pi}} - \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right) \\ &+ \frac{R}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) - \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{1+R}{\sqrt{2}\sigma}a\right) + \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right). \end{aligned} \tag{D.34}$$

$$\begin{aligned} I_3 &= \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) - \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) \\ &+ \frac{R}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) - \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) \\ &+ \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) - \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) \end{aligned} \tag{D.35}$$

$$\begin{aligned} I_4 &= \frac{\sigma^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) - \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) \\ &+ \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) + \frac{R^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) \\ &- \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) - \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right) \end{aligned} \tag{D.36}$$

Therefore, the total is given as $\ll uS_a\Theta(-uT_a(v))\gg = 2(I_1+I_2+I_3+I_4)$, namely,

$$\frac{(1-R)}{\sqrt{2\pi}}(1-2\Delta). \tag{D.37}$$

3. $\ll vS_a(u)\Theta(-uT_a(v))\gg$

$$\begin{aligned}
I_1 &= \int_a^\infty \frac{dv}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}\int_{-\infty}^{-\frac{a+Rv}{\sigma}}\frac{dv}{\sqrt{2\pi}}(\sigma v+Ru)e^{-\frac{v^2}{2}} \\
&= \frac{\sigma}{\sqrt{2\pi}}\int_a^\infty \frac{dv}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}\left[-e^{-\frac{v^2}{2}}\right]_{-\infty}^{-\frac{a+Ru}{\sigma}}+\frac{R}{\sqrt{2\pi}}\int_a^\infty\frac{dv}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}u\int_{-\infty}^{-\frac{a+Ru}{\sigma}}e^{-\frac{v^2}{2}}dv \\
&= -\frac{\sigma}{\sqrt{2\pi}}\int_a^\infty\frac{dv}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}e^{-\frac{(a+Ru)^2}{2\sigma^2}}+\frac{R}{2}\int_a^\infty\frac{udu}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) \\
&= -\frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right)+\frac{R}{2\sqrt{2}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right) \\
&- \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}\right)
\end{aligned} \tag{D.38}$$

Similarly,

$$\begin{aligned}
I_2 &= \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{Ra}{\sqrt{2}\sigma}\right)-\frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right)-\frac{\sigma^2}{2\sqrt{2\pi}} \\
&+ \frac{\sigma^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right)+\frac{R}{2\sqrt{2\pi}}\mathrm{erfc}\left(-\frac{a}{\sqrt{2}\sigma}\right)-\frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{1+R}{\sqrt{2}\sigma}a\right) \\
&+ \frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{R}{\sqrt{2}\sigma}a\right)-\frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1+R}{\sqrt{2}\sigma}a\right)-\frac{R}{\sqrt{2\pi}} \\
&+ \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{R}{\sqrt{2}\sigma}a\right)-\frac{R^2}{2\sqrt{2\pi}}+\frac{R^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right)
\end{aligned} \tag{D.39}$$

$$\begin{aligned}
I_3 &= \frac{\sigma^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right)-\frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)+\frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right) \\
&+ \frac{R^2}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right)-\frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)-\frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)
\end{aligned} \tag{D.40}$$

$$\begin{aligned}
I_4 &= \frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right)-\frac{\sigma^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)+\frac{R}{2\sqrt{2\pi}}\mathrm{erfc}\left(\frac{a}{\sqrt{2}\sigma}\right) \\
&- \frac{R}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)+\frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(-\frac{Ra}{\sqrt{2}\sigma}\right)-\frac{R^2}{2\sqrt{2\pi}}\Delta\mathrm{erfc}\left(\frac{1-R}{\sqrt{2}\sigma}a\right)
\end{aligned} \tag{D.41}$$

Therefore, the total is given as $\ll vS_a\Theta(-uT_a(v))\gg = 2(-I_1+I_2-I_3+I_4)$, namely,

$$\ll vS_a(u)\Theta(-uT_a(v))\gg = -\frac{(1-R)}{\sqrt{2\pi}}(1-2\Delta). \tag{D.42}$$

4. $\ll u^2\Theta(-uT_a(v))\gg$

$$\begin{aligned}
\mathcal{I}_1 &= \int_a^\infty u^2du\int_{-\infty}^{-a}dv\frac{1}{2\pi\sigma}e^{-\frac{u^2}{2}}e^{-\frac{(v-Ru)^2}{2\sigma^2}} \\
&= \frac{1}{2}\int_a^\infty\frac{u^2du}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}\mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)
\end{aligned} \tag{D.43}$$

Similarly,

$$\mathcal{I}_2 = \frac{1}{2}\int_{-a}^{0} \frac{u^2 du}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) \tag{D.44}$$

$$\mathcal{I}_3 = \frac{1}{2}\int_{0}^{a} \frac{u^2 du}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \left[\mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)\right] \tag{D.45}$$

$$\mathcal{I}_4 = \frac{1}{2}\int_{-\infty}^{-a} \frac{u^2 du}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \left[\mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)\right] \tag{D.46}$$

The total makes

$$\begin{aligned} \ll u^2\Theta(-uT_a(v))\gg &= \left(\int_a^{\infty} + \int_{-a}^{0}\right) \frac{u^2 du}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right) \\ &+ \left(\int_0^{a} + \int_{-\infty}^{-a}\right) \frac{u^2 du}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \left[\mathrm{erfc}\left(\frac{Ru}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a+Ru}{\sqrt{2}\sigma}\right)\right] \end{aligned} \tag{D.47}$$

5. $\ll uv\Theta(-uT_a(v))\gg$

$$\begin{aligned} I_1 &= \int_a^{\infty} \frac{udu}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \int_{-\infty}^{-a} \frac{vdv}{\sqrt{2\pi}\sigma} e^{-\frac{(v-Ru)^2}{2\sigma^2}} \\ &= \frac{udu}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} (Ru) \int_{-\infty}^{-\frac{a+Ru}{\sigma}} \frac{dt}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} + \sigma \int_a^{\infty} \frac{udv}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \int_{-\infty}^{-\frac{(a+Ru)}{\sigma}} \frac{tdt}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} \\ &= R\mathcal{I}_1 + \frac{\Delta a R\sigma^2}{2\sqrt{2\pi}} \mathrm{erfc}\left(\frac{a(1+R)}{\sqrt{2}\sigma}\right) - \frac{\sigma^3\Delta}{2\pi} e^{-\frac{a^2(1+R)^2}{2\sigma^2}} \end{aligned} \tag{D.48}$$

Using the same technique as the above calculations,

$$\begin{aligned} I_2 &= R\mathcal{I}_2 + \frac{\Delta a R\sigma^2}{2\sqrt{2\pi}} \left[\mathrm{erfc}\left(-\frac{-a(1-R)}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{aR}{\sqrt{2}\sigma}\right)\right] \\ &- \frac{\sigma^3\Delta}{2\pi} \left[e^{-\frac{a^2(1+R)^2}{2\sigma^2}} - e^{-\frac{a^2R^2}{2\sigma^2}}\right] \end{aligned} \tag{D.49}$$

$$\begin{aligned} I_3 &= R\mathcal{I}_3 + \frac{\sigma^3}{2\pi} \left[\Delta e^{-\frac{a^2R^2}{2\sigma^2}} - \Delta e^{-\frac{a^2(1+R)^2}{2\sigma^2}} - 1 + e^{-\frac{a^2}{2\sigma^2}}\right] \\ &- \frac{\Delta a R\sigma^2}{2\sqrt{2\pi}} \left[\mathrm{erfc}\left(\frac{aR}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a(1+R)}{\sqrt{2}\sigma}\right)\right] \end{aligned} \tag{D.50}$$

$$\begin{aligned} I_4 &= R\mathcal{I}_4 + \frac{\sigma^3}{\sqrt{2\pi}} \left[e^{-\frac{a^2}{2\sigma^2}} - \Delta e^{-\frac{a^2(1-R)^2}{2\sigma^2}}\right] \\ &- \frac{\Delta a R\sigma^2}{2\sqrt{2\pi}} \mathrm{erfc}\left(\frac{a(1-R)}{\sqrt{2}\sigma}\right) \end{aligned} \tag{D.51}$$

The total makes

$$\begin{aligned} \ll uv\Theta(-uT_a(v))\gg &= R\ll u^2\Theta(-uT_a(v))\gg \\ &+ \frac{4\Delta a R\sigma^2}{2\sqrt{2\pi}} \left[\mathrm{erfc}\left(\frac{a(1+R)}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{aR}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(\frac{a(1-R)}{\sqrt{2}\sigma}\right) + \frac{1}{2}\right] \\ &+ \frac{2\sigma^3}{\pi} \left[\Delta e^{-\frac{a^2R^2}{2\sigma^2}} - \Delta e^{-\frac{a^2(1+R)^2}{2\sigma^2}} - \Delta e^{-\frac{a^2(1-R)^2}{2\sigma^2}} + e^{-\frac{a^2}{2\sigma^2}} - \frac{1}{2}\right] \end{aligned} \tag{D.52}$$

Appendix E

The weight function in the modified AdaTron learning algorithm

In this appendix, we explain how we introduced the modified weight function $\Theta(-T_a(v)S_a(u))h(u)l$ appearing in the AdaTron learning algorithm in subsection 5.3.2. From equations (5.77) and (5.84) in subsection 5.3.2, the weight function using the Bayes formula is written as

$$< f^* >= \frac{1-R^2}{R^2}\, l\, \frac{\partial}{\partial u} \log \Omega\,(y|u). \tag{E.1}$$

As this expression contains the unknown parameter R to the student, we try to find the suitable learning weight function which agrees with the asymptotic form of $< f^* >$ in the limit of $R \to 1$ [98]. For this purpose, we investigate the asymptotic form of $\Omega\,(y|u)$ as follows. We consider the cases of $T_a \equiv y = 1$ and $y = -1$ separately.

(I) $y = 1$
Using the relation $R = 1 - \varepsilon, \varepsilon \to 0$, we find

$$\begin{aligned}\Omega(y|u) &= H\left(-\frac{Ru}{\sqrt{1-R^2}}\right) - H\left(\frac{a-Ru}{\sqrt{1-R^2}}\right) + H\left(\frac{a+Ru}{\sqrt{1-R^2}}\right) \\ &\simeq \frac{1}{\sqrt{\pi}}\left[\mathrm{erfc}\left(\frac{-u}{2\sqrt{\varepsilon}}\right) - \mathrm{erfc}\left(\frac{a-u}{2\sqrt{\varepsilon}}\right) + \mathrm{erfc}\left(\frac{a+u}{2\sqrt{\varepsilon}}\right)\right].\end{aligned} \tag{E.2}$$

The asymptotic form of $\Omega\,(y|u)$ depends on the range of u. For $u > a$, the asymptotic form of $\Omega\,(y|u)$ is

$$\Omega \sim \frac{1}{u-a}\sqrt{\frac{\varepsilon}{\pi}} \exp\left(-\frac{(u-a)^2}{4\varepsilon}\right). \tag{E.3}$$

Therefore, $< f^* > /l = -(u-a)$. Similarly, we find $< f^* > /l = 0$ ($0 < u < a$ and $u < -a$), $< f^* > /l = -u$ ($-a/2 < u < 0$) and $< f^* > /l = -(u+a)$ ($-a < u < -a/2$).
(II) $y = -1$
Using the relation $R = 1 - \varepsilon$, we find for $u > a$

$$\Omega \sim 1 - \frac{1}{u-a}\sqrt{\frac{\varepsilon}{\pi}} \exp\left(-\frac{(u-a)^2}{4\varepsilon}\right). \tag{E.4}$$

Therefore, the weight function $< f^* > /l$ is 0 asymptotically. Similarly, we find $< f^* > /l = 0$ ($a/2 < u < a$ and $-a < u < 0$), $< f^* > /l = -u$ ($0 < u < a/2$) and $< f^* > /l = -(a+u)$ ($u < -a$).

From the results of (I) and (II), we find the modified AdaTron learning algorithm as

$$\boldsymbol{J}^{m+1} = \boldsymbol{J}^m + \Theta(-T_a(v)S_a(u))h(u)\, l\, \boldsymbol{x} \tag{E.5}$$

where

$$h(u) = \begin{cases} a - u & (u > \frac{a}{2}) \\ -u & (-\frac{a}{2} < u < \frac{a}{2}) \\ -a - u & (u < -\frac{a}{2}) \end{cases} \tag{E.6}$$

Appendix F

Derivation of the Fokker-Planck equation

Here we explain the derivation of the Fokker-Plank equation for the general transition probability in chapter 6. We first consider the master equation which describes the hopping between the nearest neighbor local minima;

$$\frac{dP_i}{dt} = \left(1+(q-1)\frac{B}{T}\right)^{1/(1-q)} P_{i+1} + \left(1+(q-1)\frac{B+\Delta_{i-1}}{T}\right)^{1/(1-q)} P_{i-1} - \left(1+(q-1)\frac{B+\Delta_i}{T}\right)^{1/(1-q)} P_i - \left(1+(q-1)\frac{B}{T}\right)^{1/(1-q)} P_i. \tag{F.1}$$

For simplicity, we put

$$A \equiv \left[1+(q-1)\frac{B}{T}\right]^{1/(q-1)}. \tag{F.2}$$

Then, the master equation (F.1) is rewritten as

$$\begin{aligned}\frac{dP_i}{dt} &= \frac{P_i}{A} + \left[\frac{1}{A} - \frac{(ax-\frac{a^2}{2})}{TA^q}\right] P_{i-1} - \left[\frac{1}{A} - \frac{(ax+\frac{a^2}{2})}{TA^q}\right] P_i - \frac{P_i}{A} \\ &= \frac{1}{A}(P_{i+1}-P_i) - \frac{1}{A}(P_i - P_{i-1}) + \frac{ax}{TA^q}P_i + \frac{a^2}{2TA^q}P_i - \frac{ax}{TA^q}P_{i-1} + \frac{a^2}{2TA^q}P_{i-1}. \end{aligned} \tag{F.3}$$

Introducing γ as

$$\gamma \equiv \frac{a^2}{TA^2}, \tag{F.4}$$

we can rewrite the first two terms on the right hand side of equation (F.3) in the limit $a \to \infty$ as

$$\begin{aligned}\frac{1}{A}[P_{i+1}-P_i] - \frac{1}{A}[P_i - P_{i-1}] &= T\gamma \left\{ \frac{\left[\frac{P_{i+1}-P_i}{a} - \frac{P_i - P_{i-1}}{a}\right]}{a} \right\} \\ &= T\gamma \frac{\partial^2 P(x)}{\partial x^2}. \end{aligned} \tag{F.5}$$

Then, using same technique as the above, the rest of the terms appearing in the right hand side of equation (F.3) lead to

$$\begin{aligned}-\frac{x\gamma}{aA^{q-1}}P_{i-1} + \frac{\gamma}{2A^{q-1}}P_{i-1} + \frac{x\gamma}{aA^{q-1}}P_i + \frac{\gamma}{2A^{q-1}}P_i &\simeq \frac{x\gamma}{A^{q-1}}\left\{\frac{P_i - P_{i-1}}{a}\right\} + \frac{\gamma}{A^{q-1}} \\ &= \frac{\gamma}{A^{q-1}}\frac{\partial(xP(x))}{\partial x}. \end{aligned} \tag{F.6}$$

As the result, the master equation (F.1) leads to the next Fokker-Plank equation;

$$\frac{dP}{dt} = \gamma(T)\frac{\partial}{\partial x}(xP) + D(T)\frac{\partial^2 P}{\partial x^2} \tag{F.7}$$

where $\gamma(T)$ and $D(T)$ are represented as

$$\gamma(T) = \frac{1}{T}\left(1 + (q-1)\frac{B}{T}\right)^{q/(1-q)} \tag{F.8}$$

and

$$D(T) = \left(1 + (q-1)\frac{B}{T}\right)^{1/(1-q)}. \tag{F.9}$$

Bibliography

[1] W. S. McCullock and W. Pitts, *Bulletin of Mathematical Biophysics* **5** 115 (1943).

[2] F. Rosenblatt, *Principles of Neurodynamics*, New York: Spartan (1962).

[3] S. Amari, *IEEE. Trans.*, **EC-16** 299 (1967).

[4] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Nature* **323** 533 (1986).

[5] P. J. Werbos, *IEEE Trans. on Systems, Man, and Cybernetics* **17** 7 (1987).

[6] D. B. Parker, In *IEEE First International Conference on Neural Networks, San Diego*, Eds. M. Caudill and C. Butler, **vol II** 593 (1987).

[7] T. J. Sejnowski and C. R. Rosenberg, *Complex Systems* **1** 145 (1987).

[8] G. E. Hinton and T. J. Sejnowski, *Parallel Distributed Processing* **vol. 1, chap. 7.** MIT press (1986).

[9] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, *Science* **220** 671 (1983).

[10] L. G. Valiant, *Comm. ACM.* **V27 N11** 1134 (1984).

[11] E. B. Baum and D. Haussler, *Neural Comp.* **1** 151 (1989).

[12] J. A. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, (Redwood City:Addison-Wesley 1991).

[13] V. N. Vapnik and A. YA. Chervonenkis, *Theory of Probability and Its Applications* **16** 264 (1971).

[14] T. M. Cover, *IEEE Trans. Electromagn. Comput.***14** 326 (1965).

[15] A. Engel and C. Van den Broeck, *Phys. Rev. Lett.* **71** 1772 (1993).

[16] A. Engel and W. Fink, *J. Phys. A: Math.Gen.* **26** 6893 (1993).

[17] Y. Kabashima and S. Shinomoto, *Neural Comp.* 4 712 (1992).

[18] Y. Kabashima and S. Shinomoto, *Neural Comp.* **7** 158 (1995).

[19] T. Uezu and Y. Kabashima, *J. Phys. A: Math. Gen.* **29** L55 (1996)

[20] T. Uezu, Y. Kabashima, K. Nokura and N. Nakamura, *J. Phys. Soc. Japan* **65** 3797 (1996).

[21] D. Saad and S. A. Solla, *Phys. Rev. E* **52** 4225 (1995).

[22] T. L. Watkin and A. Rau, *Phys. Rev. A* **45** 4111 (1992).

[23] E. Levin, N. Tishby and S. A. Solla, *Proc. of IEEE* **78** 1568 (1990).

[24] G. Györgyi and N. Tishby, *Statistical Theory of Learning a Rule* In Neural Networks and Spin Glasses, Eds. W. K. Theumann and R. Koeberle, Singapore: World Scientific (1990).

[25] H. S. Seung, H. Sompolinsky and N. Tishby, *Phys. Rev. A* **45** 6056 (1992).

[26] T. H. L. Watkin, A. Rau and M. Biehl, *Rev. Mod. Phys.* **65** 499 (1993).

[27] M. Opper and W. Kinzel, *in Physics of Neural Networks III*, Eds. E. Domany, J. L. van Hemmen and K. Schulten, Berlin: Springer (1995).

[28] G. Boffetta, R. Monasson and R. Zecchina, *J. Phys. A: Math. Gen.* **26** L507 (1993).

[29] R. Monasson and D. O'Kane, *Europhys. Lett.* **27** 85 (1994).

[30] R. Monasson, *PhD Thesis* L'Ecole Normale Superieure (1993).

[31] J. Inoue, *Master Thesis* Tokyo Institute of Technology (1995).

[32] M. Morita, S. Yoshizawa and K. Nakano, *Trans. IEICE* **J73-D-II** 242 (in Japanese) (1990).

[33] H. Nishimori and I. Opris, *Neural Networks* **6** 1061 (1993).

[34] M. Shiino and T. Fukai, *J. Phys. A: Math. Gen.* **26** L831 (1993).

[35] M. Shiino and T. Fukai, *Phys. Rev. E* **48** 876 (1993).

[36] T. Fukai, J. Kim and M. Shiino, *Neural Networks* **8** 391 (1995).

[37] J. Inoue, *J. Phys. A: Math. Gen.* **29** 4815 (1996).

[38] C. H. Papadimitriou, *Computational Complexity*, (Addison-Wesley 1994).

[39] S. Iwata, *Introduction to NP-complete problems*, (Kyoritsu Shyuppan 1995) (In Japansese).

[40] J. J. Hopfield and D. W. Tank, *Biological Cybernetics* **52** 141 (1985).

[41] J. J. Hopfield and D. W. Tank, *Science* **233** 625 (1986).

[42] M. Mézard and G. Parisi, *Journal de Physique (Paris)* **47** 1285 (1986).

[43] M. Mézard and G. Parisi, *Journal de Physique (Paris)* **49** 1285 (1988).

[44] M. Mézard, G. Parisi and M. A. Virasoro, *Spin Glass Theory and Beyond* (Singapore:World Scientific 1987).

[45] Y. Fu and P. W. Anderson, *J. Phys. A: Math. Gen.* **19** 1605 (1986).

[46] S. Geman and D. Geman, *IEEE Trans. Pattern Analysis and Machine Intelligence* **6** 721 (1984).

[47] D. Sherrington and S. Kirkpatrick, *Phys. Rev. Lett* **35** 1792 (1975).

[48] S. Kirkpatrick and D. Sherrington *Phys. Rev.* B **17** 4384 (1978).

[49] E. Korutcheva, M. Opper and L. López, *J. Phys. A: Math. Gen.* **27** L645 (1994).

[50] J. F. Fontanari, *J.Phys.A: Math.Gen.* **28** 4751 (1995).

[51] J. Inoue, *J. Phys.A: Math. Gen.* **30** 1047 (1997).

[52] J. R. de Almeida and D. J. Thouless *J. Phys. A: Math. Gen.* **11** 983 (1978).

[53] C. Tsallis, *J. Stat. Phys.***52** 479 (1988).

[54] C. Tsallis and D. A. Stariolo, Physica A **233**, 395 (1996).

[55] E. Gardner, *J. Phys. A: Math. Gen.* **21** 271 (1988).

[56] E. Gardner and B. Derrida, *J. Phys. A: Math. Gen.* **21** 271 (1988).

[57] R. Meir and J. F. Fontanari, *Phys. Rev. A* **45** 8874 (1992).

[58] R. O. Duda and P. E. Hert, *Pattern Classification and Scene Analysis* (Wiley, New York 1973).

[59] S. Agmon, *Can. J. Math.* **6** 382 (1954).

[60] J. K. Anlauf and M. Biehl, *Europhys. Lett.* **10** 687 (1989).

[61] B. Widrow, *Generalization and Information Storage in Networks of Adaline "Neurons"* In Self-Organizing Systems 1962 (Chicago 1962) 435-461, Eds. M. C. Yovits, G. T. Jaccobi and G. D. Goldstein, (Washington: Spartan 1962).

[62] M. Bouten, *J. Phys.A: Math.Gen.* **27** 6021 (1994).

[63] W. Whyte and D. Sherrington, *J. Phys. A: Math. Gen.* **29** 3036 (1996).

[64] D. M. Carlucci, *PhD Thesis*, Scuola Normal Superiore 1997.

[65] D. M. Carlucci, private communication.

[66] N. Tsukahara and M. Kwato, In *Computation and Cooperation in Neural Nets.* p-430 Eds. S. Amari and M. A. Arbib, Springer 1982.

[67] M. Kawato, M. Etoh, Y. Oda and N. Tukahara, *Biol. Cybern.* **53** 57 (1985).

[68] C. Jutten and J. Herault, *Signal Processing* **24** 1 (1991).

[69] P. Comon, *Signal Processing* **36** 287 (1994).

[70] S. Amari, A. Cichocki and H. H. Yang, In *Advances in Neural Information Processing System (NIPS)* **VIII**, Eds. D. Touretzky, M. Mozer and M. Haseelmo, p-757 MIT Press (1996).

[71] J. P. Nadal and N. Paraga, *Neural Comp.* **9** 1421 (1997).

[72] N. Murata, *PhD Thesis*, University of Tokyo 1992 (In Japanese).

[73] H. Robbins and S. Monro, *Ann. Math. Statist.* **22** 400 (1951).

[74] T. M. Heskes and B. Kappen, *Phys. Rev. A* **44** 2718 (1991).

[75] M. Opper and D. Haussler, *Proc. 4th ACM Workshop on Computational Learning Theory* (San Maetro:Morgan Kaufmann 1991).

[76] S. Amari, N. Fujita and S. Shinomoto, *Neural Comp.* **4** 605 (1992).

[77] A. Engel and L. Reimers, *Europhys. Lett.* **28** 531 (1994).

[78] M. Copelli and N. Caticha, *J. Phys. A: Math. Gen.* **28** 1615 (1995).

[79] D. Saad and S. A. Solla, *Phys. Rev. Lett.* **74** 4337 (1995).

[80] Y. Kabashima, *J. Phys. A: Math. Gen.* **27** 1917 (1994).

[81] N. Barkai, S. H. Seung and H. Sompolinsky, *Proc. of Advance in Neural Information Processing System (NIPS)* **VII** 303 (1995).

[82] M. Biehl, P. Reigler and M. Stechert, *Phys. Rev. E* **52** 4624 (1995).

[83] P. Reigler, M. Biehl, S. A. Solla and C. Marangi, *Proc. of Italian Workshop on Neural Nets VII* (1995).

[84] J. W. Kim and H. Sompolinsky, *Phys. Rev. Lett.* **76** 3021 (1996).

[85] H. Sompolinsky and J. W. Kim, *On-line Gibbs Learning I: General Theory*, Preprint (1997).

[86] J. W. Kim and H. Smpolinsky, *On-line Gibbs Learning II: Application to Perceptron and Multi-layer Networks*, Preprint (1997).

[87] E. B. Baum, *Neural Comp.* **2** 248 (1990).

[88] M. Biehl and H. Schwarze, *Europhys. Lett.* **20** 733 (1992).

[89] F. Vallet, *Europhys. Lett.* **9** 315 (1989).

[90] J. Inoue, H. Nishimori and Y. Kabashima, *J. Phys. A: Math. Gen.* **30** 3795 (1997).

[91] M. Opper, W. Kinzel, J. Kleinz and R. Nehl, *J. Phys. A: Math. Gen.* **23** L581 (1990).

[92] O. Kinouchi and N. Caticha, *J. Phys. A: Math. Gen.* **25** 6243 (1992).

[93] S. Shinomoto and Y. Kabashima, *J. Phys. A: Math. Gen.* **24** L141 (1991).

[94] H. Nishimori and J. Inoue, Submitted to *J. Phys. A: Math. Gen.*

[95] Y. Kabashima and J. Inoue, To appear in *J. Phys. A: Math. Gen.* **30** (1997).

[96] W. Kinzel and P. Ruján, *Europhys. Lett.* **13** 473 (1990).

[97] O. Kinouchi and N. Caticha, *J. Phys. A: Math. Gen.* **26** 6161 (1993).

[98] M. Biehl and M. Riegler, *Europhys. Lett.* **28** 525 (1994).

[99] J. Inoue and H. Nishimori, *Phys. Rev. E* **55** 4544 (1997).

[100] C. Van den Broeck, *Proc. of Theoretical Aspects of Neural Computation 97 (TANC-97)*, to be publised, Springer-Verlag.

[101] C. Van den Breock and P. Reimann, *Phys. Rev. Lett.* **76** 2188 (1996).

[102] R. Simonetti and N. Caticha, *J. Phys. A: Math. Gen.* **29** 4859 (1996).

[103] Y. Uesaka, *Mathematical Foundations of Neurocomputing* (Kindai Kagakusha, Tokyo 1993) (in Japanese).

[104] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* (Wiley, 1989) Chap. 3.

[105] H. Szu and R. Hartley, Phys. Lett. **122A**, 157 (1987).

[106] T. J. P. Penna, Phys. Rev. E**51**, R1 (1995).

[107] I. Andricioaei and J. E. Straub, *Phys. Rev, E* **53** R3055 (1996).