THE INDUCED TOPOLOGY OF LOCAL MINIMA WITH APPLICATIONS TO ARTIFICIAL NEURAL NETWORKS

By

Yun Chung CHU

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Information Engineering

The Chinese University of Hong Hong

May 1992

Acknowledgement

I would like to express my gratitude to Dr. C. P. Kwong for his supervision throughout these two years. In particular, with his encouragement it became possible to broaden my knowledge into many different areas, which is essential to the material presented in this thesis.

I also want to thank Dr. Edmund M. K. Lai for providing the raw speech data and a C program which computes the LPC coefficients, and Mr. Jones Chui for providing a C program which integrates the Mackey-Glass differential delay equation and converts the chaotic time series into state vectors.

Some good friends have contributed a lot to the material presented in Chapter 6. They are Mr. Jones Chui, Mr. Roger H. K. Wong, Mr. W. H. Mow, Mr. C. S. Leung and Mr. K. W. Choi. Without their enthusiastic discussions with me, the theory about the dimension presented in Chapter 6 would be much far from complete or even not exist. I would like to express my appreciation to all of them.

In addition, I greatly appreciate the financial support of the Croucher Foundation in my master's studies.

Abstract

In this thesis we want to acquire a deep understanding of topological map, and this is achieved by giving it a rigorous mathematical treatment. Our work is initiated by trying to precisely define the meaning of preserving the topological order. Now the set of neurons is considered as the underlying set of a fully looped symmetrical graph. In other words, we have superimposed a graph structure on the set of neurons, and this rigorous graph structure replaces the traditional vague "spatial relationships" of the neurons. Kohonen's algorithm can be easily generalized to handle arbitrary graphs rather than the traditional well-known cases only. Then we define the local minimum of a real-valued function defined on this graph, and by considering the distance function as a particular case, we easily give a precise definition of preserving the topological order. Accompany this definition is a quantity J_1 which, roughly speaking, is the average number of local minima of the distance function minus 1. J_1 should be almost zero if the topological order is preserved. Hence we can use J_1 to detect whether Kohonen's algorithm has been successful or not. This method is especially useful in high dimensions. Then we reformulate the theory using Voronoi regions. By this we obtain a clear picture about preserving the topological order, that is, the locally defined Voronoi regions are identical to the globally defined Voronoi

regions. From this the concept of induced graph emerges naturally. Here we do not place the neurons in the input space in such a way that the pre-specified graph preserves the topological order (which has been the job of Kohonen's algorithm), but rather to construct a graph according to the placement of the neurons such that this graph preserves the topological order. The induced graph, roughly speaking, is the minimal graph which preserves the topological order of the input space, and it always tries to reflect the structure of the input space. If the neurons are suitably placed in the input space, we can obtain a good representation of the input space through the induced graph. Hence the problem of studying the structure of the input space is transformed to the problem of studying the induced graph. This leads to a lot of new tools and new directions for studying the input space. In this thesis we try to extract the information about the dimension of the input space from the induced graph. Although the results are satisfactory even for the dimension of attractor of a chaotic time series or the dimension of speech space, there are too many mysteries in this area and our understanding of it is still far from enough.

Contents

Chapter 1

Background

1.1 Introduction

This beginning section contains some personal viewpoints on this thesis. We hope that the material presented in the later sections of this thesis, like other mathematical objects, is *rigorous enough to have its meanings self-evident*, so that we can confine our personal viewpoints and personal interpretations to this section only.

In the field "Artificial Neural Networks", one may often hear that "Kohonen's algorithm preserves the topological order". When we first heard of this, we asked two questions naturally. The first question was, "What is the precise meaning of preserving the topological order?" After we studied more, we found that its precision meaning was only given individually to some special cases, but not globally to any general cases. *Moreover, we did not know actually what the general cases were.* The second question was, "What is the importance of preserving the topological order?" In fact, we thought that its importance had

been underestimated for a long time. At last we found that the reason of being this was simply we could not answer the first question. Our conclusion was that "preserving the topological order" had been so far *a description only rather than a statement.* It just described a class of phenomena which were observed after Kohonen's algorithm had been used. Whenever we read "preserving the topological order", we just associated it with this class of phenomena. The success of the association shadowed the need to give it a precise meaning. However, because of the lack of a precise meaning, it could never become a statement. As a result *it had no logical or mathematical consequences, and the power of preserving the topological order was so far not completely disclosed.* To completely reveal its power, and to get a deep understanding of it, *a rigorous mathematical treatment becomes unavoidable.*

All our work in this thesis was initiated by trying to give a precise definition of preserving the topological order. It may be surprising at a glance, but not surprising at all after you go through our work, the definition of preserving the topological order is not the main product of this thesis. It is even not a very important product. It is often the case that when you know more and more about the nature of the problem, and eventually get a macroscopic understanding of it, the original matter which attracts your attention becomes so negligible.

There are two main products in this thesis. The first one, which we consider as the fusion of topological ideas and graph ideas, is to define the local minimum of a real-valued function defined on a graph. "Local" is clearly a topological concept, which is defined in terms of a neighborhood. The real-valued function defined on the graph is in fact defined on the vertices, with the edges constituting the topology. Note that it is quite different from the usual case that the values

are assigned to the edges instead of the vertices. By considering a special real-valued function, the distance function, we easily construct our primary definition of preserving the topological order. The second main product of this thesis is the induced graph, which converts the studies of the structure of the input space to the studies of the structure of a graph. In fact the approach is not new. The reader may be familiar with the transformation of a periodic signal to its Fourier coefficients, or using Laplace transform to solve a differential equation, or the matrix representation of a linear operator in a finite dimensional vector space, or studying the topology through the fundamental group. Of course we do not mean that we have set up such a strong correspondence between the input space and the induced graph, but the idea is indeed similar. After the transformation new tools are available to tackle the original problem. In our case if one is familiar with the techniques in graph theory, one may discover a lot of new methods to analyze the structure of the input space, and a lot of new applications of our theory. The power is up to one's experience, knowledge and imagination. (Unfortunately we are not such experts in graph theory.) Someone may argue that there is a third main product in this thesis: the studies of the relation between the structure of the induced graph and the dimension of the input space. Although we only consider it as a special application of the induced graph, undoubtedly it is a surprisingly large and wonderful area of researches, and there is a lot of undiscovered and amazing knowledge in it.

In the remainder of this chapter we would introduce some basic notations used in this thesis and then define the mathematical objects we shall be discussing about. In Chapter 2 we give a brief review of Kohonen's algorithm, with a discussion on its characteristics and limitations. In Chapter 3 we introduce

the local minimum of a real-valued function defined on a graph, and its related theory and applications. In Chapter 4 we give a formal treatment to the theory of the induced graph. In Chapter 5 we present some special examples based on the concept of the induced graph. In Chapter 6 we focus on a special topic: the application of the induced graph to the determination of the dimension of the input space. We conclude our work in Chapter 7.

To read this thesis, some background knowledge in neural networks, Kohonen's algorithm, topology, graph theory and Voronoi regions is helpful but not important. However, we think that training in mathematical argumentation may help a lot, because the meanings are often embedded in both the statements and the arguments. The reader may often see what we cannot see. That is why we do not want to add too many personal interpretations which may mislead the thinking of the reader.

1.2 Basic notations

In this section we introduce some basic notations we shall use in this thesis.

Suppose A is a set. Then card(A) denotes the cardinality of A. $\mathcal{P}(A)$ denotes the power set of A, i.e. the set of all subsets of A. Suppose B is also a set. For $A \subset B$ or $B \supset A$ we mean that A is a subset of B. We may also say that A is contained in B or B contains A. The word "contain" is reserved for this purpose only. We do not exclude the possibility that $A = B$. If we want to emphasize that $A \subset B$ but $A \neq B$, we would say that A is strictly contained in B.

Some sets have special symbols. $\mathbf{Z}$ always denotes the set of integers. $\mathbf{R}^n$ is the n-dimensional Euclidean space, i.e. its topology is the usual topology induced

from the Euclidean norm. $\mathbf{S}^1$ is a subspace of $\mathbf{R}^2$ which is defined by

$$\mathbf{S}^1 = \{x \in \mathbf{R}^2 : \|x\| = 1\}$$

The topology of $\mathbf{S}^1$ is obviously the relative topology induced from the usual topology of $\mathbf{R}^2$.

Suppose f is a function, or a map, such that its domain is a set B and its range is contained in a set C. Then we shall write $f : B \to C$. The range of f will be denoted by $f(B)$, i.e.

$$f(B) = \{f(x) : x \in B\}$$

Suppose g is a function such that its domain contains $f(B)$ and its range is contained in D. Then $g \circ f : B \to D$ denotes the composition, i.e.

$$g \circ f(x) = g(f(x)) \quad \forall x \in B$$

Suppose $A \subset B$. Then $f|_A$ denotes the restriction of f to A.

We shall use $\lfloor\ \rfloor$ to denote the floor function, i.e.

$$\lfloor x \rfloor = \text{ the largest integer which is smaller than or equal to } x$$

In this thesis a matrix is always boldfaced. Suppose $\mathbf{A}$ is a matrix. Then $(\mathbf{A})_{ij}$ denotes the entry in the i-th row and j-th column. $\mathbf{A}^{\mathrm{T}}$ denotes the transpose of $\mathbf{A}$ and $\mathbf{A}^{-1}$ denotes the inverse of $\mathbf{A}$.

When we discuss algorithms, especially iterating algorithms, some functions (or more commonly called variables) may be updated. We use the superscript $+$ to denote "after updated" and the prefix Δ to denote the difference between "after updated" and "before updated".

Other conventions we shall adopt are the followings. If we want to use brackets as delimiters, we shall only use [ ]. ( ) is reserved for function values, coordinates, sequences, matrices, etc. and { } is reserved for sets. However, we shall still follow the tradition in representing intervals in real line, such as $[-1, 1)$. We shall try to avoid superscript whenever possible. Even if we use it, we ever try to use special symbols, such as + or adding brackets ( ). Whenever there is a number in the upper right corner, it always means power (except for $\mathbf{S}^1$). For examples, $\mathbf{R}^2 = \mathbf{R} \times \mathbf{R}$ and $[\mathbf{S}^1]^3 = \mathbf{S}^1 \times \mathbf{S}^1 \times \mathbf{S}^1$. A tilde $\tilde{}$ over a symbol usually means that it is an approximation of the original symbol, and a bar $\bar{}$ over a symbol usually means that it is an average value.

A box □ denotes the end of a definition, proposition, assumption, etc. and three boxes □□□ denote the end of a remark or example.

1.3 Object of study

In this section we define the mathematical objects we are discussing about.

Let $X \subset \mathbf{R}^n$ be the input space. For input space we mean the closure of the set of all possible input patterns. For $X \subset \mathbf{R}^n$ we mean that each input pattern has n attributes, although in general they may not be independent. The topology of X, $\mathcal{T}_X$, is the relative topology on X which is naturally induced from the usual topology of $\mathbf{R}^n$. (For relative topology see [20, page 93].)

Let $\tilde{x} : \{1, 2, \ldots, M\} \to X$ be a finite sequence of input patterns. We call $\tilde{x}$ the sequence of training patterns. We assume that $\tilde{x}$ "sufficiently represents" X. "Sufficiently represents" may be the most unclear concept in this thesis, especially since $\tilde{x}$ has finite length, concepts like dense are not applicable. Although

X is what we are primarily interested in, we can never observe X. What we can obverse is $\tilde{x}$. Any theoretical measurement on X will eventually be transformed to a practical measurement on $\tilde{x}$. For "sufficiently represents", we hope that such a transformation will preserve the result of measurement. We shall encounter such transformations in Chapter 3 and Chapter 4.

To generate $\tilde{x}$ from X, we assume that there exists a probability distribution defined on X. Then $\tilde{x}$ is a random sequence generated based on this probability distribution. (The probability distribution is assumed to be uniform if not specified.) As the length of $\tilde{x}$, M, is sufficiently large, we assume that $\tilde{x}$ may sufficiently represents X.

Let (Q, N) be a finite set of K elements $Q = \{q_1, q_2, \ldots, q_K\}$ equipped with a mapping $N : Q \to \mathcal{P}(Q)$ which satisfies the following two conditions:

[N1] $\forall q_i \in Q, \ \ q_i \in N(q_i)$

[N2] $\forall q_i, q_j \in Q, \ \ q_i \in N(q_j)$ iff $q_j \in N(q_i)$

We call q_i a neuron and $N(q_i)$ the 1-neighborhood of q_i. (Grabec [6, 7] called q_i a "formal neuron" to emphasize that it is a formal object and does not necessarily have any physical meaning.)

> **Remark 1.1** (Q, N) is in fact a symmetrical graph. Therefore we shall adopt the terminology of graph theory whenever suitable. (For reference see [10, pages 4–11].) We say that there is a link between q_i and q_j whenever $q_i \in N(q_j)$ or equivalently $q_j \in N(q_i)$. Also we define
>
> $$\deg(q_i) = \mathrm{card}(N(q_i)) \quad \forall q_i \in Q$$
>
> □□□

Remark 1.2 (Q, N) is, of course, a topological space in the sense that its topology $\mathcal{T}_Q$ is the topology generated by $S = \{N(q_i) : q_i \in Q\}$. (For generated topology see [20, pages 101–102]. In brief, $S = \{N(q_i) : q_i \in Q\}$ can serve as an open subbase for a topology on Q, and this topology is called the topology generated by S. Equivalently this topology is the weakest topology on Q which contains S.) However, it is obvious that many N's can generate the same $\mathcal{T}_Q$ and for our purpose the topology only is not sufficient to characterize Q.

A direct consequence of Condition **[N1]** of N is

$$\bigcup_{q_i \in Q} N(q_i) = Q \tag{1.1}$$

i.e. $S = \{N(q_i) : q_i \in Q\}$ is in fact an open cover of $(Q, \mathcal{T}_Q)$.

□□□

Define $w : Q \to \mathbf{R}^n$. Traditionally $w(Q) = \{w(q_i) : q_i \in Q\}$ is called the set of weights, although this name does not fit the physical meaning well in cases like Kohonen's algorithm. Anyway, what we want to do is, through the mapping w, to place a set of K neurons in $\mathbf{R}^n$ such that $w(Q)$ is in some sense a good representation of X. Under this consideration the task is quite similar to performing a vector quantization on X. The difference is that in addition we want to consider the graph structure of (Q, N).

There are two graph structures on Q which we shall frequently use as examples:

1.

$$N(q_i) = \begin{cases} \{q_K, q_1, q_2\} & \text{if} \quad i = 1 \\ \{q_{i-1}, q_i, q_{i+1}\} & \text{if} \quad 2 \le i \le K-1 \\ \{q_{K-1}, q_K, q_1\} & \text{if} \quad i = K \end{cases}$$

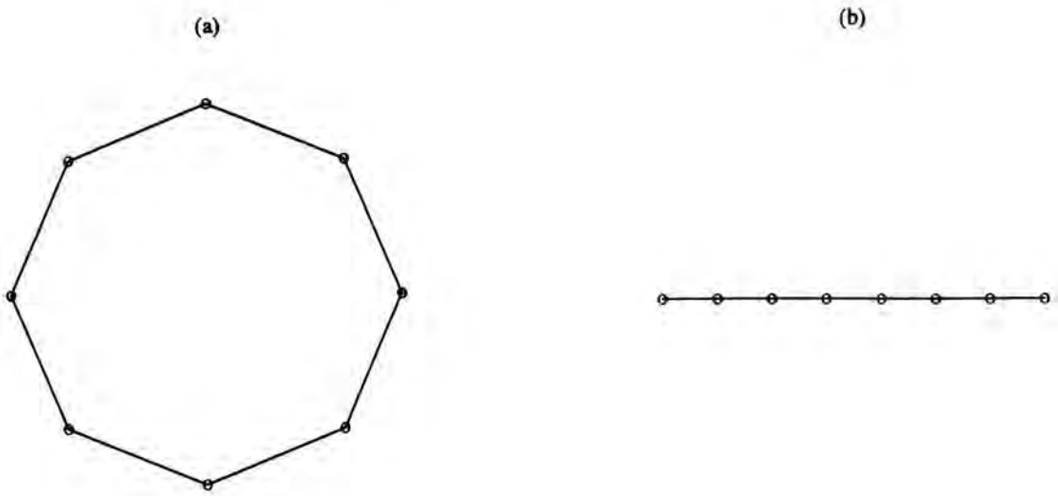

Figure 1.1: (a) $\mathbf{S}_8^1$. (b) $\mathbf{R}_8$.

We denote Q equipped with this N by $\mathbf{S}_K^1$. Figure 1.1(a) tries to illustrate this graph structure, in which each small circle represents a neuron and each straight line between two small circles represents a link between these two neurons. (The trivial link from a neuron to itself is always not drawn.) In this case $\deg(q_i) = 3 \ \ \forall q_i \in Q$, and w can be represented by a wrapped around one dimensional array in computer programming.

2.

$$N(q_i) = \begin{cases} \{q_1, q_2\} & \text{if} \quad i = 1 \\ \{q_{i-1}, q_i, q_{i+1}\} & \text{if} \quad 2 \leq i \leq K-1 \\ \{q_{K-1}, q_K\} & \text{if} \quad i = K \end{cases}$$

We denote Q equipped with this N by $\mathbf{R}_K$. Figure 1.1(b) tries to picture this graph structure. In this case $\deg(q_i) = 3 \ \ \forall q_i \in Q$ except q_1 and q_K whose deg $= 2$, and w can be represented by a non-wrapped around one dimensional array in computer programming.

We remark that the mapping N has a matrix representation, denoted by $\mathbf{N}$, defined by

$$(\mathbf{N})_{ij} = \begin{cases} 1 & \text{if} \quad q_i \in N(q_j) \\ 0 & \text{if} \quad q_i \notin N(q_j) \end{cases}$$

Then, immediately from the definition of N, $\mathbf{N}$ is a $K \times K$ real symmetric matrix with all 1's on the diagonal.

(Although we shall not do so in this thesis, it is in fact possible to generalize the matrix representation $\mathbf{N}$ such that $(\mathbf{N})_{ij}$ can be any real number in $[0, 1]$ instead of either 0 or 1 only, in order to represent, for example, the probability of q_i to be in the 1-neighborhood of q_j.)

Example 1.1 The matrix representation of N for $(Q, N) = \mathbf{S}_K^1$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 1 & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ 1 & 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix}$$

□□□

Example 1.2 The matrix representation of N for $(Q, N) = \mathbf{R}_K$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 1 & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix}$$

□□□

Next we introduce the sum and product of two graphs. (For sum and product of graphs see [10, pages 244–247].) Suppose $Q_1 = \{q_1^1, q_2^1, \ldots, q_{K_1}^1\}$ and $Q_2 = \{q_1^2, q_2^2, \ldots, q_{K_2}^2\}$ with $K_1 K_2 = K$. Define the sum $(Q, N_+) = (Q_1, N_1) + (Q_2, N_2)$ to be the set $Q = Q_1 \times Q_2$ equipped with the mapping $N_+ : Q \to \mathcal{P}(Q)$ defined by

$$N_+((q_i^1, q_j^2)) = [N_1(q_i^1) \times \{q_j^2\}] \cup [\{q_i^1\} \times N_2(q_j^2)]$$

We shall adopt the numbering convention that $Q = \{q_1, q_2, \ldots, q_K\}$ where $q_{[i-1]K_2+j} = (q_i^1, q_j^2)$. Similarly we can define any finite sum of graphs $(Q, N_+) = (Q_1, N_1) + (Q_2, N_2) + \cdots + (Q_n, N_n)$. The numbering convention is easily checked to be associative.

The sums generated by $\mathbf{R}_K$ and $\mathbf{S}_K^1$ are of special interest and thus we give symbols for them:

$$n\mathbf{R}_K = \underbrace{\mathbf{R}_{K^{\frac{1}{n}}} + \mathbf{R}_{K^{\frac{1}{n}}} + \cdots + \mathbf{R}_{K^{\frac{1}{n}}}}_{n}$$

$$n\mathbf{S}_K^1 = \underbrace{\mathbf{S}_{K^{\frac{1}{n}}}^1 + \mathbf{S}_{K^{\frac{1}{n}}}^1 + \cdots + \mathbf{S}_{K^{\frac{1}{n}}}^1}_{n}$$

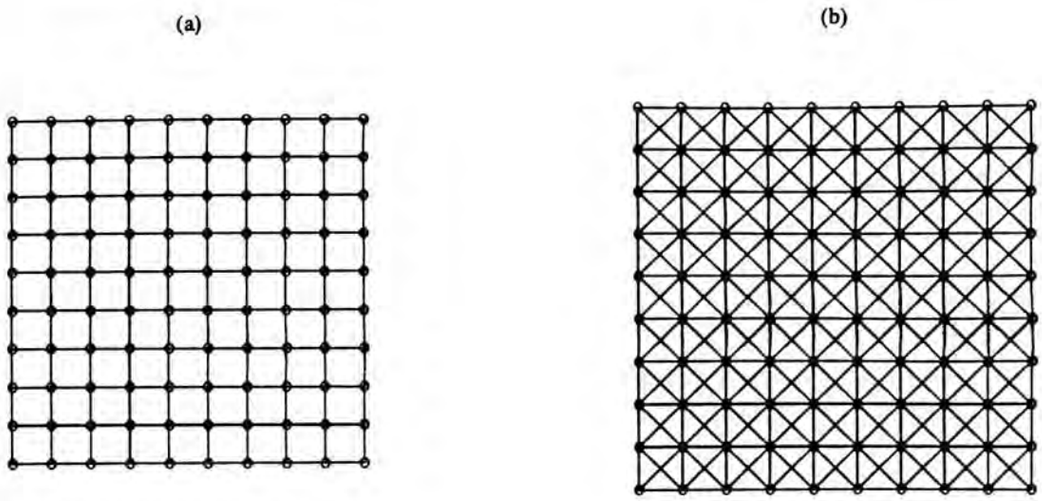

Figure 1.2: (a) $2\mathbf{R}_{100}$. (b) $\mathbf{R}^2_{100}$.

Hence w defined on $(Q, N) = n\mathbf{R}_K$ and $(Q, N) = n\mathbf{S}^1_K$ may be represented by non-wrapped around and wrapped around n-dimensional arrays respectively. The reader may notice the close correspondences between $n\mathbf{R}_K$ and $\mathbf{R}^n$ as well as $n\mathbf{S}^1_K$ and $[\mathbf{S}^1]^n$.

Example 1.3 Figure 1.2(a) tries to picture $2\mathbf{R}_{100}$.

□□□

Example 1.4 The matrix representation of N for $(Q, N) = 2\mathbf{S}_9^1$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

□□□

Example 1.5 The matrix representation of N for $(Q, N) = 2\mathbf{R}_9$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

□□□

Remark 1.3 Suppose $(Q, N_+) = (Q_1, N_1) + (Q_2, N_2)$. As in Remark 1.2, let $\mathcal{T}_{Q+}$, $\mathcal{T}_{Q_1}$, $\mathcal{T}_{Q_2}$ be the topologies generated by

$$\begin{aligned} S_+ &= \{N_+(q_{[i-1]K_2+j}) : q_{[i-1]K_2+j} \in Q\} \\ &= \{[N_1(q_i^1) \times \{q_j^2\}] \cup [\{q_i^1\} \times N_2(q_j^2)] : q_i^1 \in Q_1, q_j^2 \in Q_2\} \\ S_1 &= \{N_1(q_i^1) : q_i^1 \in Q_1\} \\ S_2 &= \{N_2(q_i^2) : q_i^2 \in Q_2\} \end{aligned}$$

respectively. We can, of course, form the product of topological spaces $(Q, \mathcal{T}_{Q_1Q_2}) = (Q_1, \mathcal{T}_{Q_1}) \times (Q_2, \mathcal{T}_{Q_2})$. The product topology $\mathcal{T}_{Q_1Q_2}$ is generated by

$$S_{12} = \{N_1(q_i^1) \times Q_2 : q_i^1 \in Q_1\} \cup \{Q_1 \times N_2(q_j^2) : q_j^2 \in Q_2\}$$

(For product topology see [20, pages 115–118].) The underlying sets of $(Q, \mathcal{T}_{Q+})$ and $(Q, \mathcal{T}_{Q_1Q_2})$ are the same, but the topologies are in general different. *$\mathcal{T}_{Q+}$ is in general stronger than the product topology $\mathcal{T}_{Q_1Q_2}$.* To show this, it suffices to show that any set in S_{12} is an open set in $(Q, \mathcal{T}_{Q+})$, i.e. some union of finite intersections of sets in S_+, and since $\mathcal{T}_{Q_1Q_2}$ is the weakest topology containing S_{12}, $\mathcal{T}_{Q+} \supset \mathcal{T}_{Q_1Q_2}$. The proof is at once completed if we note that

$$N_1(q_i^1) \times Q_2 = \bigcup_{q_j^2 \in Q_2} [N_1(q_i^1) \times \{q_j^2\}] \cup [\{q_i^1\} \times N_2(q_j^2)] \quad (1.2)$$

$$Q_1 \times N_2(q_j^2) = \bigcup_{q_i^1 \in Q_1} [N_1(q_i^1) \times \{q_j^2\}] \cup [\{q_i^1\} \times N_2(q_j^2)] \quad (1.3)$$

In obtaining Equations 1.2 and 1.3, we have applied Condition [N1] of N and its consequence Equation 1.1.

□□□

Example 1.6 $\mathcal{T}_{Q+}$ may really strictly contain $\mathcal{T}_{Q_1Q_2}$. Consider

$$\begin{aligned} Q_1 &= \{q_1^1, q_2^1\}\ ; \quad N_1(q_1^1) = N_1(q_2^1) = Q_1 \\ Q_2 &= \{q_1^2, q_2^2\}\ ; \quad N_2(q_1^2) = N_2(q_2^2) = Q_2 \end{aligned}$$

Then $\mathcal{T}_{Q+}$ is the discrete topology, the strongest topology on Q, and $\mathcal{T}_{Q_1Q_2}$ is the indiscrete topology, the weakest topology on Q.

□□□

Now we introduce the product of two graphs. Define the product $(Q, N_\times) = (Q_1, N_1) \times (Q_2, N_2)$ to be the set $Q = Q_1 \times Q_2$ equipped with the mapping $N_\times : Q \to \mathcal{P}(Q)$ defined by

$$N_\times((q_i^1, q_j^2)) = N_1(q_i^1) \times N_2(q_j^2)$$

The numbering convention is the same as before. Similarly we can define any finite product of graphs $(Q, N_\times) = (Q_1, N_1) \times (Q_2, N_2) \times \cdots \times (Q_n, N_n)$ and adopt the following two symbols:

$$\begin{aligned} \mathbf{R}_K^n &= \underbrace{\mathbf{R}_{K\frac{1}{n}} \times \mathbf{R}_{K\frac{1}{n}} \times \cdots \times \mathbf{R}_{K\frac{1}{n}}}_{n} \\ [\mathbf{S}^1]_K^n &= \underbrace{\mathbf{S}^1_{K\frac{1}{n}} \times \mathbf{S}^1_{K\frac{1}{n}} \times \cdots \times \mathbf{S}^1_{K\frac{1}{n}}}_{n} \end{aligned}$$

Once again w defined on $(Q, N) = \mathbf{R}_K^n$ and $(Q, N) = [\mathbf{S}^1]_K^n$ may be represented by non-wrapped around and wrapped around n-dimensional arrays respectively.

Example 1.7 Figure 1.2(b) tries to illustrate $\mathbf{R}_{10}^2$. The reader may compare it with Figure 1.2(a).

□□□

Example 1.8 The matrix representation of N for $(Q, N) = [\mathbf{S}^1]_9^2$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

□□□

Example 1.9 The matrix representation of N for $(Q, N) = \mathbf{R}_9^2$ is

$$\mathbf{N} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

□□□

Remark 1.4 Suppose $(Q, N_\times) = (Q_1, N_1) \times (Q_2, N_2)$. As in Remark 1.3, let $\mathcal{T}_{Q\times}$ be the topology generated by

$$\begin{aligned} S_\times &= \{N_\times((q_i^1, q_j^2)) : (q_i^1, q_j^2) \in Q\} \\ &= \{N_1(q_i^1) \times N_2(q_j^2) : q_i^1 \in Q_1, q_j^2 \in Q_2\} \end{aligned}$$

It is obvious that $\mathcal{T}_{Q_1Q_2} \supset \mathcal{T}_{Q\times}$ since

$$N_1(q_i^1) \times N_2(q_j^2) = [N_1(q_i^1) \times Q_2] \cap [Q_1 \times N_2(q_j^2)]$$

Hence each set in $S_\times$ is the intersection of two sets in S_{12}, and thus an open set in $(Q, \mathcal{T}_{Q_1Q_2})$. On the other hand, by applying Equation 1.1 again we may write each set in S_{12} as the union of some sets in $S_\times$

$$N_1(q_i^1) \times Q_2 = \bigcup_{q_j^2 \in Q_2} N_1(q_i^1) \times N_2(q_j^2) \tag{1.4}$$

$$Q_1 \times N_2(q_j^2) = \bigcup_{q_i^1 \in Q_1} N_1(q_i^1) \times N_2(q_j^2) \tag{1.5}$$

Hence $\mathcal{T}_{Q_1Q_2} \subset \mathcal{T}_{Q\times}$ and it follows that $\mathcal{T}_{Q_1Q_2} = \mathcal{T}_{Q\times}$.

□□□

Remark 1.5 Combining the results of **Remark 1.3 and Remark 1.4** we have

$$\mathcal{T}_{Q+} \supset \mathcal{T}_{Q_1Q_2} = \mathcal{T}_{Q\times} \tag{1.6}$$

In both proofs we made use of the fact that N satisfies Condition **[N1]**. For arbitrary graphs Relation 1.6 is in general not true. We shall only have $\mathcal{T}_{Q_1Q_2} \supset \mathcal{T}_{Q\times}$.

□□□

Example 1.10 Consider

$$\begin{aligned} Q_1 &= \{q_1^1, q_2^1\}\ ; \quad N_1(q_1^1) = \emptyset\ ,\ N_1(q_2^1) = \{q_2^1\} \\ Q_2 &= \{q_1^2, q_2^2\}\ ; \quad N_2(q_1^2) = \{q_1^2\}\ ,\ N_2(q_2^2) = \{q_2^2\} \end{aligned}$$

Then

$$\begin{aligned} \mathcal{T}_{Q+} &= \text{discrete topology} \\ \mathcal{T}_{Q_1Q_2} &= \left\{ \text{any unions of } \begin{array}{l} \{(q_2^1, q_1^2)\}, \\ \{(q_2^1, q_2^2)\}, \\ \{(q_1^1, q_1^2)\} \cup \{(q_2^1, q_1^2)\}, \\ \{(q_1^1, q_2^2)\} \cup \{(q_2^1, q_2^2)\} \end{array} \right\} \\ \mathcal{T}_{Q\times} &= \left\{ \text{any unions of } \begin{array}{l} \{(q_2^1, q_1^2)\}, \\ \{(q_2^1, q_2^2)\}, \\ Q \end{array} \right\} \end{aligned}$$

Hence $\mathcal{T}_{Q+}$ strictly contains $\mathcal{T}_{Q_1Q_2}$ and $\mathcal{T}_{Q_1Q_2}$ strictly contains $\mathcal{T}_{Q\times}$. This example shows that $\mathcal{T}_{Q_1Q_2}$ is not necessarily equal to $\mathcal{T}_{Q\times}$ for arbitrary graphs.

□□□

Example 1.11 Consider

$$\begin{aligned} Q_1 &= \{q_1^1, q_2^1\}\ ; \quad N_1(q_1^1) = \{q_2^1\}\ ,\ N_1(q_2^1) = \{q_1^1\} \\ Q_2 &= \{q_1^2, q_2^2\}\ ; \quad N_2(q_1^2) = \{q_2^2\}\ ,\ N_2(q_2^2) = \{q_1^2\} \end{aligned}$$

Then

$$\begin{aligned} \mathcal{T}_{Q+} &= \left\{ \text{any unions of } \begin{array}{l} \{(q_1^1, q_2^2)\} \cup \{(q_2^1, q_1^2)\}, \\ \{(q_1^1, q_1^2)\} \cup \{(q_2^1, q_2^2)\} \end{array} \right\} \\ \mathcal{T}_{Q_1Q_2} &= \text{discrete topology} \\ \mathcal{T}_{Q\times} &= \text{discrete topology} \end{aligned}$$

Hence $\mathcal{T}_{Q+}$ is strictly contained in $\mathcal{T}_{Q_1Q_2} = \mathcal{T}_{Q\times}$. This example shows that both $\mathcal{T}_{Q+} \supset \mathcal{T}_{Q_1Q_2}$ and $\mathcal{T}_{Q+} \supset \mathcal{T}_{Q\times}$ are not necessarily true for arbitrary graphs.

□□□

Before ending this section we discuss an assumption which we shall often add on X.

Assumption 1 X is convex.

□

In fact, convexity is a very demanding condition which is very difficult to achieve, but of course has a lot of useful consequences. Practically we only need some of such useful consequences (see Chapter 3 and Chapter 4), and convexity, such a strong condition, may be actually not needed. However, convexity is needed to assign a natural geometrical interpretation to what we shall do. To explain this, we start with the concept of distance.

Suppose $s, s' \in X$. For distance between s and s' we always mean the length of geodesic joining s and s'. Geodesic, roughly speaking, is the shortest path along the space. When we consider s, s' as points in X, we call the length of geodesic joining them the *geodesic distance* between them. If X is path-connected, there must exist a path connecting s and s', and also since X is closed, the shortest path, i.e. the geodesic, must exist. Hence the geodesic distance is always defined. However, when we consider s, s' as points in $\mathbf{R}^n$, the geodesic joining them, to which we often refer as the "straight line" joining them, is in general different from the geodesic joining them along X, and we shall call its length the *direct distance* between s and s'. Figure 1.3 pictures the

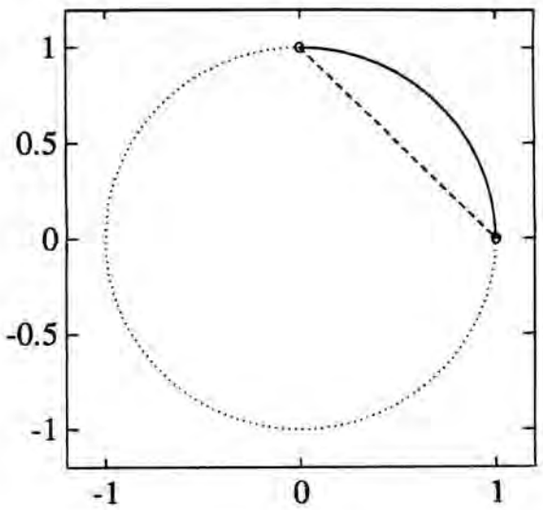

Figure 1.3: $X = \mathbf{S}^1$ (dotted line). $s = (0,1)$ and $s' = (1,0)$. The geodesic distance (length of solid line) between them is $\frac{\pi}{2}$. The direct distance (length of dashed line) between them is $\sqrt{2}$.

difference between these two distances for the case $X = \mathbf{S}^1$.

Since we are studying X, whenever we talk about distance we should use geodesic distance. However, it is very difficult to measure the geodesic distance. Almost in all cases direct distance will be measured instead. Practically such a substitution often produces satisfactory results, but theoretical problems may be encountered. We shall encounter some in Chapter 3 and Chapter 4. To avoid such problems it requires that the geodesic distance between any two points in X is always equal to the direct distance between them, and hence the solution is, of course, convexity.

It will be clearer in Chapter 3 and Chapter 4 that geodesic distance is the correct distance we should use, as we shall see that using geodesic distance the whole theory will not need the convexity of X.

Since in practice it is very difficult to measure the geodesic distance, we shall

from now on reserve the word "distance" for the direct distance, although direct distance and geodesic distance become identical if X is convex.

Chapter 2

Review of Kohonen's algorithm

In this chapter we give a brief review of Kohonen's algorithm [11, 12, 15]. In Section 2.1 we introduce the general form of Kohonen's algorithm, so that we can consider arbitrary graphs. For example, we consider in Example 2.4 $(Q, N) = \mathbf{R}^4 + \mathbf{S}^1_{16}$. We shall also consider in Chapter 4, Example 4.5 that (Q, N) is the induced graph, and then the general form of Kohonen's algorithm becomes indispensable. Accompany the general form of Kohonen's algorithm in Section 2.1 is the definition of the r-neighborhood, and how to calculate *all* the r-neighborhoods is the main topic of Section 2.2. In Section 2.3 we give some remarks on Kohonen's algorithm in order to reveal its characteristics and limitations.

2.1 General form of Kohonen's algorithm

In this section we present the general form of Kohonen's algorithm.

Kohonen's algorithm is in terms of a neighborhood whose size can be varied.

To make it precise, we give the following definition.

Definition 2.1 The r-neighborhood, where r is a non-negative integer, of a neuron q_i is a subset of Q which is defined recursively by

[Nr1] 0-neighborhood of $q_i = \{q_i\}$

[Nr2] r-neighborhood of $q_i = \bigcup N(q_j) : q_j \in r-1$-neighborhood of q_i

□

We shall denote the r-neighborhood of q_i as $N^{(r)}(q_i)$. r is also called the radius of the neighborhood. The matrix representation of $N^{(r)}$ is denoted by $\mathbf{N}^{(r)}$, which is naturally defined by

$$(\mathbf{N}^{(r)})_{ij} = \begin{cases} 1 & \text{if} \quad q_i \in N^{(r)}(q_j) \\ 0 & \text{if} \quad q_i \notin N^{(r)}(q_j) \end{cases}$$

Now we define the general form of Kohonen's algorithm as follows. Suppose the training patterns are presented sequentially. Let $\tilde{x}_l$ be the current training pattern and $q_i \in Q$ such that

$$d(w(q_i), \tilde{x}_l) \leq d(w(q_j), \tilde{x}_l) \quad \forall q_j \in Q$$

where d is the distance function. Then

$$w^+(q_j) = w(q_j) + \eta[\tilde{x}_l - w(q_j)] \quad \forall q_j \in \text{ } r\text{-neighborhood of } q_i \qquad (2.1)$$

where the superscript $+$ means "after updated" and η is the learning rate. r is in general decreasing with time. It is also known that η decreasing with time may also increase the performance of Kohonen's algorithm. However, since the examples and the applications in this thesis are not very complicated, we shall

use a fixed learning rate and the performance is still sufficiently good. The initial weights are just set to the first K training patterns.

Practically we shall use the following recursive procedure to find out the r-neighborhood of q_i. (The procedure is implemented using C programming language.)

```
void set_in_neighborhood (int q, int r)
{
  int j;
  if (r > 0)
  {
    for (j = 1; j <= K; j++)
      if (q_j ∈ N(q))
        set_in_neighborhood (q_j, r − 1);
  }
  else
    in_neighborhood[q] = 1;
}
```

Before calling the procedure, we first reset `in_neighborhood[`q_j`]`$= 0$ for all $q_j \in Q$. Then after calling `set_in_neighborhood (`q_i`, r)`, we have

$$\texttt{in\_neighborhood}[q_j] = 1 \quad \text{iff} \quad q_j \in N^{(r)}(q_i)$$

The reason of defining the r-neighborhood and the general form of Kohonen's algorithm is that we can discuss Kohonen's algorithm for arbitrary mappings N instead of restricting our discussions to some well-known cases. So it is much

convenient for us to employ the concept "r-neighborhood" in doing some theoretical generalizations. One may notice that, according to the neighborhoods traditionally chosen for Kohonen's algorithm, the corresponding graph structure should be $\mathbf{R}_K^n$, but it was always drawn as $n\mathbf{R}_K$. In Section 1.3 we saw that $\mathbf{R}_K^n$ has much more connections than $n\mathbf{R}_K$. Moreover, one link may intersect another when drawn, which makes the graph difficult to visualize. Therefore we shall follow the tradition. Both $n\mathbf{R}_K$ and $\mathbf{R}_K^n$ will be drawn as $n\mathbf{R}_K$, and we shall explicitly point out which graph we are talking about. (We remark that in practice we would not use the above recursive procedure to find out $N^{(r)}(q_i)$ if we are considering the traditional graph $(Q, N) = \mathbf{R}_K^n$, since it is much slower when r is large. For this special case we can find out $N^{(r)}(q_i)$ easily, and the recursive procedure is designed for arbitrary graphs.)

For simplicity we may sometimes fix the radius of neighborhood in Kohonen's algorithm. If the radius is fixed to r, we shall call the corresponding learning rule r-Kohonen's algorithm. We shall often use 1-Kohonen's algorithm. For cases where K is small, 1-Kohonen's algorithm is sufficient to illustrate the ideas. Another special case is 0-Kohonen's algorithm, which is identical to the well-known "standard competitive learning" [8, pages 218–220]. We shall simply call it competitive learning.

2.2 r-neighborhood by matrix

In the previous section we defined the r-neighborhood recursively. In fact we can define it using $\mathbf{N}$, the matrix representation of N. To see this, we start with a definition.

Definition 2.2 A path of length r, $r \geq 0$, from a neuron q_i to a neuron q_j is a finite sequence $\xi : \{1, 2, \ldots, r+1\} \rightarrow Q$ such that

[P1] $\xi_1 = q_i$

[P2] $\xi_{r+1} = q_j$

[P3] For $2 \leq m \leq r+1$, $\xi_m \in N(\xi_{m-1})$

□

Then the following result is well-known.

Proposition 2.1 $(\mathbf{N}^r)_{ij}$ is the number of distinct paths of length r from q_j to q_i.

Proof. We prove Proposition 2.1 by induction. Proposition 2.1 is trivially true for $r = 0$ and $r = 1$. For r, it is clear that

number of distinct paths of length r from q_j to q_i =

$$\sum_{k=1}^{K} [\text{ number of distinct paths of length } r-1 \text{ from } q_j \text{ to } q_k] \times$$
$$[\text{ number of distinct paths of length 1 from } q_k \text{ to } q_i]$$

Hence if Proposition 2.1 is true for $r - 1$,

$$\text{number of distinct paths of length } r \text{ from } q_j \text{ to } q_i$$
$$= \sum_{k=1}^{K} (\mathbf{N}^{r-1})_{kj} \times (\mathbf{N})_{ik} = (\mathbf{N} \times \mathbf{N}^{r-1})_{ij} = (\mathbf{N}^r)_{ij}$$

□

The following Corollary is evident from Proposition 2.1 and the definitions.

Corollary 2.2 $q_i \in$ r-neighborhood of q_j iff $(\mathbf{N}^r)_{ij} \neq 0$.

□

In other words,

$$\mathbf{N}^{(r)} = \text{nonzero}(\mathbf{N}^r)$$

where

$$(\text{nonzero}(\mathbf{A}))_{ij} = \begin{cases} 1 & \text{if} \quad (\mathbf{A})_{ij} \neq 0 \\ 0 & \text{if} \quad (\mathbf{A})_{ij} = 0 \end{cases}$$

To find out the r-neighborhood of 1 neuron, it is more convenient to use the recursive procedure. However, to find out the r-neighborhoods of all neurons, it becomes more convenient to calculate nonzero($\mathbf{N}^r$). There are many practical hints in calculating nonzero($\mathbf{N}^r$):

1. To calculate nonzero($\mathbf{N}^r$), it is no need to calculate $\mathbf{N}^r$, or even $\mathbf{N}^m$ for $2 \leq m < r$. Note that

   $$\text{nonzero}(\mathbf{N}^r) = \text{nonzero}(\mathbf{N} \times \text{nonzero}(\mathbf{N}^{r-1}))$$

   since the entries in $\mathbf{N}$ and $\mathbf{N}^{r-1}$ are all non-negative. Therefore we only need to calculate nonzero($\mathbf{N}^m$) in each intermediate stage. As we know that $(\text{nonzero}(\mathbf{N}^m))_{ij}$ can be only 0 or 1, the multiplication is reduced to a logical operation "AND". Moreover, what we want is

   $$\text{nonzero}(\mathbf{N} \times \text{nonzero}(\mathbf{N}^m))$$

   instead of

   $$\mathbf{N} \times \text{nonzero}(\mathbf{N}^m)$$

   so the following two hints can further reduce the amount of computations.

2. Note that $q_i \in N^{(m)}(q_j)$ implies $q_i \in N^{(m+1)}(q_j)$. In other words, if

$$(\text{nonzero}(\mathbf{N}^m))_{ij} = 1$$

then

$$(\text{nonzero}(\mathbf{N} \times \text{nonzero}(\mathbf{N}^m)))_{ij} = 1$$

(Purely from the computational point of view, we may think that it is because $(\mathbf{N})_{ii}$ must be 1.)

3. Note that

$$(\text{nonzero}(\mathbf{N} \times \text{nonzero}(\mathbf{N}^m)))_{ij} = 1$$

iff $\exists k$ such that

$$(\mathbf{N})_{ik} = (\text{nonzero}(\mathbf{N}^m))_{kj} = 1$$

since the entries in $\mathbf{N}$ are all non-negative. Therefore, once such a k is found, it is no need to go through other neurons. That means, besides the multiplication, it is also no need to perform the addition.

4. Note that $(\mathbf{N}^r)_{ij} = (\mathbf{N}^r)_{ji}$ since $(\mathbf{N})_{ij} = (\mathbf{N})_{ji}$, and $(\mathbf{N}^r)_{ii} = 1$ since $(\mathbf{N})_{ii} = 1$. Therefore it is no need to calculate the diagonal and the lower triangle of nonzero($\mathbf{N}^r$). We only need to calculate the upper triangle of nonzero($\mathbf{N}^r$).

2.3 Examples

At last we want to give some remarks on Kohonen's algorithm through several examples.

Example 2.1 The input space $X = \{x \in \mathbf{R}^2 : \|x\| \leq 1\}$. Each training pattern is parametrized as $(\rho\cos\phi, \rho\sin\phi)$, where ρ is randomly drawn from a uniform distribution on $[0, 1)$ and ϕ is randomly drawn from a uniform distribution on $[0, 2\pi)$. As a result, the density is higher near the origin. The total number of training patterns $M = 10000$. (Q, N) is chosen to be $\mathbf{R}^2_{64}$. The parameters of Kohonen's algorithm are set as follows. If $\tilde{x}_l$ is the current training pattern, then the radius of neighborhood

$$r = \begin{cases} \lfloor \frac{1000-l}{1000-1} \times 5 \rfloor & \text{if} \quad l \leq 1000 \\ 0 & \text{if} \quad l > 1000 \end{cases}$$

The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

η is so set because if η is not sufficiently small when $r = 0$, the original well-ordered grid[1] may be destroyed.

$w(Q)$ is plotted in Figure 2.1. The graph structure of (Q, N) is superimposed on the figure by connecting two *distinct* neurons by a straight line if there is a link. We shall, for convenience, denote this graphical representation as $w(Q, N)$. From Figure 2.1 we see that there are some pits along the boundary. It is because Kohonen's algorithm tends to place more neurons near the center and fewer neurons near the boundary in order to match the density of the training patterns.

□□□

The following examples consider the ring.

[1]Of course we have not defined what we mean by "well-ordered". The reader may interpret it in its traditional sense.

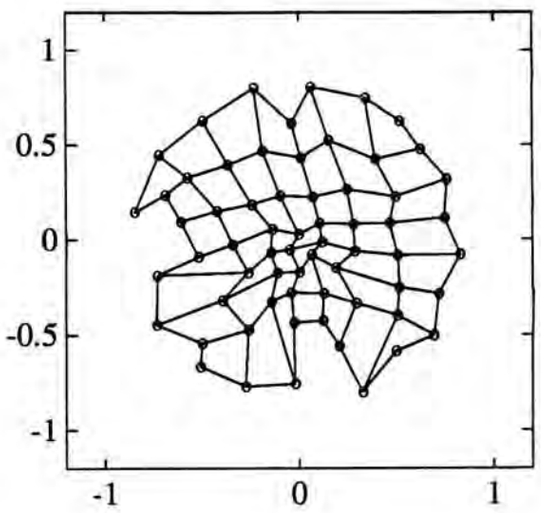

Figure 2.1: $w(Q, N)$ in Example 2.1.

Example 2.2 In this example $X = \{x \in \mathbf{R}^2 : 0.6 \leq \|x\| \leq 1\}$. The configuration is the same as that of Example 2.1 except that ρ is randomly drawn from a uniform distribution on $[0.6, 1)$ instead of $[0, 1)$. The result $w(Q, N)$ is plotted in Figure 2.2(a). We see that some neurons fall in the hole of the ring and some links run across the hole.

□□□

Example 2.3 The configuration is completely the same as that of Example 2.2, but this time the weights are restricted to the input space. This can be done by projecting the weight onto the input space immediately after it is updated by Learning Rule 2.1. As X is closed, the projection can be defined as the point in X which is closest to the weight, and practically regarded as the closest training pattern to the weight. Since the metric is Euclidean, such a projection is well-defined on a neighborhood

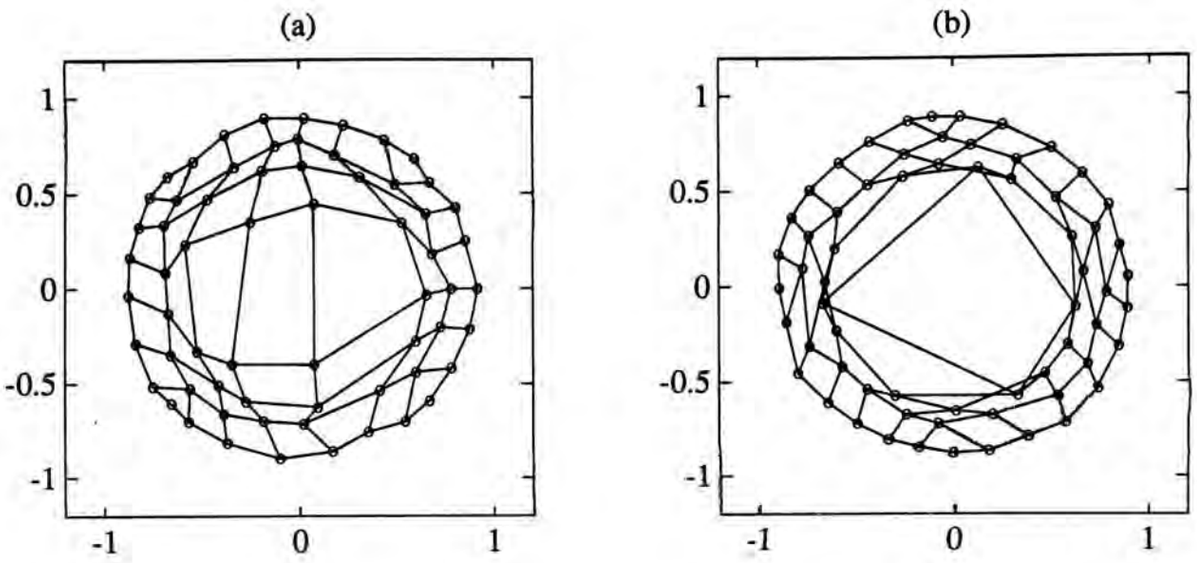

Figure 2.2: (a) $w(Q, N)$ in Example 2.2. (b) $w(Q, N)$ in Example 2.3.

of X. However, it cannot generally be defined on the whole $\mathbf{R}^n$. Therefore the learning rate η should not be set too large. In this example the input space is chosen to be so simple that we need not identify the closest training pattern, but just adjust the norm of the weight to achieve projection.

The result $w(Q, N)$ is shown in Figure 2.2(b). We see that although the neurons are restricted to the input space, some links still run across the hole.

□□□

The above two examples seem to say that Kohonen's algorithm cannot handle input spaces which are not simply connected. However, we see from the examples below that this conclusion is not absolutely correct.

Example 2.4 The input space and the sequence of training patterns are the same as those in Example 2.2, but this time $(Q, N) = \mathbf{R}^4 + \mathbf{S}^1_{16}$. Note

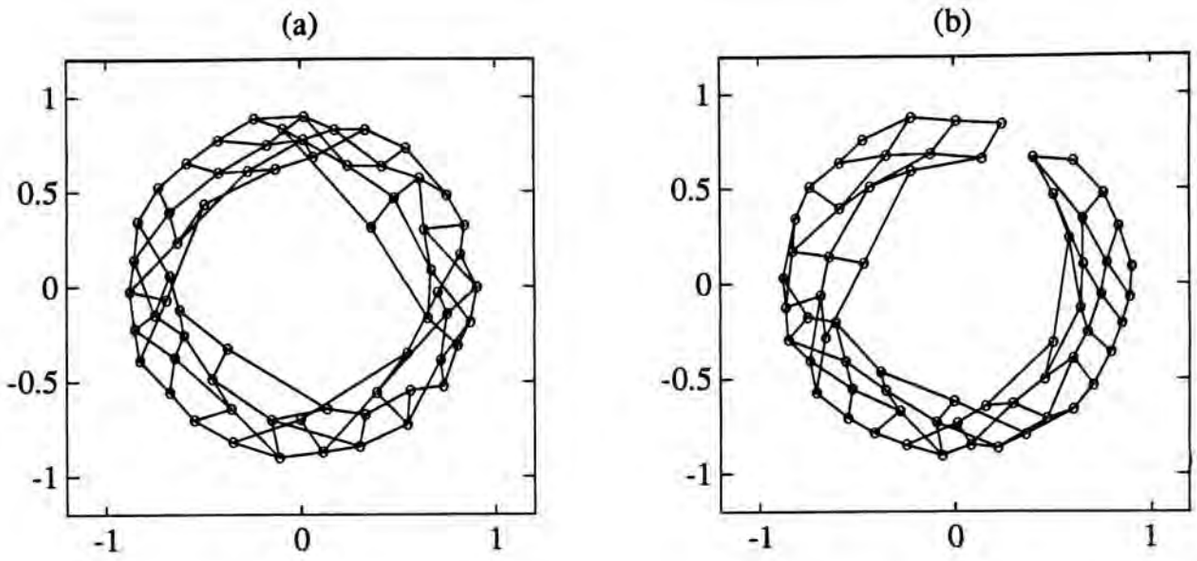

Figure 2.3: (a) $w(Q, N)$ in Example 2.4. (b) $w(Q, N)$ in Example 2.5.

that the total number of neurons is the same as before, but there are much fewer links. In fact, the number of links is approximately equal to that of $2\mathbf{R}_8$. As a result the size of r-neighborhood is much smaller than before. Therefore ,the radius of neighborhood is set much larger:

$$r = \begin{cases} \lfloor \frac{1000-l}{1000-1} \times 10 \rfloor & \text{if} \quad l \leq 1000 \\ 0 & \text{if} \quad l > 1000 \end{cases}$$

η is set the same as in the previous examples.

The result $w(Q, N)$ is shown in Figure 2.3(a). Although the grid is not completely well-ordered, we can observe a hole.

□□□

The reader may think that it is because we have used a graph which especially fits the ring. This is only partially true.

Example 2.5 The configuration is the same as that of Example 2.4, except that (Q, N) is now chosen as $\mathbf{R}_4 + \mathbf{R}_{16}$ instead of $\mathbf{R}_4 + \mathbf{S}_{16}$. The result $w(Q, N)$ is shown in Figure 2.3(b). Note that the result is quite different from that of Example 2.2. From this we expect that the result when $(Q, N) = \mathbf{R}_4 + \mathbf{R}_{16}$ is also very different from the result when $(Q, N) = \mathbf{R}_8 + \mathbf{R}_8$. In other words, besides the "dimension", the "length" in each "dimension" is also critical. In Chapter 4 we shall introduce the Voronoi regions formally. Kohonen's algorithm has a strong tendency to form isotropic Voronoi regions. (Our comments above may be viewed as a supplement to the discussions in [9, pages 94–96], in which a modification on Kohonen's algorithm was also proposed to obtain better results.)

□□□

From the above examples we see that to have a good match, we must have sufficient knowledge about the input space. That makes the induced graph we introduce in Chapter 4 more valuable if we know very little about the input space. We shall recall some of the above examples later when we discuss the induced graph and the dimension of the input space.

Chapter 3

Local minima

In this chapter we present one of the most important theoretical concepts in this thesis, which we also consider as the fusion of topological ideas and graph ideas: the local minimum of a function defined on a graph. With this concept we define in Section 3.1 what we mean by preserving the topological order, and hence a closely related quantity J_1. In Section 3.2 we study how to minimize J_1, with an analysis on Kohonen's algorithm at the end of the section. In Section 3.3 we introduce an important application of J_1: to detect whether Kohonen's algorithm succeeds or not in high dimensions. We also remark the difference between $n\mathbf{R}_K$ and $\mathbf{R}_K^n$. In Section 3.4 we see how the change of the graph structure affects the local minima, from which the vague idea of the induced graph begin to emerge. In Section 3.5, we generalize out theoretical work as well as the definition of preserving the topological order by employing the geodesic distance. At last, in Section 3.6, we see the budding of the induced graph through an equivalent formulation of our theory in terms of Voronoi regions. From this we see the close relation between the induced graph discussed in Chapter 4 and the local

minimum discussed in this chapter, although the material in Chapter 4 may also be treated independently.

3.1 Theory of local minima

Definition 3.1 Let $f : X \to \mathbf{R}$. We say that $x' \in X$ is a local minimum of f, or f has a local minimum at x', if $\exists$ a neighborhood[1] U of x' such that $f(x') \leq f(x) \ \forall x \in U$. We say that $x' \in X$ is a global minimum of f, or f has a global minimum at x', if $f(x') \leq f(x) \ \forall x \in X$. Note that a global minimum is obviously a local minimum.

□

Let $d : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ be the distance function, i.e.

$$d(x, x') = \text{ distance between } x \text{ and } x'$$

With x' fixed, $d(x, x')$ is a function of x. More precisely, we may define

$$d_{x'} : \mathbf{R}^n \to \mathbf{R} \text{ such that } d_{x'}(x) = d(x, x')$$

All our work is motivated by the following almost trivial fact

Proposition 3.1 Assume X is convex. Suppose $x' \in X$. Then $d_{x'} \mid_X$ has a unique local minimum at $x' \in X$.

□

and its weaker form

[1] A neighborhood U of $x' \in X$ is an open set U of X such that $x' \in U$.

Corollary 3.2 Assume X is convex. Suppose $x' \in X$. Then every local minimum of $d_{x'}\,|_X$ is a global minimum of $d_{x'}\,|_X$.

□

Corollary 3.2 is evident from Proposition 3.1. Here we skip the proof of Proposition 3.1. However, we shall give a proof to a similar proposition formulated in terms of geodesic in Section 3.5.

Convexity is a very strong restriction on X. Practically the methods we shall propose work well for some non-convex input spaces. So it is useful to single out the following class of input spaces. Let

$$\mathcal{C} = \{X : \forall x' \in X,\ \ d_{x'}\,|_X \ \text{ has a unique local minimum at } x' \in X\}$$

Then Proposition 3.1 is equivalent to saying that if X is convex, $X \in \mathcal{C}$.

Remark 3.1 It is easy to give an example in which X is not convex and $X \notin \mathcal{C}$. Define

$$\mathbf{P}(a,b) = \left\{(\rho\cos\phi, \rho\sin\phi) : \rho = \frac{a + b\cos^2\phi}{a+b}, \phi \in [0, 2\pi)\right\} \subset \mathbf{R}^2$$

Consider $X = \mathbf{P}(1,3)$ as shown in Figure 3.1(a). Let $x' = (0, -\frac{1}{4})$. Then $d_{x'}\,|_X$ has at least two local minima, $(0, -\frac{1}{4})$ and $(0, \frac{1}{4})$.

However, for $X \in \mathcal{C}$, it is not necessary that X be convex. $X = \mathbf{S}^1$ as shown in Figure 3.1(b) is an example.

The reader may wonder why the results for $\mathbf{P}(1,3)$ and $\mathbf{S}^1$ are different as we know that they are homotopic. It is because we have used direct distance instead of geodesic distance. (See Section 1.3.) We shall return to this question in Section 3.5.

□□□

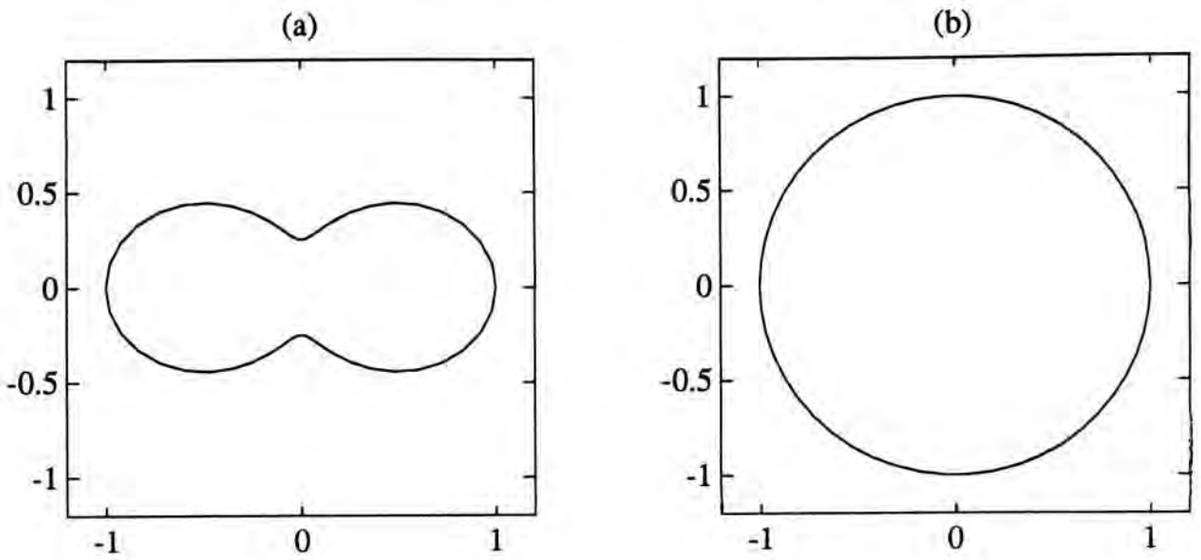

Figure 3.1: (a) $\mathbf{P}(1,3)$. (b) $\mathbf{S}^1$.

What we want to do is to construct a modified version of Corollary 3.2 which talks about local minima in Q instead of X. We proceed firstly to defining what we mean by a local minimum of a function defined on Q:

Definition 3.2 Let $f : Q \to \mathbf{R}$. We say that $q_i \in Q$ is a local minimum of f, or f has a local minimum at q_i, if $f(q_i) \leq f(q_j) \ \ \forall q_j \in N(q_i)$, the 1-neighborhood of q_i. We say that $q_i \in Q$ is a global minimum of f, or f has a global minimum at q_i, if $f(q_i) \leq f(q_j) \ \ \forall q_j \in Q$. Note that a global minimum is obviously a local minimum.

□

Remark 3.2 Comparing Definition 3.2 with Definition 3.1 the reader may notice the requirement of a local minimum in Q is stronger than the requirement of a local minimum in X. The former requires the 1-neighborhood while the latter requires $\exists$ a neighborhood U only. It is

because we are now considering (Q, N) as a graph instead of merely a topological space. (See Remark 1.2.)

□□□

Secondly we note that $d_{x'} \circ w$ (with x' fixed) is a real-valued function defined on Q, where $\circ$ denotes the composition. Being motivated by Corollary 3.2, we give the following definition

Definition 3.3 Suppose X is convex. We say that (Q, N) with w preserves the topological order of X if

[T1] $w(Q) = \{w(q_i) : q_i \in Q\} \subset X$

[T2] every local minimum of $d_{x'} \circ w$ is a global minimum of $d_{x'} \circ w$ for all $x' \in X$

□

We have used the weaker form since, not hard to see, it is too strong to require that $d_{x'} \circ w$ has a unique local minimum for all $x' \in X$. It is simply because the global minimum of $d_{x'} \circ w$ may not be unique for some $x' \in X$, even when X is convex. However, in general $d_{x'} \circ w$ has a unique global minimum for almost all $x' \in X$. Thus it is reasonable to require that $d_{x'} \circ w$ has a unique local minimum for almost all $x' \in X$. As mentioned in Section 1.3, Condition **[T1]** and **[T2]** on X will practically be transformed to conditions on $\tilde{x}$. Since $\tilde{x}$ has finite length, it is highly probable that $d_{\tilde{x}_l} \circ w$ has a unique global minimum for all $\tilde{x}_l$, and thus workable to require that $d_{\tilde{x}_l} \circ w$ also has a unique local minimum for all $\tilde{x}_l$. We make the former into an assumption.

Assumption 2 Practically we assume that for all $\tilde{x}_l$,

$$d_{\tilde{x}_l}(w(q_i)) \neq d_{\tilde{x}_l}(w(q_j)) \quad \text{if} \quad i \neq j$$

and hence $d_{\tilde{x}_l} \circ w$ has a unique global minimum.

□

With this assumption, we proceed to a practical version of Definition 3.3. Define

$$\gamma_{x'}(q_j) = \begin{cases} 1 & \text{if} \quad q_j \text{ is a local minimum of } d_{x'} \circ w \\ 0 & \text{if} \quad q_j \text{ is not a local minimum of } d_{x'} \circ w \end{cases}$$

and let

$$c(x') = \sum_{j=1}^{K} \gamma_{x'}(q_j)$$

be the number of local minima of $d_{x'} \circ w$ in Q. Assuming $\tilde{x}$ "sufficiently represents" X, we would adopt the following practice:

Practice 1 Suppose we know $X \in \mathcal{C}$. We would say that (Q, N) with w preserves the topological order of X if

[$\tilde{\mathbf{T}}$1] $\max_{q_i} \min_{\tilde{x}_l} d(w(q_i), \tilde{x}_l)$ is sufficiently small

[$\tilde{\mathbf{T}}$2] $c \circ \tilde{x} = e$, where $e = \underbrace{(1, 1, \ldots, 1)}_{M}$

□

Of course even when (Q, N) with w preserves the topological order of X in the sense of Definition 3.3, Assumption 2 may not be true and therefore **[$\tilde{\mathbf{T}}$2]** may not hold. However, since $\tilde{x}$ has finite length, it is in general not difficult to perturb w slightly to make **[$\tilde{\mathbf{T}}$2]** hold. Hence Practice 1 is acceptable.

Instead of checking whether $c \circ \tilde{x} = e$, we shall more often try to minimize

$$\|c \circ \tilde{x} - e\|$$

Since the space of finite sequences is finite dimensional, all norms on it are equivalent. (See [13, page 75].) Here we use the 1-norm. Define

$$J_1 = \frac{\|c \circ \tilde{x} - e\|_1}{M} = \frac{\sum_{l=1}^{M} |c(\tilde{x}_l) - 1|}{M} \tag{3.1}$$

Since in our case $c(\tilde{x}_l) \geq 1$ for all $\tilde{x}_l$, we may rewrite Equation 3.1 as

$$J_1 = \frac{\sum_{l=1}^{M} c(\tilde{x}_l)}{M} - 1$$

In the following sections we shall try to minimize J_1.

3.2 Minimizing the number of local minima

The most natural problem we hope to solve is

Problem 1 Given $\tilde{x}$ and (Q, N), find a mapping w such that $J_1 = 0$.

□

It is of course not easy to solve Problem 1. In this section we try to give a partial solution to this problem.

First of all, we try to solve Problem 1 directly using gradient descent method. We want to write J_1 explicitly as a sum of step functions, which count the number of local minima. Then we smooth J_1 to $\tilde{J}_1$ by sigmoid functions. Hence we can compute the gradient of $\tilde{J}_1$, and use gradient descent method to minimize it. For simplicity we shall only consider $(Q, N) = \mathbf{R}_K$ or $(Q, N) = \mathbf{S}_K^1$.

Let

$$\theta(x) = \begin{cases} 1 & \text{if} \quad x \geq 0 \\ 0 & \text{if} \quad x < 0 \end{cases}$$

be the step function. For convenience we introduce the following two symbols. Let

$$a_{l,j} = \begin{cases} \theta([d(w(q_{j+1}), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad 1 \leq j \leq K-1 \\ 1 & \text{if} \quad j = K \end{cases}$$

$$b_{l,j} = \begin{cases} \theta([d(w(q_{j-1}), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad 2 \leq j \leq K \\ 1 & \text{if} \quad j = 1 \end{cases}$$

if $(Q, N) = \mathbf{R}_K$ or

$$a_{l,j} = \begin{cases} \theta([d(w(q_{j+1}), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad 1 \leq j \leq K-1 \\ \theta([d(w(q_1), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad j = K \end{cases}$$

$$b_{l,j} = \begin{cases} \theta([d(w(q_{j-1}), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad 2 \leq j \leq K \\ \theta([d(w(q_K), \tilde{x}_l)]^2 - [d(w(q_j), \tilde{x}_l)]^2) & \text{if} \quad j = 1 \end{cases}$$

if $(Q, N) = \mathbf{S}_K^1$. The reader may easily note that

$$a_{l,j} b_{l,j} = \gamma_{\tilde{x}_l}(q_j)$$

Then we may write

$$J_1 = \frac{1}{M} \sum_{l=1}^{M} \sum_{j=1}^{K} a_{l,j} b_{l,j} - 1$$

Suppose now we use the sigmoid function

$$\tilde{\theta}(x) = \frac{1}{1 + \exp(-x)}$$

to approximate the original step function $\theta(x)$. It is well-known that $\tilde{\theta}(x)$ is smooth and

$$\frac{d\tilde{\theta}}{dx} = \tilde{\theta}[1 - \tilde{\theta}] \tag{3.2}$$

Let $\tilde{a}_{l,j}$ and $\tilde{b}_{l,j}$ be the corresponding approximations of $a_{l,j}$ and $b_{l,j}$ respectively. Then the smooth approximation of J_1 can be written as

$$\tilde{J}_1 = \frac{1}{M}\sum_{l=1}^{M}\sum_{j=1}^{K}\tilde{a}_{l,j}\tilde{b}_{l,j} - 1$$

By applying Equation 3.2 it is easy to obtain the gradient:

$$\nabla_{w(q_i)}\tilde{J}_1 = \frac{2}{M}\sum_{l=1}^{M}\alpha_{l,i}[x_l - w(q_i)]$$

where the coefficient

$$\alpha_{l,i} = \tilde{a}_{l,i}\tilde{b}_{l,i}[2 - \tilde{a}_{l,i} - \tilde{b}_{l,i}] - \tilde{a}_{l,i-1}\tilde{b}_{l,i-1}[1 - \tilde{a}_{l,i-1}] - \tilde{a}_{l,i+1}\tilde{b}_{l,i+1}[1 - \tilde{b}_{l,i+1}]$$

Hence the learning rule is

$$w^+(q_i) = w(q_i) - \eta\nabla_{w(q_i)}\tilde{J}_1 = w(q_i) - \eta\frac{2}{M}\sum_{l=1}^{M}\alpha_{l,i}[x_l - w(q_i)] \qquad (3.3)$$

where the superscript + means "after updated". In our simulations the learning rate η was chosen to be 0.05 and the initial values of $w(q_i)$'s were set to the first K training patterns.

Remark 3.3 The reader may notice how the "lateral interactions" appear as terms in $\alpha_{l,i}$. Information is passed directly from each neuron to its 1-neighborhood. Therefore we may say that each link between two neurons does really represent a connection between them, and it is not necessary for the neurons to be fully interconnected.

□□□

Example 3.1 The input space $X = [-1, 1]$. There are totally 7 neurons and 500 training patterns. Each training pattern is presented only once.

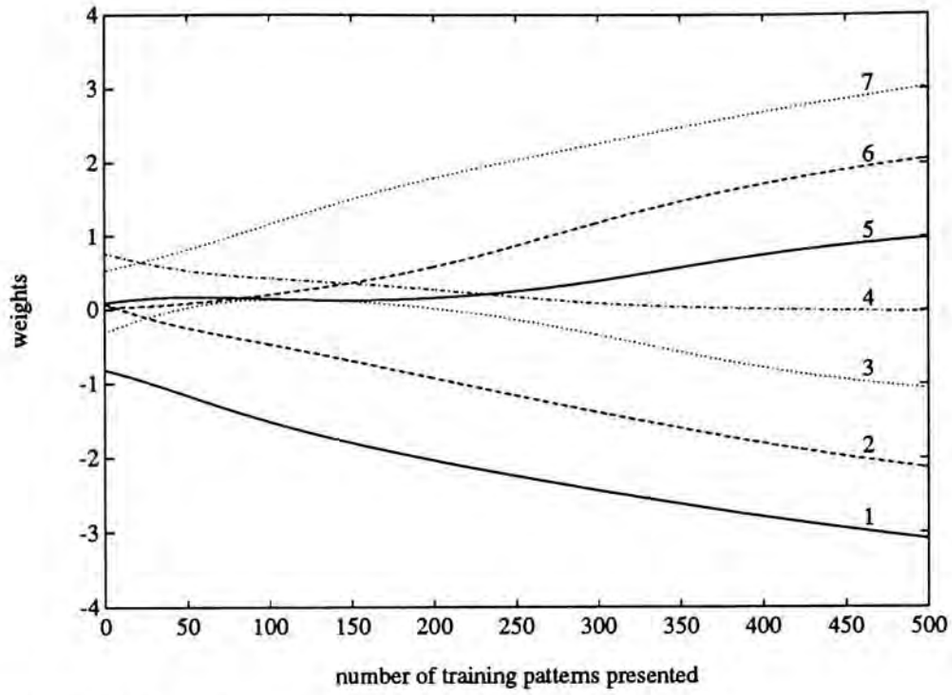

Figure 3.2: The evolution of $w(Q)$ in Example 3.1.

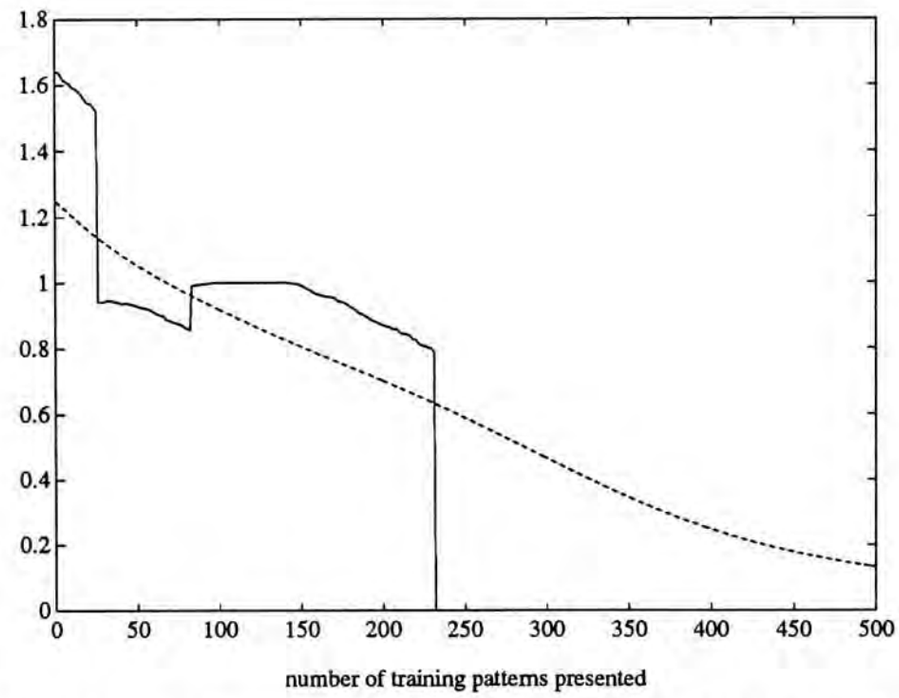

Figure 3.3: The evolutions of J_1 (solid line) and $\tilde{J}_1$ (dashed line) in Example 3.1.

Figure 3.2 plots $w(q_i)$'s, which are scalars in this example, vs the number of training patterns presented. Figure 3.3 plots J_1 (solid line) and $\tilde{J}_1$ (dashed line) vs the number of training patterns presented. We see that $\tilde{J}_1$ is in general decreasing.

□□□

Remark 3.4 Learning rule 3.3 is a "static" learning rule, in the sense that $\Delta w(q_i)$ (where Δ means the difference between "after updated" and "before updated") is a function of $w(q_i)$ only. All training patterns contribute simultaneously in each iteration. In practice one often prefers a "dynamic" learning rule. In that case each training pattern $\tilde{x}_l$ is presented sequentially, and $\Delta w(q_i)$ is also a function of $\tilde{x}_l$. Simulations showed that if we used

$$w^+(q_i) = w(q_i) - \eta 2\alpha_{l,i}[x_l - w(q_i)] \tag{3.4}$$

instead of Learning rule 3.3, there were almost no differences. (See Figure 3.4 and Figure 3.5.)

□□□

The results of our simulations showed that although Learning rule 3.3 could minimize J_1, it could not keep $w(Q)$ on X. The weights tended to spread over $\mathbf{R}^n$. It was because

1. we did not incorporate the requirement $w(Q) \subset X$ in the learning rule, and it was easier to lower J_1 if $w(Q)$ was not confined to X;

2. we smoothed J_1 by sigmoid functions. As a result $\tilde{J}_1 \neq 0$ even when $J_1 = 0$ (as shown in Figure 3.3), and thus Learning rule 3.3 tried to lower $\tilde{J}_1$ further by spreading out the weights.

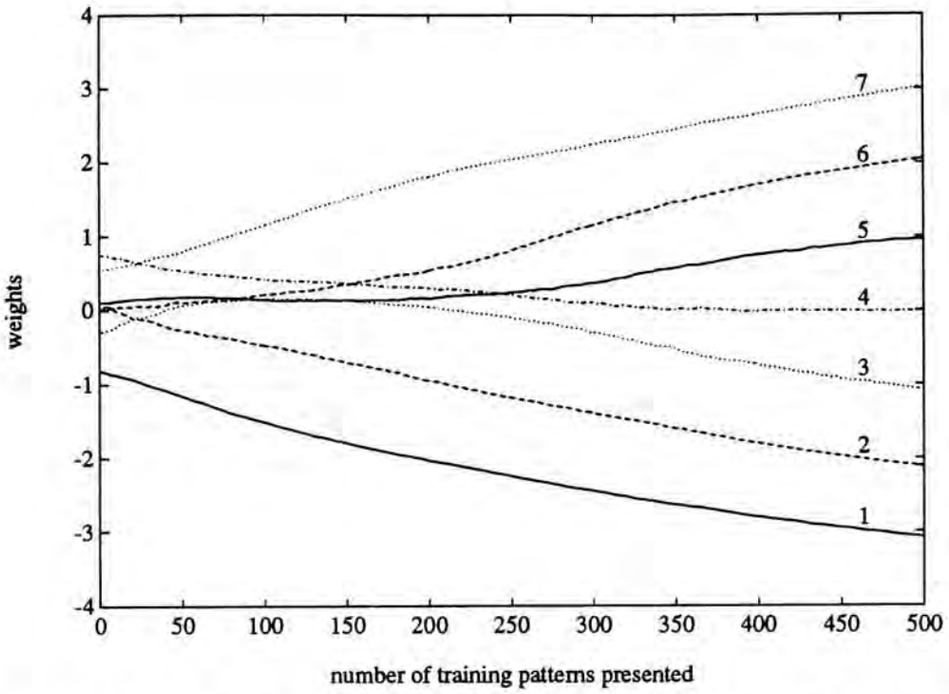

Figure 3.4: The evolution of $w(Q)$ when Learning rule 3.4 is used.

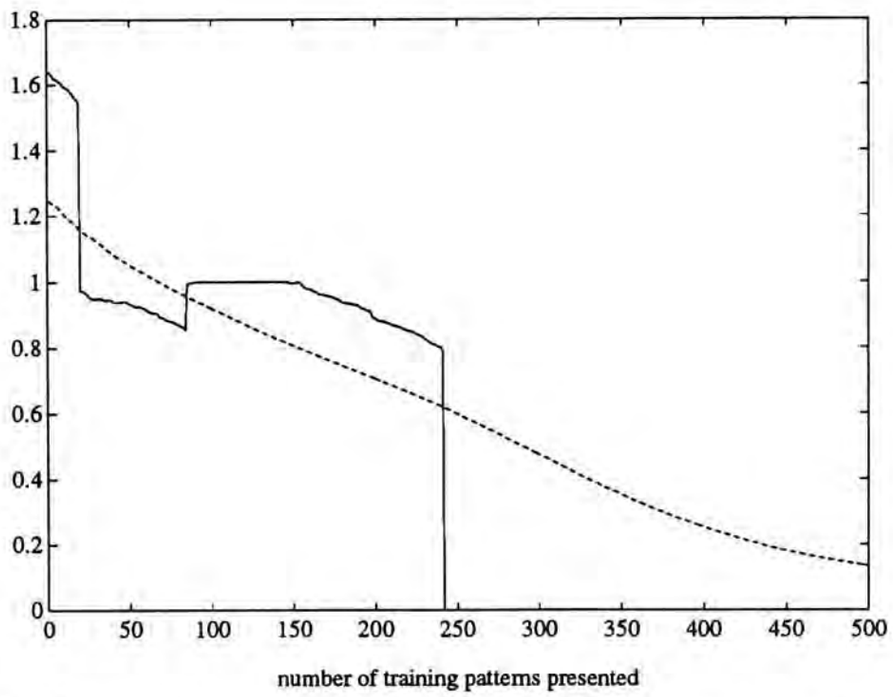

Figure 3.5: The evolutions of J_1 (solid line) and $\tilde{J}_1$ (dashed line) when Learning rule 3.4 is used.

Therefore Learning rule 3.3 is considered to be unsuccessful w.r.t. preserving the topological order, although it has given many interesting insights to us.

Next we proceed to a study on Kohonen's algorithm. It is well-known that practically Kohonen's algorithm is very effective in preserving the topological order, although it does not always solve Problem 1. It is not easy to analyze Kohonen's algorithm, and we summarize the basic results in the following proposition.

Proposition 3.3 Suppose $\tilde{x}_l$ is the current training pattern and q_i is the global minimum[2] of $d_{\tilde{x}_l} \circ w$. Denote the r-neighborhood of q_i by $N^{(r)}(q_i)$. Then Kohonen's learning rule is

$$w^+(q_j) = w(q_j) + \eta[\tilde{x}_l - w(q_j)] \quad \forall q_j \in N^{(r)}(q_i)$$

We have the following basic results:

[K1] $\gamma_{\tilde{x}_l}(q_i) = 1$

[K2] $\forall q_j \in N^{(1)}(q_i) \backslash \{q_i\}, \ \gamma_{\tilde{x}_l}(q_j) = 0$

[K3] $\forall q_j \in Q \backslash N^{(r+1)}(q_i), \ \Delta\gamma_{\tilde{x}_l}(q_j) = 0$

[K4] $\forall q_j \in N^{(r-1)}(q_i), \ \Delta\gamma_{\tilde{x}_l}(q_j) = 0$

[K5] $\forall q_j \in N^{(r+1)}(q_i) \backslash N^{(r)}(q_i), \ \Delta\gamma_{\tilde{x}_l}(q_j) \leq 0$

[K6] $\forall q_j \in N^{(r)}(q_i) \backslash N^{(r-1)}(q_i), \ \Delta\gamma_{\tilde{x}_l}(q_j) \geq 0$

[2] As stated in Assumption 2, global minimum of $d_{\tilde{x}_l} \circ w$ is assumed to be unique for all $\tilde{x}_l$. So we say "the" global minimum.

Proof. **[K1]**, **[K2]**, **[K3]** are obvious from the definition of r-neighborhood, and **[K4]**, **[K5]**, **[K6]** are also easily obtained if we observe that

$$d_{\bar{x}_l}(w^+(q_j)) = [1-\eta]d_{\bar{x}_l}(w(q_j)) \quad \forall q_j \in N^{(r)}(q_i)$$

□

We remark that Proposition 3.3 talks about the effect of a single training pattern only. It is too complex to consider the local minima of all $d_{\bar{x}_l} \circ w$, $l = 1, 2, \ldots, M$.

In general the contribution of **[K6]** does not balance the contribution of **[K5]**. It is very easy to have $\Delta\gamma_{\bar{x}_l}(q_j) < 0$ if $q_j \in N^{(r+1)}(q_i)\backslash N^{(r)}(q_i)$, but it is not so easy to have $\Delta\gamma_{\bar{x}_l}(q_j) > 0$ if $q_j \in N^{(r)}(q_i)\backslash N^{(r-1)}(q_i)$. To make it precise, we give the following two propositions. Note that the "if" parts of these two propositions may easily hold in the sense that they consists of "$\exists$" and "or" only.

Proposition 3.4 Suppose $q_j \in N^{(r+1)}(q_i)\backslash N^{(r)}(q_i)$. Then $\Delta\gamma_{\bar{x}_l}(q_j) < 0$ if

[K5-1] $\exists q_k \in N(q_j) \cap N^{(r)}(q_i)$ s.t. $[1-\eta]d_{\bar{x}_l}(w(q_k)) < d_{\bar{x}_l}(w(q_j))$

□

i.e. if η is sufficiently close to 1, $\Delta\gamma_{\bar{x}_l}(q_j)$ is surely < 0. On the other hand,

Proposition 3.5 Suppose $q_j \in N^{(r)}(q_i)\backslash N^{(r-1)}(q_i)$. Then $\Delta\gamma_{\bar{x}_l}(q_j) \not> 0$ if any of the following conditions holds

[K6-1] $\exists q_k \in N(q_j) \cap N^{(r)}(q_i)$ s.t. $d_{\bar{x}_l}(w(q_k)) < d_{\bar{x}_l}(w(q_j))$

[K6-2] $\exists q_k \in N(q_j)\backslash N^{(r)}(q_i)$ s.t. $d_{\bar{x}_l}(w(q_k)) < [1-\eta]d_{\bar{x}_l}(w(q_j))$

□

Note that even when η is very close to 1, **[K6–1]** may still hold and as a result $\Delta\gamma_{\tilde{x}_l}(q_j) \not> 0$.

3.3 Detecting the success or failure of Kohonen's algorithm

In this section we introduce an application of J_1. It is known that Kohonen's algorithm does not always preserve the topological order successfully. For low dimensions we may plot $w(Q, N)$ to see whether Kohonen's algorithm succeeds or not, as what we have done before. However, we cannot use this method in high dimensions. Now we may measure J_1 to determine whether Kohonen's algorithm succeeds or not. In the following examples we monitor J_1 after each training pattern is presented and a weight is updated using Kohonen's learning rule. Of course, this is only for illustration purpose. Practically we only need to measure J_1 at the end of training.

Example 3.2 The input space $X = [-1, 1]$. The total number of training patterns $M = 500$. (Q, N) is chosen to be $\mathbf{R}_{16}$. The parameters of Kohonen's algorithm are set as follows. If $\tilde{x}_l$ is the current training pattern, then the radius of neighborhood

$$r = \lfloor \frac{M - l}{M - 1} \times 7 \rfloor$$

The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

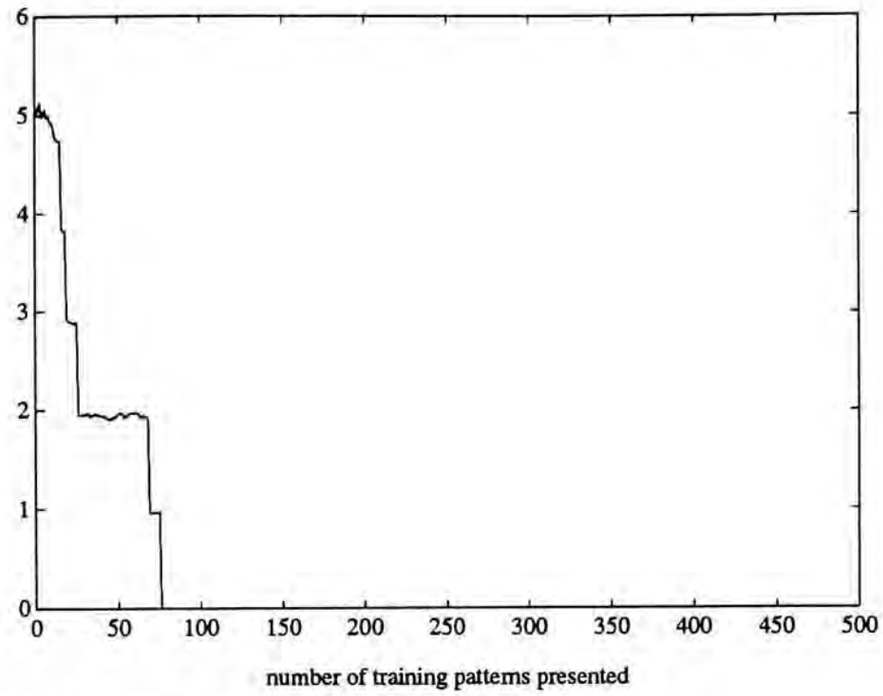

Figure 3.6: The evolution of J_1 in Example 3.2.

The result J_1 is shown in Figure 3.6. We see that Kohonen's algorithm succeeds in this example.

□□□

Example 3.3 The input space $X = \mathbf{S}^1$. $(Q, N) = \mathbf{S}^1_{16}$. All other parameters are set the same as those in Example 3.2. The result J_1 is shown in Figure 3.7. We see that Kohonen's algorithm also succeeds in this case. Then we plot $w(Q, N)$ in Figure 3.9(a) to verify our conclusion.

□□□

Example 3.4 Here we want to demonstrate a failed version of Example 3.3. The configuration is the same as that of Example 3.3, except that this time we use 1-Kohonen's algorithm and η is fixed to 0.15. The result J_1 is plotted in Figure 3.8. From it we conclude that 1-Kohonen's

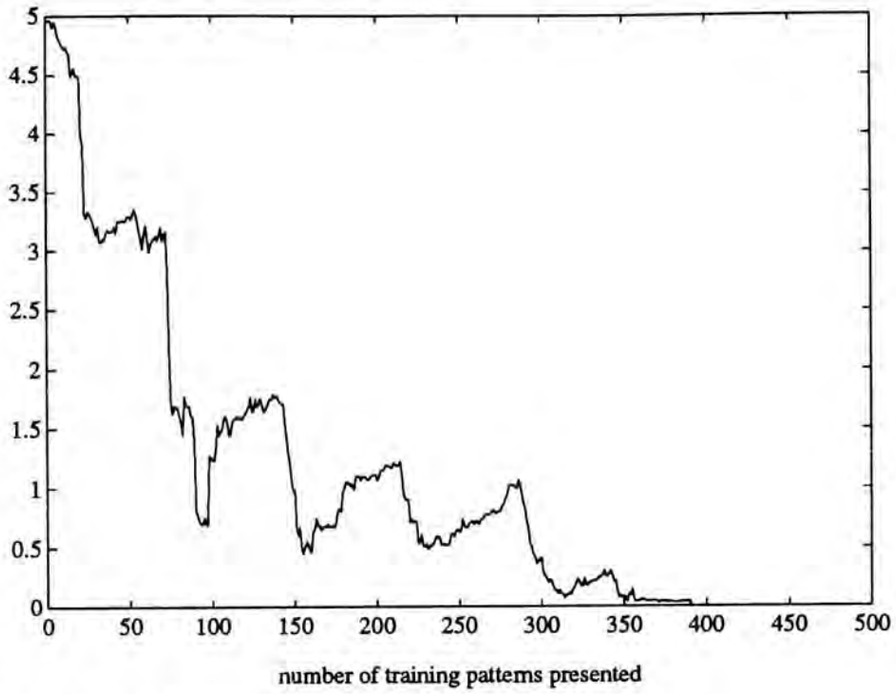

Figure 3.7: The evolution of J_1 in Example 3.3.

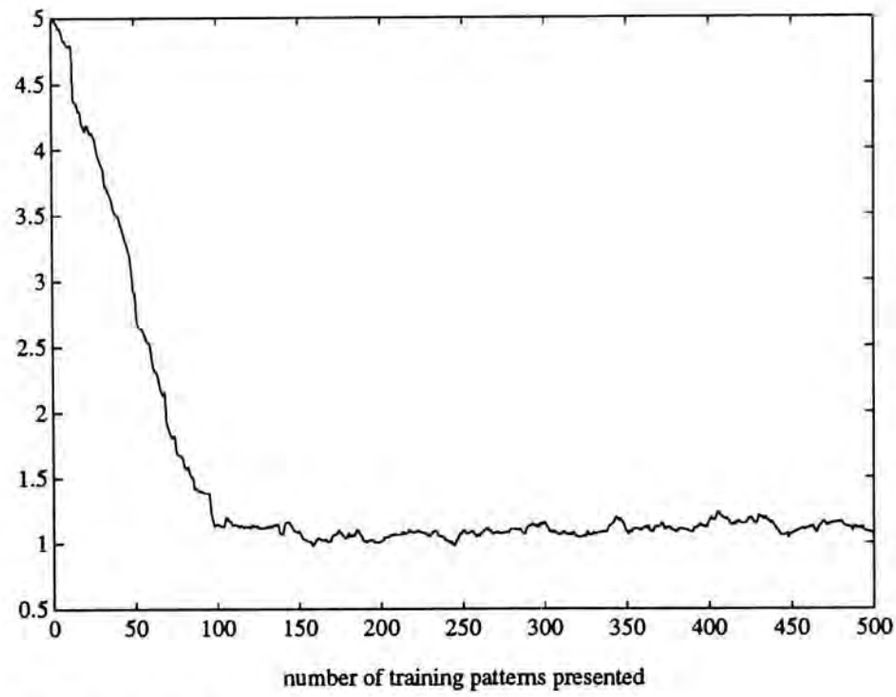

Figure 3.8: The evolution of J_1 in Example 3.4.

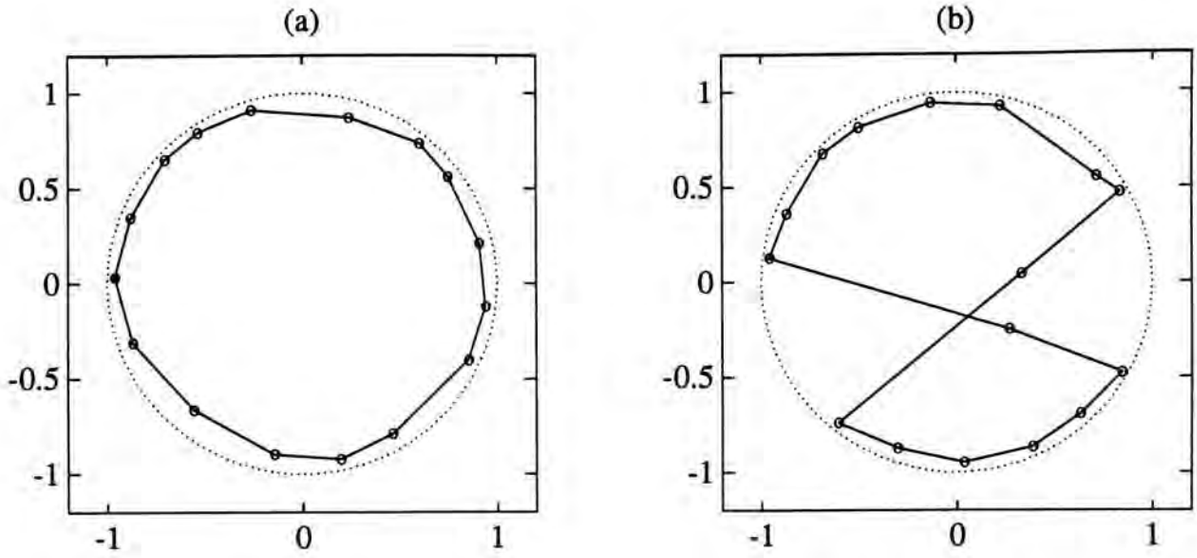

Figure 3.9: (a) $w(Q, N)$ in Example 3.3. (b) $w(Q, N)$ in Example 3.4.

algorithm fails in this example[3]. We also plot $w(Q, N)$ in Figure 3.9(b) to verify our conclusion.

□□□

Example 3.5 The input space $X = [-1, 1]^2$. The total number of training patterns $M = 1000$. (Q, N) is chosen to be $\mathbf{R}^2_{64}$. The radius of neighborhood

$$r = \lfloor \frac{M - l}{M - 1} \times 5 \rfloor$$

The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

The result J_1 is shown in Figure 3.10. We see that Kohonen's algorithm

[3]It is known that whether Kohonen's algorithm works or not depends on some random factors, such as the initial weights and the sequence of training patterns. Therefore, we can only say that it does not work in this example, but cannot say that it does not work for this configuration.

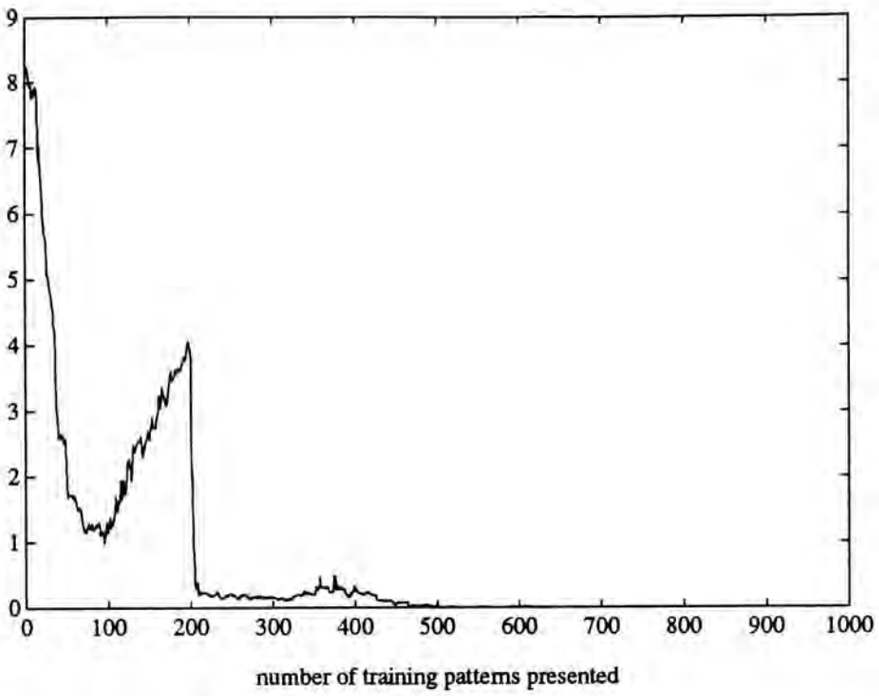

Figure 3.10: The evolution of J_1 in Example 3.5.

succeeds in this example. We also plot $w(Q, N)$ in Figure 3.12(a) to verify our conclusion. (It can be seen from our choice of r that, since we are now studying the topological order, we focus on the ordering phase of Kohonen's algorithm only. As a result, the grid shown in Figure 3.12(a) has not been completely extended.)

□□□

Example 3.6 Here we demonstrate a failed version of Example 3.5. The configuration is the same as that of Example 3.5 except that $M = 500$ and

$$r = \lfloor \frac{M - l}{M - 1} \times 4 \rfloor$$

The result J_1 is shown in Figure 3.11. From it we conclude that Kohonen's algorithm fails in this example. We also plot $w(Q, N)$ in Figure 3.12(b) to

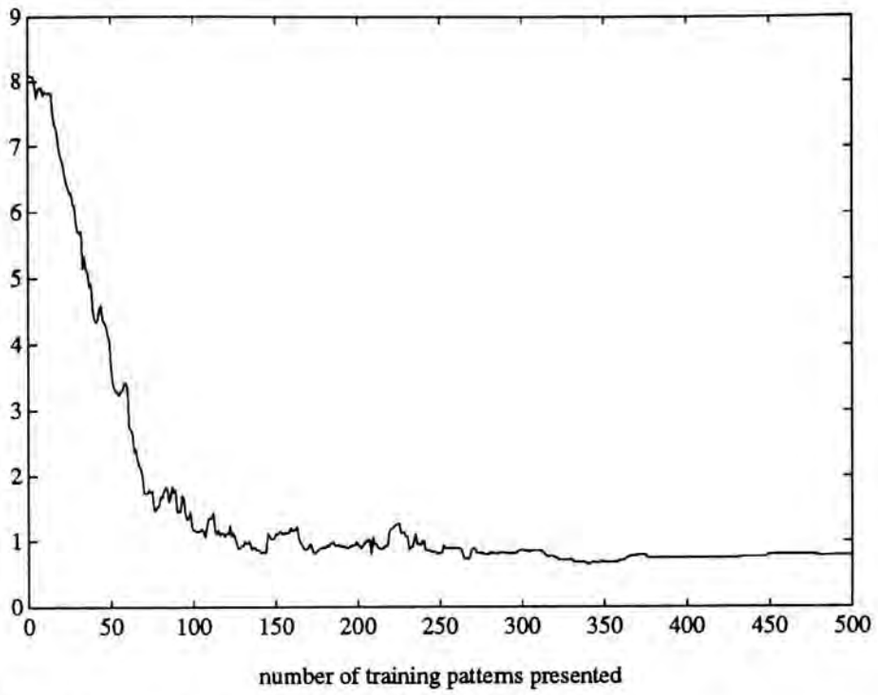

Figure 3.11: The evolution of J_1 in Example 3.6.

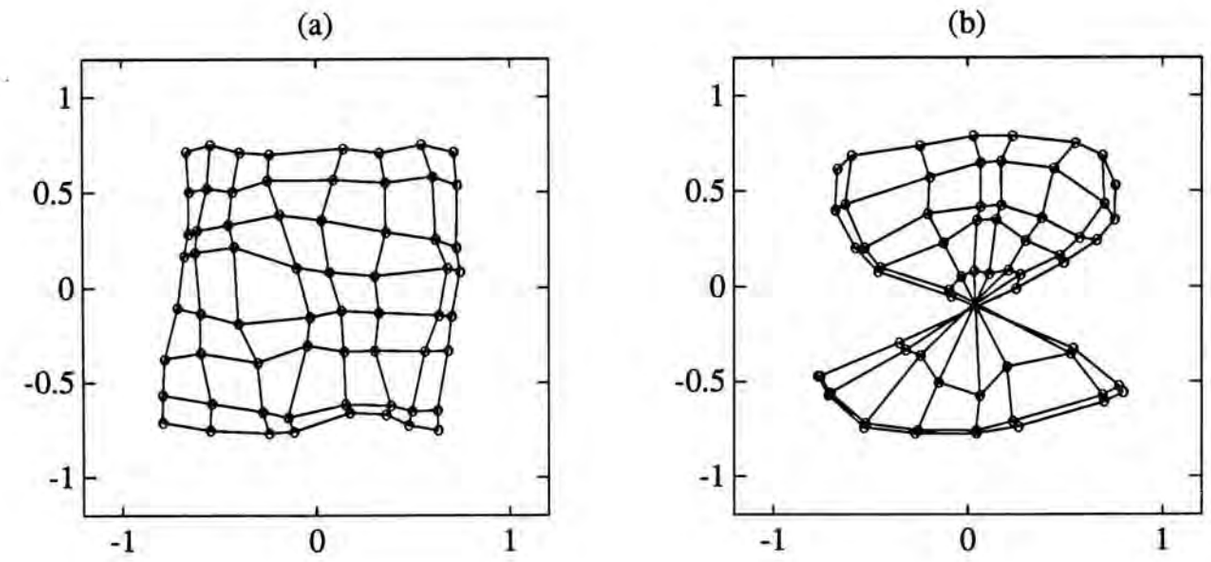

Figure 3.12: (a) $w(Q, N)$ in Example 3.5. (b) $w(Q, N)$ in Example 3.6.

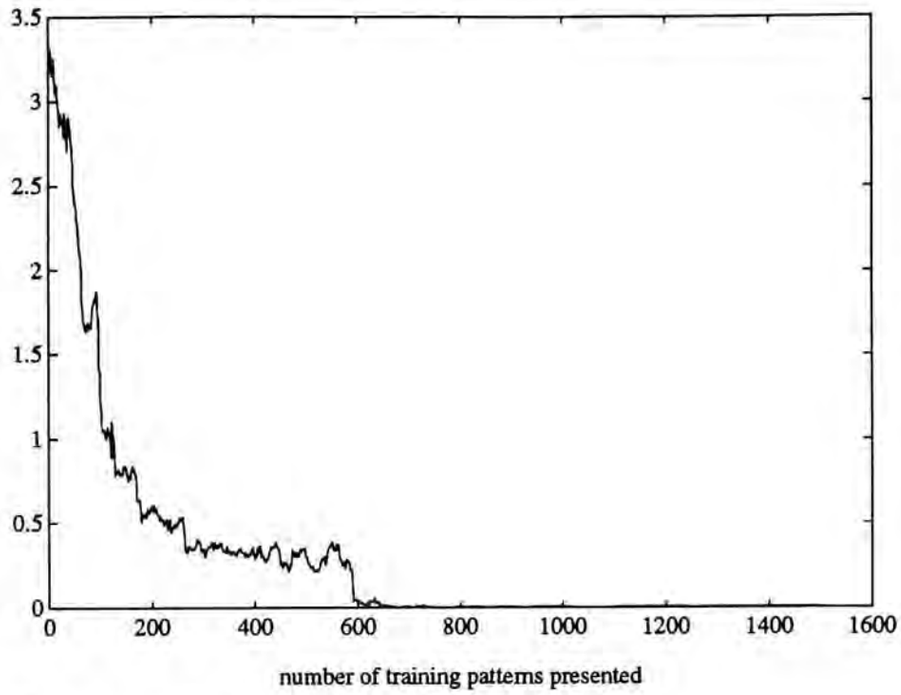

Figure 3.13: The evolution of J_1 in Example 3.7.

verify our conclusion. Note that it is a classic way of failure of Kohonen's algorithm.

□□□

Example 3.7 The input space $X = [-1, 1]^3$. The total number of training patterns $M = 1500$. (Q, N) is chosen to be $\mathbf{R}^3_{64}$. The radius of neighborhood

$$r = \lfloor \frac{M - l}{M - 1} \times 2 \rfloor$$

The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

The result J_1 is shown in Figure 3.13. We see that Kohonen's algorithm succeeds in this example. Note that this time we cannot plot $w(Q, N)$ to verify our conclusion.

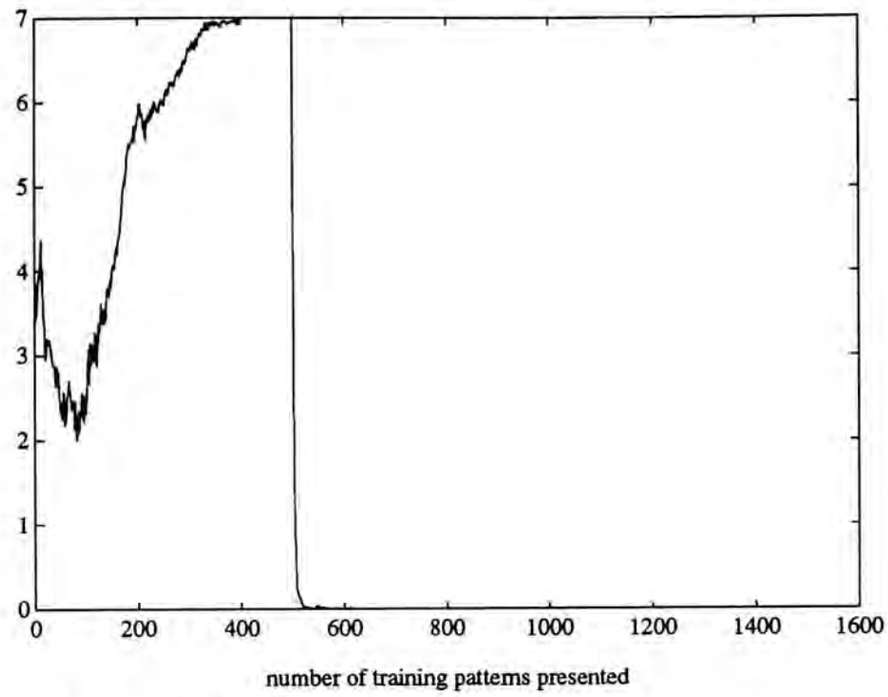

Figure 3.14: The evolution of J_1 in Example 3.8.

□□□

Example 3.8 In this example we want to demonstrate what happens when the neighborhood is set too large. The configuration is the same as that of Example 3.7 except that

$$r = \lfloor \frac{M-l}{M-1} \times 3 \rfloor$$

The result J_1 is plotted in Figure 3.14. It seems that J_1 becomes saturated before it starts to decrease again. The same phenomenon is observed for other dimensions of input space, provided that the initial r is set sufficiently large.

□□□

The above examples have one point in common. That is, $J_1 = 0$ is achievable for some w, although Kohonen's algorithm cannot always find it out. In the

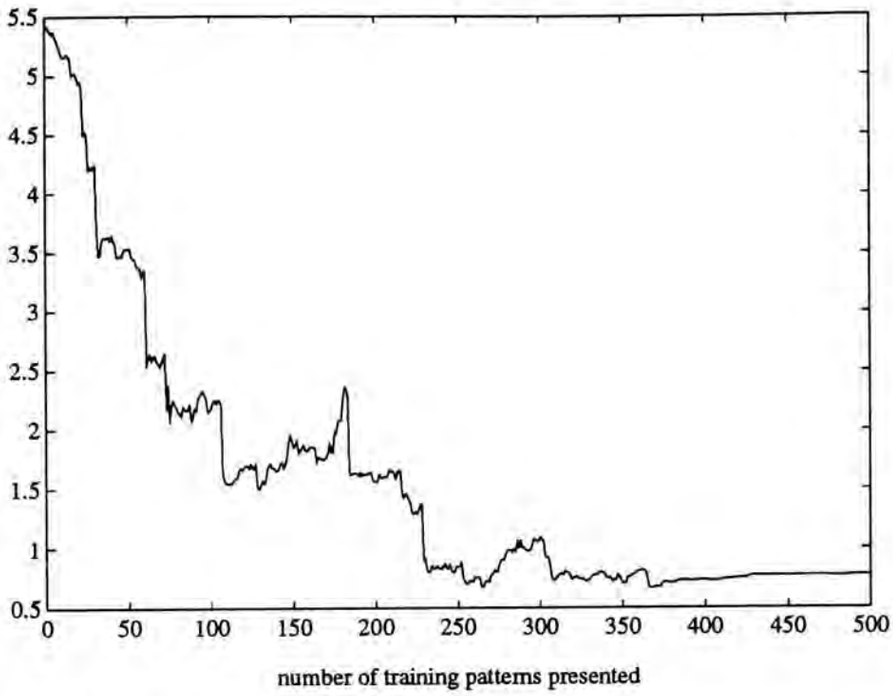

Figure 3.15: The evolution of J_1 in Example 3.9.

examples below, $J_1 = 0$ is not achievable, and we shall see how Kohonen's algorithm manages such cases.

Example 3.9 The input space $X = \mathbf{S}^1$ but $(Q, N) = \mathbf{R}_{16}$. Other parameters are set the same as those in Example 3.2. The results J_1 and $w(Q, N)$ are shown in Figure 3.15 and Figure 3.17(a) respectively. We see that although only one link is missed, J_1 can detect the difference.

□□□

Example 3.10 The input space $X = [-1, 1]$ but $(Q, N) = \mathbf{S}^1_{16}$. Other parameters are set the same as those in Example 3.9. The result J_1 is shown in Figure 3.16. Note that the final value of J_1 is quite close to that of Example 3.9.

□□□

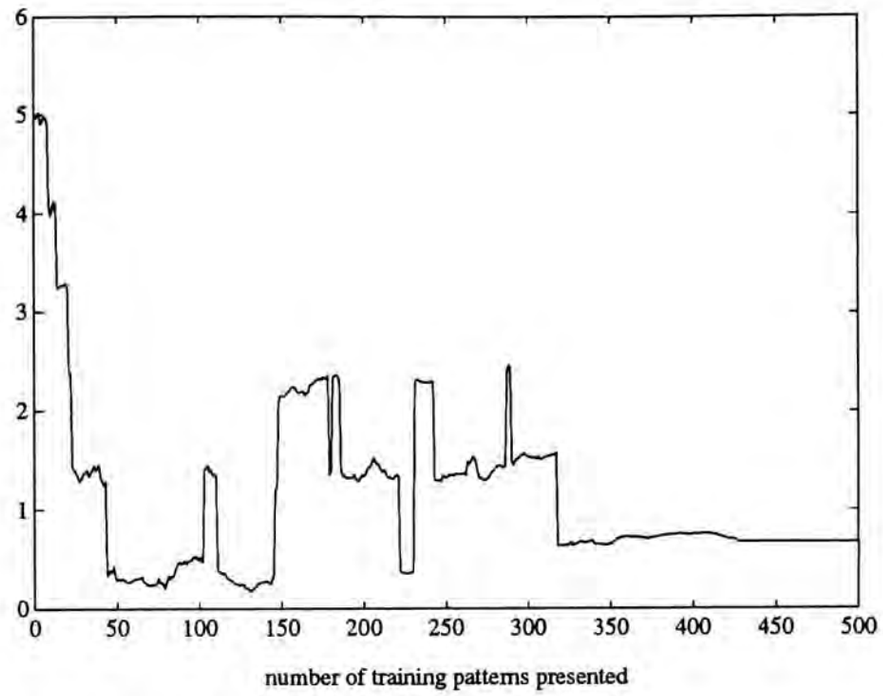

Figure 3.16: The evolution of J_1 in Example 3.10.

Example 3.11 In this example the input space $X = [-1, 1]^2$. The total number of training patterns $M = 10000$. (Q, N) is chosen to be $\mathbf{S}_{64}^1$. The radius of neighborhood

$$r = \begin{cases} \lfloor \frac{1000-l}{1000-1} \times 16 \rfloor & \text{if} \quad l \leq 1000 \\ 0 & \text{if} \quad l > 1000 \end{cases}$$

The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

$w(Q, N)$ is plotted in Figure 3.17(b) . Note that the result is classic. The final value of J_1 is 14.0475, which is far away from 0.

□□□

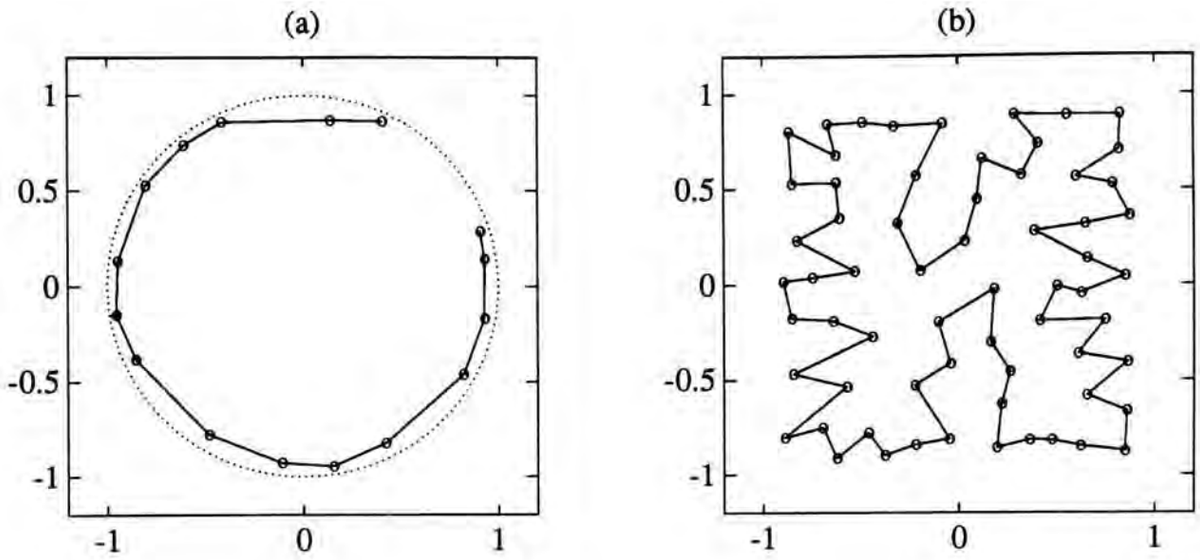

Figure 3.17: (a) $w(Q, N)$ in Example 3.9. (b) $w(Q, N)$ in Example 3.11.

Remark 3.5 The reader may readily notice that in most of the time we used $\mathbf{R}_K^n$ instead of $n\mathbf{R}_K$ in Kohonen's algorithm for the measurement of J_1. Besides the traditional reason, there is another more important reason. A neuron is much easier to be a local minimum if $(Q, N) = n\mathbf{R}_K$ than if $(Q, N) = \mathbf{R}_K^n$ when $n > 1$. This is in fact a special case of what we shall discuss in Section 3.4. Roughly speaking, for $n\mathbf{R}_K$ to preserve the topological order of $\mathbf{R}^n$, $w(Q)$ should exactly form a *rectangular* grid. Even a small perturbation in w may make $J_1 \neq 0$ when $n > 1$. Since our requirement is not so strong, it would be better to use $\mathbf{R}_K^n$ instead of $n\mathbf{R}_K$ for the measurement of J_1. However, theoretically $n\mathbf{R}_K$ is better than $\mathbf{R}_K^n$ in the sense that $n\mathbf{R}_K$ may be minimal but $\mathbf{R}_K^n$ is never minimal when $n > 1$. The meaning of this statement will become clear in the next two chapters.

□□□

3.4 Local minima for different graph structures

One may easily see from Definition 3.2 that it is more difficult for q_i to be a local minimum if $N(q_i)$ is larger. More precisely, if q_i is a local minimum in (Q, N), it may not be a local minimum in (Q, N_E) if $N_E(q_i) \supset N(q_i)$. Thus a natural but new problem arises.

Definition 3.4 Let $N_E : Q \to \mathcal{P}(Q)$. N is called a partial graph of N_E if

$$N_E(q_i) \supset N(q_i) \quad \forall q_i \in Q$$

We shall abbreviate this relation to $N_E \supset N$.

□

Problem 2 Given (Q, N), $\tilde{x}$ and w and suppose $J_1 \neq 0$, find a new graph (Q, N_E), $N_E \supset N$ and satisfies **[N2]**[4], such that (Q, N_E) with w would have $J_1 = 0$.

□

In practice w is not arbitrary given, but usually obtained using Kohonen's algorithm, so that N_E is hopefully close to N. Unlike Problem 1, this problem is obviously solvable with the trivial solution

$$N_E(q_i) = Q \quad \forall q_i \in Q$$

even if $X \notin \mathcal{C}$. Of course such a trivial solution is not useful, but it is still easy to find out a non-trivial solution. In Chapter 4 we shall introduce the *induced mapping* N_I and hence $N_E(q_i)$ can be easily set to $N(q_i) \cup N_I(q_i) \ \forall q_i \in Q$.

[4] $N_E \supset N$ implies N_E satisfies **[N1]**.

Since we have not introduced the induced mapping N_I, we would now use another systematic but natural approach:

> Suppose we have used Kohonen's algorithm to find out a w for which $J_1 \neq 0$. Then we go through the training patterns again. Let $\tilde{x}_l$ be the current training pattern and q_i be the global minimum of $d_{\tilde{x}_l} \circ w$. Suppose q_j, $j \neq i$ is also a local minimum. We would think that q_j should not be a local minimum since it is not the global minimum. So we enlarge $N(q_j)$ and $N(q_i)$ by
>
> $$\begin{aligned} N^+(q_j) &= N(q_j) \cup \{q_i\} \\ N^+(q_i) &= N(q_i) \cup \{q_j\} \end{aligned}$$
>
> ($N(q_i)$ is also enlarged to ensure that **[N2]** is satisfied.) Then q_j must not be a local minimum anymore. The process is repeated for all training patterns and at last we would have $J_1 = 0$.

With this method we present some examples. w is obtained using Kohonen's algorithm of which the parameters are set as follows. The total number of training patterns $M = 10000$. The radius of neighborhood

$$r = \begin{cases} \lfloor \frac{1000-l}{1000-1} \times \frac{K}{4} \rfloor & \text{if} \quad l \leq 1000 \\ 0 & \text{if} \quad l > 1000 \end{cases}$$

where K is the total number of neurons. The learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

Example 3.12 We start with an example in which $X \notin C$. Let $X = \mathbf{P}(1,3)$ and $(Q,N) = \mathbf{S}^1_{16}$, i.e. $K = 16$. Figure 3.18(a) shows that the

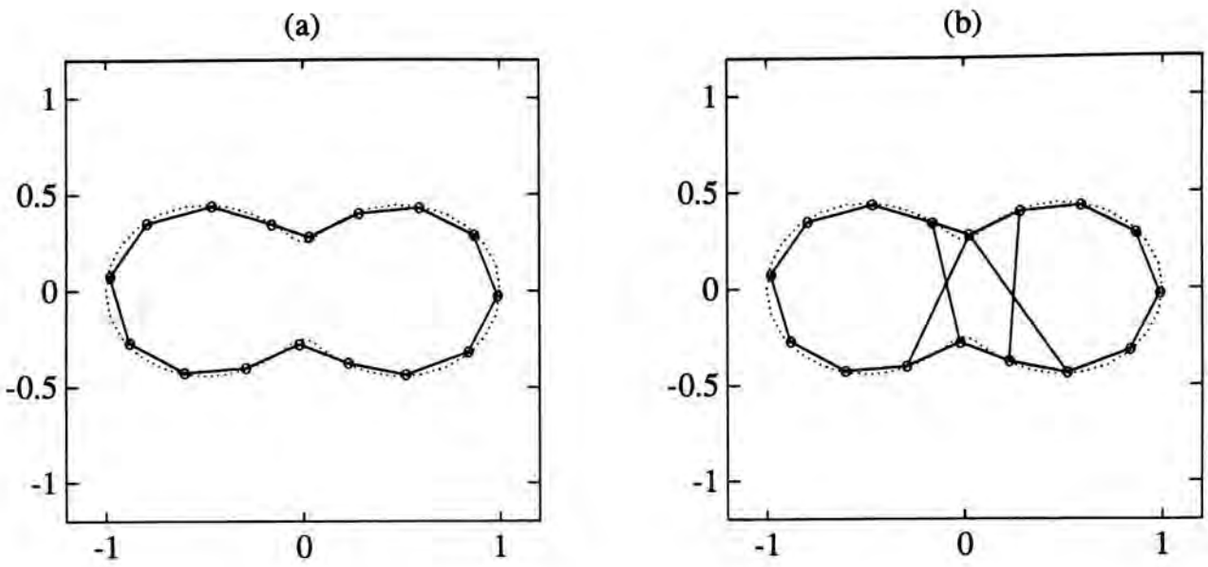

Figure 3.18: (a) $w(Q, N)$ in Example 3.12. (b) $w(Q, N_E)$ in Example 3.12.

result $w(Q, N)$ of Kohonen's algorithm is satisfactory. However, since $X \notin \mathcal{C}$, our theory does not work. J_1 is measured to be 0.5625 instead of 0. After using the above algorithm to find N_E, J_1 is forced to 0. The result $w(Q, N_E)$ is shown in Figure 3.18(b). Although we have not introduced the induced mapping N_I, we also plot $w(Q, \tilde{N}_I)$ in Figure 3.19 for comparison, where $\tilde{N}_I$ is the practical approximation of N_I. (The method to obtain $\tilde{N}_I$ is mentioned in Practice 2, which will be discussed in details in Chapter 4.)

□□□

Example 3.13 In this example $X = \mathbf{P}(0, 1)$ and $(Q, N) = \mathbf{S}^1_{16}$. Figure 3.20(a) shows the result $w(Q, N)$ of Kohonen's algorithm. J_1 is measured to be 1.0793. Then we obtain N_E using the above algorithm. $w(Q, N_E)$ is plotted in Figure 3.20(b). This time J_1 is measured to be 0. We also plot $w(Q, \tilde{N}_I)$ in Figure 3.21 for comparison.

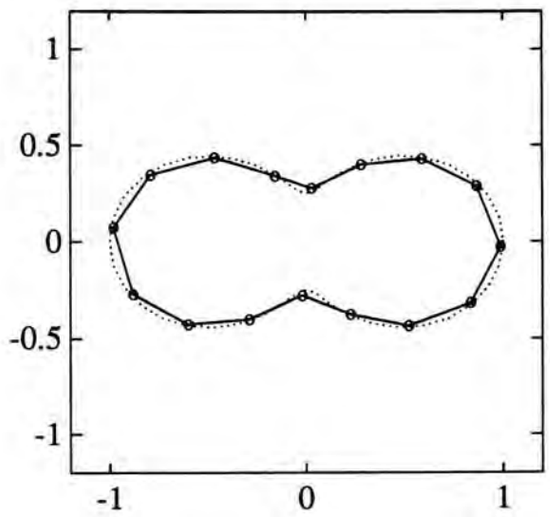

Figure 3.19: $w(Q, \tilde{N}_I)$ in Example 3.12.

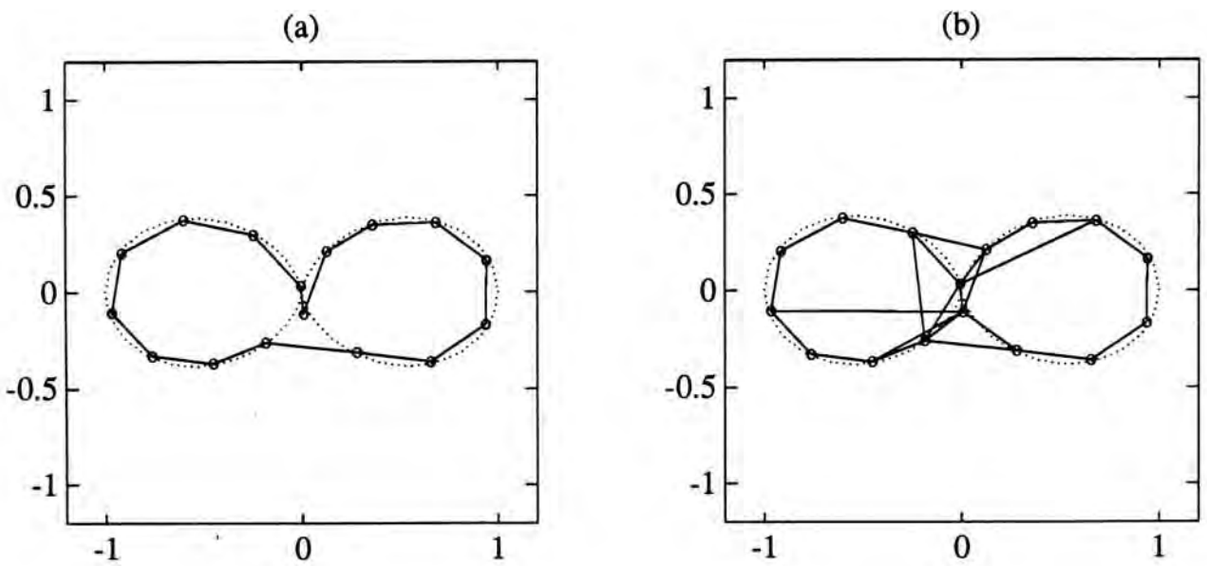

Figure 3.20: (a) $w(Q, N)$ in Example 3.13. (b) $w(Q, N_E)$ in Example 3.13.

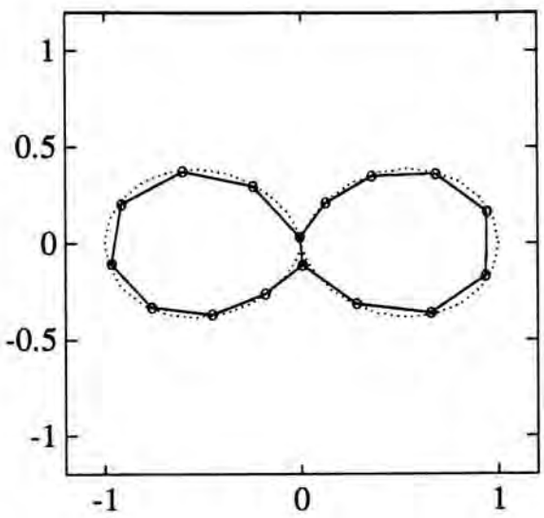

Figure 3.21: $w(Q, \tilde{N}_I)$ in Example 3.13.

□□□

Example 3.14 In this last example we find N_E for Example 3.11. The result $w(Q, N)$ of Kohonen's algorithm is already shown in Figure 3.17(b). Now we plot it again in Figure 3.22(a) for ease of comparison. $w(Q, N_E)$ and $w(Q, \tilde{N}_I)$ are plotted in Figure 3.22(b) and Figure 3.23 respectively. Note from Figure 3.22(b) that the "density" of links is especially high at some locations, which are in fact the locations of the first few training patterns. Actually it is not hard to see that in our algorithm the most significant changes in N are usually made by the first few training patterns, among which a lot are inadequate.

□□□

From the above examples we see that N_E has a lot of excess links, although it is not the trivial mapping N_Q. Such excess links are mainly produced by the

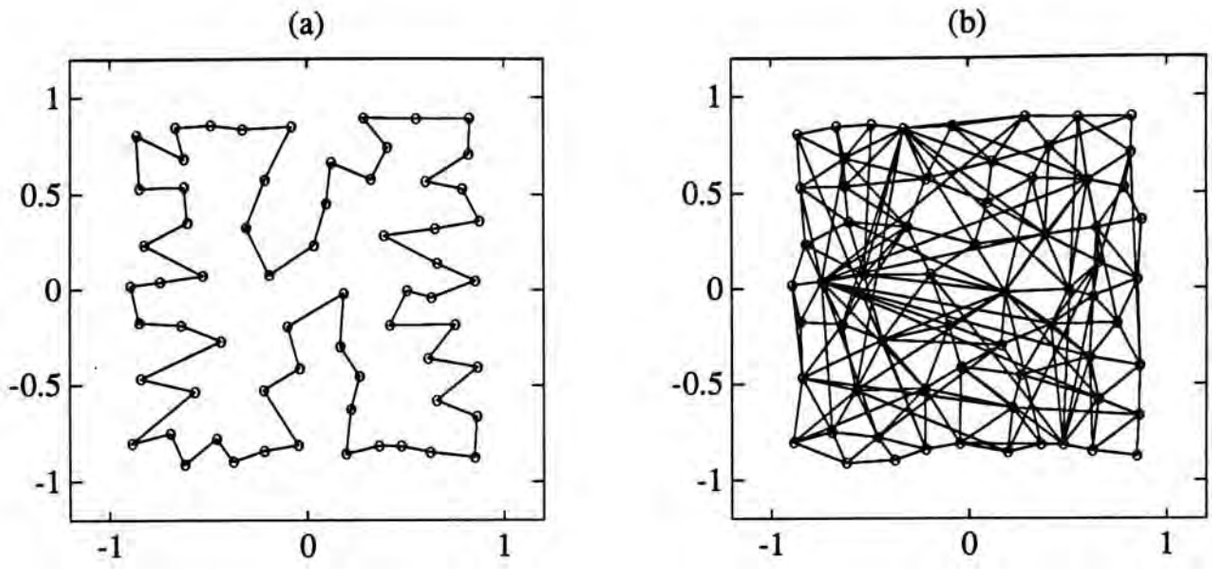

Figure 3.22: (a) $w(Q, N)$ in Example 3.14. (b) $w(Q, N_E)$ in Example 3.14.

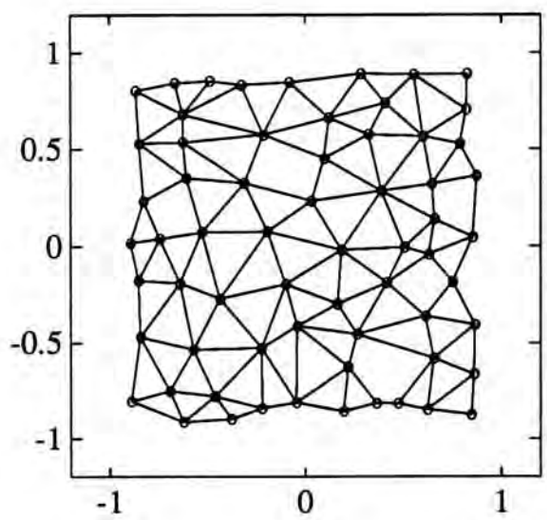

Figure 3.23: $w(Q, \tilde{N}_I)$ in Example 3.14.

first few training patterns. Thus we notice that "minimal" is very important, and we shall see in Chapter 4 that the induced mapping N_I is in some sense minimal. From the satisfactory results of the induced mapping shown in the above examples, we hope that there is a sufficient motivation for the reader to go through our work in Chapter 4.

3.5 Formulation by geodesic distance

In this section we want to generalize our theoretical work in Section 3.1.

The generalization is achieved by employing geodesic distance instead of direct distance, and as a result we do not need the convexity of X. However, to guarantee that geodesic distance is defined for any two points in X, it is still necessary to assume X is path-connected.

Let

$$\breve{d}(x, x') = \text{ geodesic distance between } x \text{ and } x'$$

Note that $\breve{d}$ is only defined on $X \times X$ instead of the whole $\mathbf{R}^n \times \mathbf{R}^n$. Similarly we may define

$$\breve{d}_{x'} : X \to \mathbf{R} \text{ such that } \breve{d}_{x'}(x) = \breve{d}(x, x')$$

Then we have the following generalization of Proposition 3.1

Proposition 3.6 Assume X is path-connected. Suppose $x' \in X$. Then $\breve{d}_{x'}$ has a unique local minimum at $x' \in X$.

> **Proof.** It is obvious that x' is a global minimum and hence a local minimum. To prove uniqueness, it suffices to show that given any $x \in X$ $(x \neq x')$ and any open neighborhood U of x, there exists

a point $x'' \in U$ s.t. $\breve{d}_{x'}(x'') < \breve{d}_{x'}(x)$. Since X is path-connected, there must exist a path joining x' and x, and since X is closed, the shortest path, i.e. the geodesic, must exist[5]. Let $\mathcal{G}$ be the trace[6] of the geodesic excluding the end points x' and x. Take $x'' \in \mathcal{G} \cap U \neq \emptyset$. Then $\breve{d}_{x'}(x'') < \breve{d}_{x'}(x)$ since

$$\breve{d}(x, x') = \breve{d}(x'', x') + \breve{d}(x, x'')$$

which is evident from the concept "shortest path" embedded in the definition of geodesic.

□

Assume $w(Q) \subset X$. Then we can still define $\breve{d}_{x'} \circ w$. Hence we obtain a generalization of Definition 3.3.

Definition 3.5 Suppose X is path-connected and $w(Q) \subset X$. We say that (Q, N) with w preserves the topological order of X if every local minimum of $\breve{d}_{x'} \circ w$ is a global minimum of $\breve{d}_{x'} \circ w$ for all $x' \in X$.

□

We remark that this definition is really consistent with Definition 3.3 in the sense that if X is convex, $\breve{d}_{x'} = d_{x'}$.

We have seen that if we use direct distance, we must at least restrict our class of input spaces to $\mathcal{C}$, otherwise undesirable results may be obtained. (See Example 3.12.) If we could measure geodesic distance, the results would be quite different. We could hence study a much larger class of input spaces. (It

[5]Geodesic may not be unique, so it may not be suitable to call it "the" geodesic. However, the geodesic distance is well defined.

[6]Strictly speaking a path is a map and its trace is the range of this map.

can be easily seen that $\mathcal{C}$ is contained in the class of path-connected spaces.) In Section 3.1, we mentioned that $\mathbf{S}^1 \in \mathcal{C}$ but $\mathbf{P}(1,3) \notin \mathcal{C}$, although they are homotopic. In Example 3.12 and Example 3.13, we saw that both $J_1 \neq 0$ when $(Q,N) = \mathbf{S}^1_K$, although $\mathbf{P}(1,3)$ and $\mathbf{P}(0,1)$ are not homotopic. However, in fact we know that $\mathbf{S}^1_K$ preserves the topological order of $\mathbf{S}^1$ and $\mathbf{P}(1,3)$, with different suitable w, in the sense of Definition 3.5, but $\mathbf{S}^1_K$ with no w would preserve the topological order of $\mathbf{P}(0,1)$ in the sense of Definition 3.5. Therefore, if we had used geodesic distance, we should notice the similarity between $\mathbf{S}^1$ and $\mathbf{P}(1,3)$ and the difference between $\mathbf{P}(1,3)$ and $\mathbf{P}(0,1)$.

3.6 Local minima and Voronoi regions

Finally we want to point out how the concepts developed in this chapter arrive at the birth of the main idea discussed in the next chapter, the induced mapping. It is based on an equivalent formulation of Definition 3.5 in terms of Voronoi regions. Suppose X is path-connected. Recall that the Voronoi region [8, page 225] of q_i is

$$B_i(Q) = \{x \in X : \breve{d}(w(q_i), x) \leq \breve{d}(w(q_j), x) \quad \forall q_j \in Q\}$$

Here we introduce a local version of Voronoi regions. Define

$$B_i(N(q_i)) = \{x \in X : \breve{d}(w(q_i), x) \leq \breve{d}(w(q_j), x) \quad \forall q_j \in N(q_i)\}$$

to be the local Voronoi region of q_i. It is clear from the definitions that $B_i(Q) \subset B_i(N(q_i))$. The equivalent formulation of Definition 3.5 is based on the following trivial but important fact

Proposition 3.7 Suppose X is path-connected. Then the following two statements are equivalent:

[L1] Every local minimum of $\breve{d}_x \circ w$ is a global minimum of $\breve{d}_x \circ w$ for all $x \in X$.

[L2] $B_i(Q) = B_i(N(q_i))$ for all $q_i \in Q$.

□

The proposition is evident from the definitions. Hence we have

Proposition 3.8 Suppose X is path-connected and $w(Q) \subset X$. Then (Q, N) with w preserves the topological order of X iff $B_i(Q) = B_i(N(q_i))$ for all $q_i \in Q$.

□

It is important to note that the Voronoi region of q_i is completely determined by X and w, and does not depend on the mapping N. On the other hand, the local Voronoi region of q_i is characterized by its 1-neighborhood $N(q_i)$. So conversely if we are formally given the local Voronoi region of q_i without knowing its 1-neighborhood $N(q_i)$, we can pick out a set of neurons from Q which can define this local Voronoi region, and treat this set of neurons as $N(q_i)$. Proposition 3.8 says that if (Q, N) with w preserves the topological order of X, we can identify the Voronoi regions with the local Voronoi regions. Conversely, if we are only given X and w, we can find a mapping N such that (Q, N) with w would preserve the topological order of X. This is done by first obtaining the Voronoi regions from X and w, and then obtaining the local Voronoi regions by identifying them with the Voronoi regions, and at last grasping a collection of neurons for each local Voronoi region to define the corresponding 1-neighborhood. This is the

basic idea of the induced mapping N_I. Of course many mapping N's satisfy Statement **[L2]**. For induced mapping we hope to find one which is almost minimal, in a sense which will be precisely defined in the next chapter.

Chapter 4

Induced graph

In this chapter we present the most natural but important theoretical concept in this thesis: the induced graph. We (Section 4.1) first introduce the Voronoi regions formally, and then present our last generalization of the definition of preserving the topological order, and at last define the induced graph. It follows by several remarks which are originally designed to fasten the concept of the induced graph, but eventually become an essential part of our theory. In particular, the results presented in Remark 4.3 and Remark 4.5 are significant, which also form the basis of our analysis (Section 4.2) in the practical method we use to find the induced graph. Remark 4.4, on the other hand, discusses the central idea when (Q, N) with w would be a good representation of X. At the end of this chapter (Section 4.3) we present some examples of the induced graph.

4.1 Formalism

Let X be the input space. Suppose each component[1] of X is path-connected and $w(Q) \subset X$. Since $\breve{d}(x, x')$ is defined only when x and x' fall in the same component of X, it would be convenient if, given any set of neurons O, we can grasp a subset of neurons O' of O such that $w(O')$ is contained in the same component of X. Let $C : Q \to \mathcal{P}(X)$ be defined by

$$C_i = \text{ component of } X \text{ s.t. } w(q_i) \in C_i$$

We always adopt the following notation: For any set $O \subset Q$,

$$\breve{O}_i = \{q_j \in O : w(q_j) \in C_i\}$$

Then it is clear that for any $q_k \in \breve{O}_i$, $C_k = C_i$. Define $B_i : \mathcal{P}(Q) \to \mathcal{P}(X)$ by

$$B_i(O) = \{x \in C_i : \breve{d}(w(q_i), x) \leq \breve{d}(w(q_j), x) \quad \forall q_j \in \breve{O}_i\}$$

It is evident from the definition that

$$B_i(O) = B_i(\breve{O}_i) \tag{4.1}$$

Moreover, we have the following basic results.

Proposition 4.1 B_i has the following properties:

[B1] $\forall O \subset Q$, $B_i(O)$ is closed in X.

[B2] $\forall O \subset Q$, $B_i(O \backslash \{q_i\}) = B_i(O) = B_i(O \cup \{q_i\})$.

[B3] $\forall O_1, O_2 \subset Q$, $B_i(O_1) \subset B_i(O_2)$ iff $O_1 \supset O_2$. In particular, $B_i(Q) \subset B_i(O)$ $\forall O \subset Q$.

[1] A maximal connected subspace of a topological space is called a component of the space. See [20, page 146].

[B4] $\forall O_1, O_2 \subset Q$, $B_i(O_2)$ strictly contains $B_i(O_1) \Rightarrow O_1$ strictly contains O_2.

[B5] Suppose $O_2 \subset O_1$ and $B_i(O_2) = B_i(O_1)$. Then $\forall O \subset Q$ s.t. $O_2 \subset O \subset O_1$, $B_i(O_2) = B_i(O) = B_i(O_1)$.

[B6] $\forall O \subset Q$,

$$\bigcup_{q_j \in O} B_j(O) = \bigcup_{q_j \in O} C_j$$

In particular,

$$\bigcup_{q_j \in Q} B_j(Q) = \bigcup_{q_j \in Q} C_j$$

[B7] $\forall O_1, O_2, \ldots, O_K \subset Q$,

$$\bigcup_{q_j \in Q} B_j(O_j) = \bigcup_{q_j \in Q} C_j$$

[B8] Suppose $x \in C_i = C_k$ s.t. $\breve{d}(w(q_i), x) = \breve{d}(w(q_k), x)$. Then $\forall O \subset Q$, $x \in B_i(O)$ iff $x \in B_k(O)$. In particular, $x \in B_i(Q)$ iff $x \in B_k(Q)$.

[B9] $\forall O \subset Q$, $B_i(O)$ is path-connected. Moreover, $\forall x \in B_i(O)$, if $\mathcal{G}$ is the trace of a geodesic in C_i joining $w(q_i)$ and x, then $\mathcal{G} \subset B_i(O)$.

Proof. **[B1]**, **[B2]**, **[B3]** are trivial from the definition. **[B4]** and **[B5]** can be easily obtained from **[B3]**. The proof of **[B6]** is evident. The proof of **[B7]** becomes easy if we apply the particular case $O = Q$ of **[B3]** and **[B6]**. So we start with **[B8]**.

Proof of [B8]. We note that $x \in B_i(O)$ means

$$\breve{d}(w(q_i), x) \leq \breve{d}(w(q_j), x) \quad \forall q_j \in \breve{O}_i \tag{4.2}$$

Putting $\breve{d}(w(q_i), x) = \breve{d}(w(q_k), x)$ into Equation 4.2, we have

$$\breve{d}(w(q_k), x) \leq \breve{d}(w(q_j), x) \quad \forall q_j \in \breve{O}_i = \breve{O}_k$$

which means $x \in B_k(O)$.

Proof of [B9]. Suppose $\exists x' \in \mathcal{G}$ s.t. $x' \notin B_i(O)$. Let

$$O' = \breve{O}_i \cup \{q_i\}$$

Then it is clear from Equation 4.1 and **[B2]** that $B_i(O) = B_i(O')$, and hence $x' \notin B_i(O')$. **[B6]** says that

$$\bigcup_{q_j \in O'} B_j(O') = C_i$$

Hence $x' \in B_k(O')$ for some $q_k \in O'$ and $q_k \neq q_i$. Since $q_i \in O'$, this implies $\breve{d}(w(q_k), x') \leq \breve{d}(w(q_i), x')$. But $\breve{d}(w(q_k), x') \neq \breve{d}(w(q_i), x')$, otherwise **[B8]** implies that $x' \in B_i(O')$. Hence we have

$$\breve{d}(w(q_k), x') < \breve{d}(w(q_i), x')$$

However, according to the definition of a geodesic,

$$\begin{aligned} \breve{d}(w(q_k), x) &\leq \breve{d}(w(q_k), x') + \breve{d}(x', x) \\ &< \breve{d}(w(q_i), x') + \breve{d}(x', x) = \breve{d}(w(q_i), x) \end{aligned}$$

contradicting to the fact that $x \in B_i(O)$.

□

$B_i(Q)$ is called the Voronoi region of q_i.

Definition 4.1 Suppose each component of X is path-connected and $w(Q) \subset X$. We say that (Q, N_G) with w preserves the topological order of X if $\bigcup_i C_i = X$ and $B_i(Q) = B_i(N_G(q_i))$ for all $q_i \in Q$.

□

According to Proposition 3.8, Definition 4.1 is really a generalization of Definition 3.5. Note that we have used the symbol N_G instead of N to emphasize that it is the *given mapping*, in contrast to the induced mapping N_I we shall define.

Next we proceed to the construction of the induced mapping N_I. Let $\mathcal{N}$ be the class of mappings $N : Q \to \mathcal{P}(Q)$ which satisfy

[N1] $\forall q_i \in Q, \ \ q_i \in N(q_i)$

[N2] $\forall q_i, q_j \in Q, \ \ q_i \in N(q_j)$ iff $q_j \in N(q_i)$

We define a partial order in $\mathcal{N}$. For $N_1, N_2 \in \mathcal{N}$, we say that $N_1 \leq N_2$ iff

[≤1] $\forall q_i \in Q, \ \ N_1(q_i) \subset N_2(q_i)$, and

[≤2] $\forall q_i \in Q, \ \ B_i(N_1(q_i)) = B_i(N_2(q_i))$.

It can be easily checked that $\leq$ satisfies the axioms of a partial order. Note that **[≤1]** is in fact saying that N_1 is a partial graph of N_2, and we shall continuing adopt the abbreviation $N_2 \supset N_1$. The reader may refer to Definition 3.4.

Proposition 4.2 Let $N_Q \in \mathcal{N}$ denotes the trivial mapping

$$N_Q(q_i) = Q \quad \forall q_i \in Q$$

and suppose $N_1 \leq N_Q$. Then for any $N \in \mathcal{N}$ s.t. $N \supset N_1$, $N_1 \leq N \leq N_Q$.

Proof. Since $N_1 \leq N_Q$,

$$B_i(N_1(q_i)) = B_i(N_Q(q_i)) \quad \forall q_i \in Q$$

Hence Proposition 4.2 follows from Proposition 4.1 [**B5**].

□

Recall that N_m is called a minimal element iff $N \leq N_m$ implies $N = N_m$ for any $N \in \mathcal{N}$.

Definition 4.2 Let $\{N_\alpha\}$ be the set of minimal elements which satisfy $N_\alpha \leq N_Q$. Then the induced mapping N_I is defined by

$$N_I(q_i) = \bigcup_\alpha N_\alpha(q_i) \quad \forall q_i \in Q$$

and we shall call the corresponding graph (Q, N_I) the induced graph.

□

It is clear that $N_I \in \mathcal{N}$. Moreover, we have

Proposition 4.3 Let N_I be the induced mapping and N be any mapping in $\mathcal{N}$ s.t. $N_\alpha \leq N$ for all α. Then

$$N_\alpha \leq N_I \leq N \leq N_Q \quad \forall \alpha \tag{4.3}$$

Proof. It is evident that $N_I \supset N_\alpha$ for all α. Also,

$$N(q_i) \supset N_\alpha(q_i) \quad \forall q_i \in Q \quad \forall \alpha$$

implies that

$$N(q_i) \supset \bigcup_\alpha N_\alpha(q_i) = N_I(q_i) \quad \forall q_i \in Q$$

i.e. $N \supset N_I$. Hence Inequality 4.3 follows from Proposition 4.2 immediately.

□

The reader should not be surprised to have

Proposition 4.4 Suppose each component of X is path-connected and $w(Q) \subset X$ s.t. $\bigcup_i C_i = X$. Then (Q, N_I) with w preserves the topological order of X.

□

Actually the above proposition holds for any $N \leq N_Q$. The special importance of N_I is owing to the fact that it is "minimal" in the sense of Proposition 4.3.

The reader may be a little bit disappointed in the definition of the induced mapping. The definition is in fact not complicated, but more complicated than we thought. Since Q is finite, $\mathcal{P}(Q)$ is finite, and hence $\mathcal{N}$ is finite. So theoretically there is no difficulty in finding the minimal elements. However, as we know from our experiences that we can in general easily identify N_I without any hesitation, it is understandable that the reader may not be satisfied with Definition 4.2 which cannot reveal the ease. Therefore we think that it is necessary to add some remarks to our definition.

Remark 4.1 Now we have a clear picture in mind what "preserving the topological order" means. It is characterized by the Voronoi regions. A Voronoi region is formally defined by all neurons, but in fact is constructed by the neurons in the 1-neighborhood only. In other words, many neurons are in fact useless in constructing the Voronoi region. Let $\hat{O}_i$ be Q minus the set of neurons which are useless in constructing the Voronoi region $B_i(Q)$. We want the 1-neighborhood $N_I(q_i)$ to merely contain $\hat{O}_i$. (Note that $N_I(q_i) \neq \hat{O}_i$ since q_i never contributes to the construction of $B_i(Q)$ by Proposition 4.1 **[B2]**, but we always want $q_i \in N_I(q_i)$.) It is natural to

determine $\hat{O}_i$ in the following way. First we initialize $\hat{O}_i = Q$. Then for $q_j = 1, 2, \ldots, K$, we examine whether $B_i(Q) = B_i(\hat{O}_i \backslash \{q_j\})$. If they are equal, we set $\hat{O}_i^+ = \hat{O}_i \backslash \{q_j\}$. After all neurons are presented *once*, the $\hat{O}_i$ obtained in this way is minimal, in the sense that

Proposition 4.5 If O is any subset of Q s.t. $O \subset \hat{O}_i$ and $B_i(O) = B_i(Q)$, then $O = \hat{O}_i$.

> **Proof.** Suppose $\exists q_j \in \hat{O}_i \backslash O$. Then by Proposition 4.1 **[B5]** $B_i(\hat{O}_i \backslash \{q_j\}) = B_i(Q)$. On the other hand, let $\hat{O}_i^-$ denote the $\hat{O}_i$ when q_j was examined. Since q_j was retained, $B_i(\hat{O}_i^- \backslash \{q_j\})$ strictly contains $B_i(Q)$. Then $B_i(\hat{O}_i^- \backslash \{q_j\})$ strictly contains $B_i(\hat{O}_i \backslash \{q_j\})$, and by Proposition 4.1 **[B4]** $\hat{O}_i^-$ is strictly contained in $\hat{O}_i$. This is impossible.
>
> □

Similarly we obtain $\hat{O}_i$ for all q_i. Then it seems more natural than Definition 4.2 to define the mapping N_I as $N_{\hat{O}}$ where

$$N_{\hat{O}}(q_i) = \hat{O}_i \cup \{q_i\} \quad \forall q_i \in Q$$

The above method is equivalent to defining a partial order in $\mathcal{P}(Q)$ instead of $\mathcal{N}$ and then finding out the minimal elements in order to define the 1-neighborhood. Unfortunately, $N_{\hat{O}}$ defined in this way *may not have symmetry*, i.e. satisfy Condition **[N2]**, since we try to define each 1-neighborhood individually. (See Example 4.1 below.) Therefore we must consider N_I as a whole, and Definition 4.2 becomes a suitable one. Of course there is a natural way to make $N_{\hat{O}}$ into a symmetrical mapping

$$N_{sym}(q_i) = \{q_j \in Q : q_j \in N_{\hat{O}}(q_i) \text{ or } q_i \in N_{\hat{O}}(q_j)\}$$

However, N_{sym} so defined may not be minimal in $(\mathcal{N}, \leq)$. (See Example 4.2 below.)

The reader may readily notice that the reasons of $N_{\hat{O}}$ being non-symmetrical are different in Example 4.1 and in Example 4.2. In Example 4.2 the sequential presentation of the neurons plays an important role, while in Example 4.1 it does not. The dependence on the sequence of the neurons can be eliminated by taking the union. (See discussions under Remark 4.2.) After that the mapping $N_{\hat{O}}$ is seldom non-symmetrical. If $N_{\hat{O}}$ is *still* non-symmetrical, as in Example 4.1, then it is reasonable to conjecture that there may be a stimulating and informative reason, i.e. $N_{\hat{O}}$ "should in some sense" really be non-symmetrical. So far we have assumed that N is symmetrical, mainly because intuitively we think "neighborhood" is a symmetrical relation. It is not so evident for symmetry to be a necessary condition. Studying non-symmetrical graphs, although interesting, is a very large topic and we shall continue to confine our studies to symmetrical graphs in this thesis.

□□□

Example 4.1 Let X be the region bounded by the solid line in Figure 4.1(a), $Q = \{q_1, q_2, q_3\}$ and $w(q_1) = (-\frac{1}{2}, \frac{1}{2})$, $w(q_2) = (\frac{1}{2}, \frac{1}{2})$, $w(q_3) = (-\frac{1}{2}, -\frac{1}{2})$. The Voronoi regions are separated by dashed lines in the figure. Then

$$\begin{aligned} N_{\hat{O}}(q_1) &= \{q_1, q_2, q_3\} \\ N_{\hat{O}}(q_2) &= \{q_1, q_2\} \\ N_{\hat{O}}(q_3) &= \{q_1, q_2, q_3\} \end{aligned}$$

Note that $N_{\hat{O}}$ is not symmetrical since $q_2 \in N_{\hat{O}}(q_3)$ but $q_3 \notin N_{\hat{O}}(q_2)$.

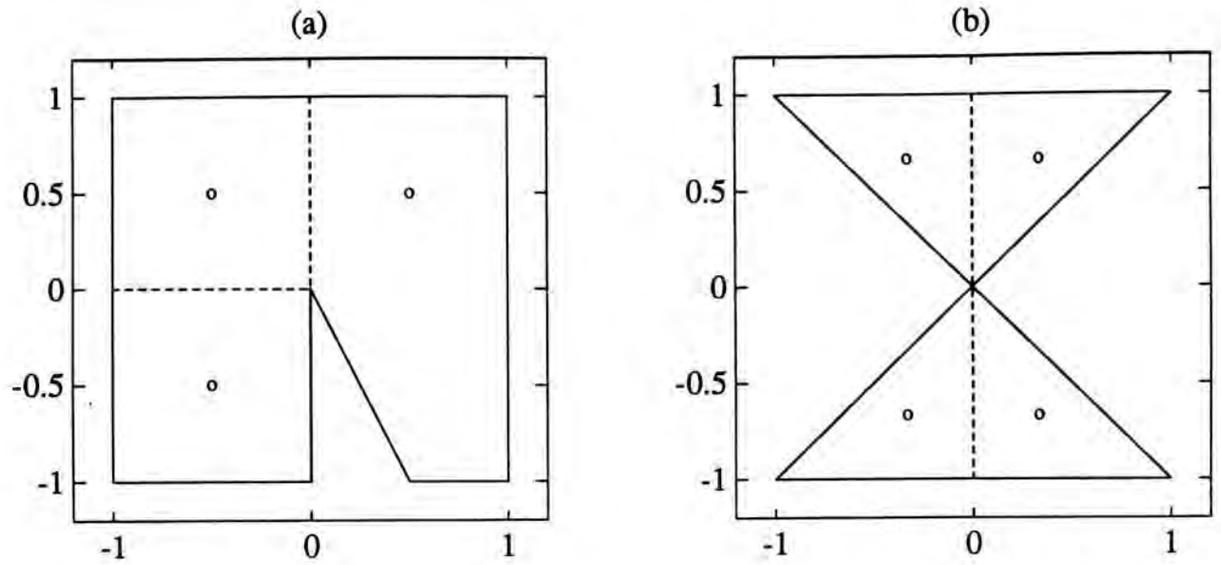

Figure 4.1: (a) The input space considered in Example 4.1. (b) The input space considered in Example 4.2.

□□□

Example 4.2 Let X be the region bounded by the solid line in Figure 4.1(b), $Q = \{q_1, q_2, q_3, q_4\}$ and $w(q_1) = (-\frac{1}{3}, \frac{2}{3})$, $w(q_2) = (\frac{1}{3}, \frac{2}{3})$, $w(q_3) = (-\frac{1}{3}, -\frac{2}{3})$, $w(q_4) = (\frac{1}{3}, -\frac{2}{3})$. The Voronoi regions are separated by dashed lines in the figure. Then

$$\begin{aligned} N_{\hat{O}}(q_1) &= \{q_1, q_2, q_4\} \\ N_{\hat{O}}(q_2) &= \{q_1, q_2, q_4\} \\ N_{\hat{O}}(q_3) &= \{q_2, q_3, q_4\} \\ N_{\hat{O}}(q_4) &= \{q_2, q_3, q_4\} \end{aligned}$$

Note that $N_{\hat{O}}$ is not symmetrical since, for example, $q_4 \in N_{\hat{O}}(q_1)$ but $q_1 \notin N_{\hat{O}}(q_4)$.

Suppose we make $N_{\hat{O}}$ into a symmetrical mapping N_{sym} by

$$N_{sym}(q_i) = \{q_j \in Q : q_j \in N_{\hat{O}}(q_i) \text{ or } q_i \in N_{\hat{O}}(q_j)\}$$

Then

$$\begin{aligned} N_{sym}(q_1) &= \{q_1, q_2, q_4\} \\ N_{sym}(q_2) &= \{q_1, q_2, q_3, q_4\} \\ N_{sym}(q_3) &= \{q_2, q_3, q_4\} \\ N_{sym}(q_4) &= \{q_1, q_2, q_3, q_4\} \end{aligned}$$

Clearly if we define N_1 s.t.

$$\begin{aligned} N_1(q_1) &= \{q_1, q_2, q_4\} \\ N_1(q_2) &= \{q_1, q_2, q_3\} \\ N_1(q_3) &= \{q_2, q_3, q_4\} \\ N_1(q_4) &= \{q_1, q_3, q_4\} \end{aligned}$$

then $N_1 \leq N_{sym}$ and hence N_{sym} is not a minimal element. In fact, N_1 itself is a minimal element.

However, if $w(q_1) = (-\frac{1}{3}, \frac{2}{3})$, $w(q_2) = (\frac{1}{3}, \frac{2}{3})$, $w(q_3) = (\frac{1}{3}, -\frac{2}{3})$, $w(q_4) = (-\frac{1}{3}, -\frac{2}{3})$, then

$$\begin{aligned} N'_{\hat{O}}(q_1) &= \{q_1, q_2, q_3\} \\ N'_{\hat{O}}(q_2) &= \{q_1, q_2, q_3\} \\ N'_{\hat{O}}(q_3) &= \{q_2, q_3, q_4\} \\ N'_{\hat{O}}(q_4) &= \{q_2, q_3, q_4\} \end{aligned}$$

(We have added a prime to distinguish it from before.) Clearly $N'_{\hat{O}}$ is also not symmetrical since, for example, $q_3 \in N_{\hat{O}}(q_1)$ but $q_1 \notin N_{\hat{O}}(q_3)$.

If similarly we make $N'_{\hat{O}}$ into a symmetrical mapping N'_{sym}, we would have

$$N'_{sym}(q_1) = \{q_1, q_2, q_3\}$$

$$
\begin{aligned}
N'_{sym}(q_2) &= \{q_1, q_2, q_3, q_4\} \\
N'_{sym}(q_3) &= \{q_1, q_2, q_3, q_4\} \\
N'_{sym}(q_4) &= \{q_2, q_3, q_4\}
\end{aligned}
$$

Once again if we define N_2 s.t.

$$
\begin{aligned}
N_2(q_1) &= \{q_1, q_2, q_3\} \\
N_2(q_2) &= \{q_1, q_2, q_4\} \\
N_2(q_3) &= \{q_1, q_3, q_4\} \\
N_2(q_4) &= \{q_2, q_3, q_4\}
\end{aligned}
$$

then $N_2 \leq N'_{sym}$ and hence N'_{sym} is not a minimal element. In fact, N_2 itself is another minimal element.

Hence we see that N_α may not be unique.

□□□

Remark 4.2 The reader may notice that we define N_I to be the union of N_α. It is because the N_α may not be unique (although it often is). Therefore, if we want uniqueness, we cannot define N_I to be any N_α. Imagine we obtain a procedure similar to the previous remark for the mapping N considered as a whole, in which one link (i.e. pair of neurons) is examined and may be discarded at a time. Although this procedure guarantees that we can obtain a minimal element, owing to the non-uniqueness, the minimal element we obtain may depend on the order of the links presented. Therefore, if we define N_I to be this minimal element, we may obtain another N_I if we renumber the neurons. This situation is in general undesirable. To eliminate its dependence on the sequence of the links presented, we simply take the union, and hence we obtain a unique

N_I which is still "as minimal as possible" in the sense of Proposition 4.3. (Of course we cannot take the intersection otherwise we cannot guarantee $N_I \leq N_Q$.)

□□□

Remark 4.3 Another reason of saying that the reader may not be satisfied with Definition 4.2 is the following. Suppose we are given $q_i, q_j \in Q$. We cannot tell immediately from the definition whether $q_j \in N_I(q_i)$. It is because the condition of telling whether $q_j \in N_I(q_i)$ is not so simple, and it is not good to use it as the definition. However, simple condition really exists for non-minimal mapping. The following proposition is an example, which is in fact saying that if two Voronoi regions have empty intersection, the corresponding neurons do not contribute to building the Voronoi region of each other.

Proposition 4.6 Let $\tilde{O}_i = \{q_j \in Q : B_i(Q) \cap B_j(Q) \neq \emptyset\}$. Suppose O is any subset of Q s.t. $B_i(O) = B_i(Q)$. Then $B_i(O \cap \tilde{O}_i) = B_i(Q)$.

Proof. We choose an indirect but interesting proof. It suffices to show that $B_i(O \cap \tilde{O}_i) \neq B_i(O)$ will result in contradiction. Suppose $B_i(O \cap \tilde{O}_i) \supset B_i(O)$ but $B_i(O \cap \tilde{O}_i) \neq B_i(O)$. Consider $B_i(O \cap \tilde{O}_i)$ itself as a topological space[2]. Then $B_i(O)$ is closed in $B_i(O \cap \tilde{O}_i)$ since $B_i(O)$ is closed in X by Proposition 4.1 **[B1]**. Therefore $B_i(O)$ cannot be open in $B_i(O \cap \tilde{O}_i)$. (Otherwise $B_i(O)$ both open and closed in $B_i(O \cap \tilde{O}_i)$ implies that $B_i(O)$ is a component of $B_i(O \cap \tilde{O}_i)$, which implies

[2]The topology of $B_i(O \cap \tilde{O}_i)$ is the relative topology on $B_i(O \cap \tilde{O}_i)$ which is induced from the topology of X.

$B_i(O) = B_i(O \cap \tilde{O}_i)$ since $B_i(O \cap \tilde{O}_i)$ is connected by Proposition 4.1 **[B9]**.) That means $\exists x \in B_i(O)$ s.t. all neighborhoods[3] of x cannot be contained in $B_i(O)$. Then it is impossible to have

$$\breve{d}(w(q_i), x) < \breve{d}(w(q_j), x) \quad \forall q_j \in \breve{O}_i \backslash \tilde{O}_i$$

(Otherwise, $\exists$ a neighborhood U of x s.t. for all $x' \in U$,

$$\breve{d}(w(q_i), x') < \breve{d}(w(q_j), x') \quad \forall q_j \in \breve{O}_i \backslash \tilde{O}_i$$

Then U is a neighborhood of x which is contained in $B_i(O)$.) Therefore we may take $q_k \in \breve{O}_i \backslash \tilde{O}_i$ s.t.

$$\breve{d}(w(q_i), x) = \breve{d}(w(q_k), x)$$

Since $x \in B_i(O) = B_i(Q)$, then by Proposition 4.1 **[B8]** $x \in B_k(Q)$. Hence $B_i(Q) \cap B_k(Q) \neq \emptyset$, contradicting to the fact that $q_k \notin \tilde{O}_i$.

□

A trivial consequence is

Corollary 4.7 $B_i(\tilde{O}_i) = B_i(Q)$.

□

According to Corollary 4.7 we may naturally define a mapping $N_{\tilde{O}}$ by

$$N_{\tilde{O}}(q_i) = \tilde{O}_i \quad \forall q_i \in Q$$

It is clear that $N_{\tilde{O}}$ satisfies Condition **[N1]** and **[N2]** and hence in $\mathcal{N}$. Moreover, $N_\alpha \leq N_{\tilde{O}}$ for all α and it follows from Proposition 4.3 that

[3]Here we consider $B_i(O \cap \tilde{O}_i)$ itself as a topological space, and a neighborhood U of x is a subset U of $B_i(O \cap \tilde{O}_i)$ s.t. U is open in $B_i(O \cap \tilde{O}_i)$ and $x \in U$.

$N_I \leq N_{\tilde{O}}$. (We skip the proof here, although it is easy, since it is a special case of Proposition 4.10 below, of which we shall give a proof.) In fact it may happen that $N_I \neq N_{\tilde{O}}$, and therefore we cannot use $N_{\tilde{O}}$ to replace N_I in the theoretical work. However, the usefulness of Corollary 4.7 should not be underestimated, especially we know that it is very effective in telling when two neurons should not be in the 1-neighborhood of each other.

It is not hard to see that Proposition 4.6 is equivalent to saying that if $q_j \notin \tilde{O}_i$, then for any $O \subset Q$ s.t. $B_i(O) = B_i(Q)$, $B_i(O \backslash \{q_j\}) = B_i(Q)$. This motivates us to give a precise definition of what we mean by a neuron q_j contributing to the construction of the Voronoi region $B_i(Q)$. To distinguish it from the term "contribution" we used before without precisely defined, we give it a special name.

Definition 4.3 We say that the neuron q_j does not weakly contribute to $B_i(Q)$ iff for any $O \subset Q$ s.t. $B_i(O) = B_i(Q)$, $B_i(O \backslash \{q_j\}) = B_i(Q)$. Otherwise we say that q_j weakly contributes to $B_i(Q)$.

□

An important result is Proposition 4.9. We start with a lemma.

Lemma 4.8 Let $q_j \neq q_i$ s.t. q_j does not weakly contribute to $B_i(Q)$ and q_i does not weakly contribute to $B_j(Q)$. Then $q_j \notin N_\alpha(q_i)$ and $q_i \notin N_\alpha(q_j)$ for any N_α.

Proof. q_j does not weakly contribute to $B_i(Q)$ and q_i does not weakly contribute to $B_j(Q)$ imply that

$$\begin{aligned} B_i(N_\alpha(q_i)) &= B_i(N_\alpha(q_i) \backslash \{q_j\}) \\ B_j(N_\alpha(q_j)) &= B_j(N_\alpha(q_j) \backslash \{q_i\}) \end{aligned}$$

Then we may define a mapping $N : Q \to \mathcal{P}(Q)$ by

$$\begin{aligned} N(q_i) &= N_\alpha(q_i)\backslash\{q_j\} \\ N(q_j) &= N_\alpha(q_j)\backslash\{q_i\} \\ N(q_k) &= N_\alpha(q_k) \quad \forall q_k \in Q\backslash\{q_i, q_j\} \end{aligned}$$

Clearly $N \in \mathcal{N}$. Since N_α is minimal, we have

$$\begin{aligned} N_\alpha(q_i) &= N_\alpha(q_i)\backslash\{q_j\} \\ N_\alpha(q_j) &= N_\alpha(q_j)\backslash\{q_i\} \end{aligned}$$

and hence $q_j \notin N_\alpha(q_i)$ and $q_i \notin N_\alpha(q_j)$.

□

Proposition 4.9 Suppose $q_j \in N_I(q_i)$ but $q_j \neq q_i$. Then q_j weakly contributes to $B_i(Q)$ or q_i weakly contributes to $B_j(Q)$.

Proof. Suppose both are not true. Then by Lemma 4.8 $q_j \notin N_\alpha(q_i)$ for all α, contradicting to the fact that $q_j \in N_I(q_i)$.

□

Proposition 4.9 in fact says that N_I discards all neurons which do not weakly contribute, except those retained for the reflexive (Condition [**N1**]) or symmetrical (Condition [**N2**]) reason.

Another important result is

Proposition 4.10 Let $N \in \mathcal{N}$. Suppose $q_j \notin N(q_i)$ implies that q_j does not weakly contribute to $B_i(Q)$. Then $N_I \leq N$.

Proof. First we show that $N_\alpha \leq N$ for all α. Suppose $q_j \notin N(q_i)$. Then q_j does not weakly contribute to $B_i(Q)$. On the

other hand, $q_i \notin N(q_j)$ since N satisfies Condition **[N2]**. So q_i does not weakly contribute to $B_j(Q)$. It is also clear that $q_j \neq q_i$ since N satisfies Condition **[N1]**. Applying Lemma 4.8 we know that $q_j \notin N_\alpha(q_i)$. Hence $N(q_i) \supset N_\alpha(q_i)$. Since this is true for all q_i, $N \supset N_\alpha$ and by Proposition 4.2, $N_\alpha \leq N$. Then it follows from Proposition 4.3 that $N_I \leq N$.

□

Proposition 4.10, of course, once again tells us that N_I is in some sense "minimal". However, the main importance of Proposition 4.10 is that *it guarantees how we can obtain a mapping N which is well-defined, i.e. independent of the numbering of the neurons.* This is simply achieved by employing "not weakly contribute" to be a necessary condition to discarding a neuron. It is in fact natural since "not weakly contribute" means that the neuron can be discarded from a set independent of the set holding it. In particular, if the 1-neighborhood is obtained by examining and possibly discarding one neuron at a time, then "not weakly contribute" says that the neuron can be discarded at any time, and hence independent of the specific sequence. In other words, all "not weakly contributing" neurons can in fact be discarded simultaneously.

The reader may readily understand why we call the kind of contribution we have defined in Definition 4.3 the "weak" contribution. Since N_I is possibly non-minimal in $(\mathcal{N}, \leq)$, Proposition 4.10 implies that some minimal elements N_α may discard some neurons which *do* weakly contribute, but the same Voronoi regions still result. So we call the contribution "weak".

□□□

Remark 4.4 In this remark we want to discuss when (Q, N) with w is a good representation of X. Of course we have not precisely defined what we mean by a good representation of X, but we can still discuss it in the intuitive sense. First it is natural to ask if

[R1] (Q, N) with w preserves the topological order of X

whether it is always a good representation of X. The answer is no. For example, if $N = N_Q$, then **[R1]** always holds provided that $w(Q) \subset X$ and $\bigcup_i C_i = X$, but (Q, N_Q) with w is in general not a good representation of X. That means, in addition to **[R1]** we need a suitable choice of N. An obvious choice is the induced mapping, i.e.

[R2] $N = N_I$

However, **[R1]** together with **[R2]** still cannot guarantee a good representation of X. (See Example 4.3 below.) In fact, there exists an induced mapping N_I for arbitrary w which satisfies $w(Q) \subset X$. Intuitively a suitable choice of w is necessary for a good representation of X. It is because w determines how the neurons are placed in X and hence the Voronoi regions. To be a good representation, we hope that

[R3] w places the neurons in X approximately according to the probability distribution defined on it

[R3] is not very rigorous. Roughly speaking, we want the "firing probability" of each neuron is approximately the same, and each Voronoi region is approximately isotropic, i.e. the number of training patterns in each direction is approximately the same. Such a w can in general be obtained using competitive learning. (For a more rigorous treatment of **[R3]** see [6, 7], in which Grabec studied intensively how to place the neurons in the

input space in order to reflect the probability distribution defined on it. We may employ the idea of Grabec in our case. Here we explain it briefly. Let $\Omega(x, x') : X \times X \to \mathbf{R}$ be a "window" function. In general $\Omega(x, x')$ is a decreasing function of $\breve{d}(x, x')$. Grabec used the Gaussian function for some simple input spaces. Then w is chosen in order to minimize

$$\left\| \sum_{i=1}^{K} \Omega(x, w(q_i)) - f(x) \right\|$$

where $f(x)$ denotes the probability distribution defined on X.)

If **[R1]**, **[R2]** and **[R3]** are all satisfied, we may say that (Q, N) with w is a good representation of X.

□□□

Example 4.3 Suppose $X = [-1, 1]^2$ and

$$w(Q) = \left\{ (-\frac{7}{8}, 0), (-\frac{5}{8}, 0), (-\frac{3}{8}, 0), (-\frac{1}{8}, 0), (\frac{1}{8}, 0), (\frac{3}{8}, 0), (\frac{5}{8}, 0), (\frac{7}{8}, 0) \right\}$$

Then $w(Q, N_I)$ is as shown in Figure 4.2. Clearly **[R1]** and **[R2]** are satisfied. Moreover, the area of each Voronoi region is the same, i.e. the firing probability of each neuron is the same. However, (Q, N_I) with this w is of course not a good representation of X. We may notice that the Voronoi regions are not isotropic.

□□□

In the next section we introduce the method we practically use to find the induced mapping.

4.2 Practical way to find the induced graph

Before we introduce the practical method, we first give a remark.

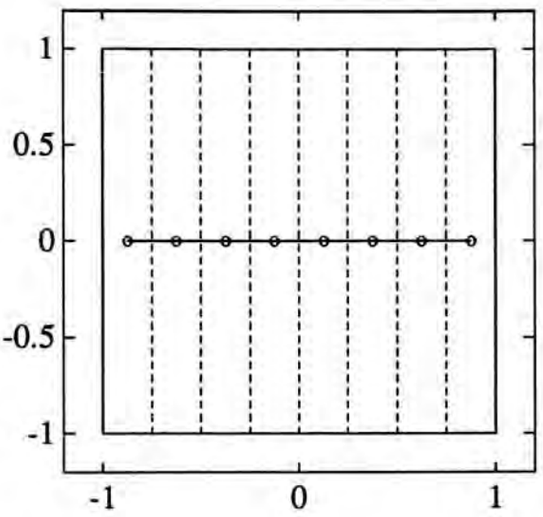

Figure 4.2: $w(Q, N_I)$ in Example 4.3. X is the region bounded by the solid square and the Voronoi regions are separated by dashed lines.

Remark 4.5 For practical reason it is natural to ask what happens if we use direct distance instead of geodesic distance. As mentioned before, if we use direct distance, we must assume X to be convex in order to guarantee the consistence of the theory. So it suffices to study the role of convexity, which cannot be easily seen at a glance, when direct distance is used. Define

$$\tilde{B}_i(O) = \{x \in X : d(w(q_i), x) \leq d(w(q_j), x) \quad \forall q_j \in O\}$$

If we say that $B_i(Q)$ is the Voronoi region of q_i in X, then $\tilde{B}_i(Q)$ is the intersection of X and the Voronoi region of q_i in $\mathbf{R}^n$. If X is convex, then $B_i(O) = \tilde{B}_i(O)$ for any $O \subset Q$, and Proposition 4.1 [**B9**] still holds, i.e. $\tilde{B}_i(O)$ is path-connected. In fact we know that $\tilde{B}_i(O)$ is moreover convex. So everything is fine. However, if X is not convex, then $\tilde{B}_i(O)$

may not be connected. (See Example 4.4 below.) The proof of Proposition 4.6, although simple, in fact needs $B_i(O)$ to be connected. If we do not have Proposition 4.6, a direct consequence is the disappearance of Corollary 4.7, the most powerful tools of telling when two neurons should not be in the 1-neighborhood of each other. The importance of Proposition 4.6 is much beyond this. It is the corner-stone of our Definition 4.3. Then all results in terms of "weakly contribute", including Lemma 4.8, Proposition 4.9 and Proposition 4.10 will disappear. In other words, *almost all theoretical tools which help to find out the induced mapping disappear simultaneously.* From this we see that guaranteeing connected is very informative to us in studying the induced mapping. Moreover, even if we can find out an induced mapping, we may not obtain a good representation of X if $\tilde{B}_i(O)$ is not really connected. (See Example 4.4 below. Also see discussions in Section 3.5.) It is intuitively evident if we picture in our mind that each neuron represents a piece of input space and the links between neurons tell how such pieces are pasted together. If a Voronoi region is not connected, that means the corresponding neuron in fact representing several pieces of input space, which is extremely unnatural.

The reader may readily see that the main reason of using geodesic distance is to guarantee that $B_i(O)$ is path-connected. Fortunately, even if we use direct distance and X is not convex, generally $\tilde{B}_i(O)$ is path-connected provided that the cardinality of O is sufficiently large. (See Example 4.4.) So practically there is almost no difficulty in obtaining path-connected Voronoi regions if we use more neurons to represent the input space X.

□□□

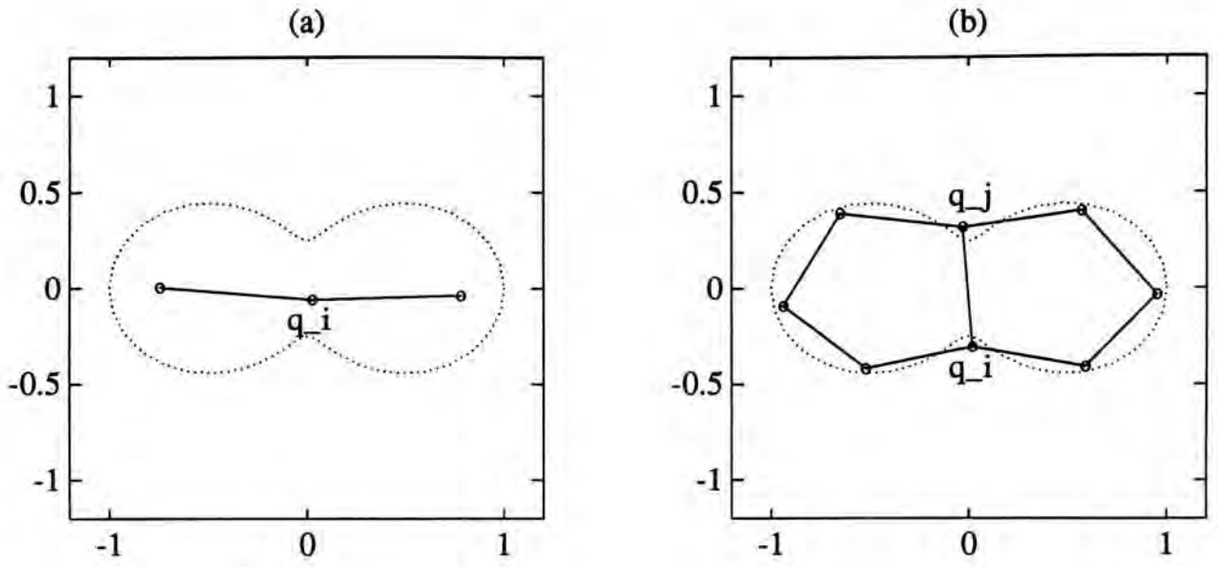

Figure 4.3: (a) $w(Q, \tilde{N}_I)$ for the case $K = 3$ in Example 4.4. (b) $w(Q, \tilde{N}_I)$ for the case $K = 8$ in Example 4.4.

Example 4.4 The configuration is almost the same as that of Example 3.12. $X = \mathbf{P}(1, 3)$; the total number of training patterns $M = 10000$; $(Q, N) = \mathbf{S}^1_K$; w is obtained using Kohonen's algorithm; the radius of neighborhood

$$r = \begin{cases} \lfloor \frac{1000-l}{1000-1} \times \frac{K}{4} \rfloor & \text{if} \quad l \leq 1000 \\ 0 & \text{if} \quad l > 1000 \end{cases}$$

and the learning rate

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

The result when $K = 3$ is plotted in Figure 4.3(a), in which a small circle represents a neuron and a straight line represents a link in $(Q, \tilde{N}_I)$, where $\tilde{N}_I$ is the practical approximation of N_I which is obtained using Practice 2 mentioned below. (For convenience we denote this graphical

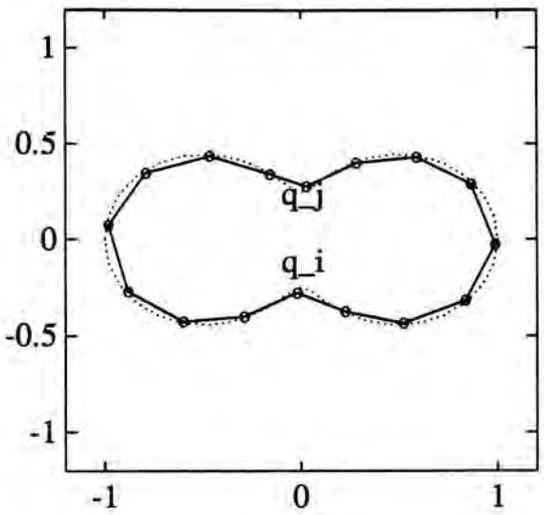

Figure 4.4: $w(Q, \tilde{N}_I)$ for the case $K = 16$ in Example 4.4.

representation as $w(Q, \tilde{N}_I)$.) Clearly $\tilde{B}_i(Q)$ is not connected. Suppose we increase K to 8. The result $w(Q, \tilde{N}_I)$ is plotted in Figure 4.3(b). Then we see that $\tilde{B}_i(Q)$ is now path-connected, but $\tilde{N}_I$ is still incorrect since $\tilde{B}_i(Q \backslash \{q_j\})$ is still not connected. At last we increase K to 16. The result $w(Q, \tilde{N}_I)$ is plotted in Figure 4.4. Then $\tilde{B}_i(Q \backslash \{q_j\})$ is eventually path-connected and hence a correct $\tilde{N}_I$ is obtained. Of course $\tilde{B}_i(O)$ is still not connected for some $O \subset Q$, but as we mentioned below Practice 2 is powerful enough to solve the problem.

□□□

Next we introduce practically how we obtain an approximation of the induced mapping.

Practice 2 To obtain an approximation $\tilde{N}_I$ of the induced mapping N_I, we first initialize $\tilde{\mathbf{N}}_I$, the matrix representation of $\tilde{N}_I$, to be the identity matrix. Then the training patterns are presented sequentially. Let $\tilde{x}_l$ be the current

training pattern. Suppose q_i is the global minimum of $d_{\tilde{x}_l} \circ w$ and q_j is the global minimum of $d_{\tilde{x}_l} \circ w \mid_{Q\setminus\{q_i\}}$. Then we set $(\tilde{\mathbf{N}}_I)_{ji} = (\tilde{\mathbf{N}}_I)_{ij} = 1$.

Or equivalently, for any $q_j \in Q$, $q_j \in \tilde{N}_I(q_i)$ iff

[$\tilde{\mathbf{N}}$1] $q_j = q_i$, or

[$\tilde{\mathbf{N}}$2] $\exists \tilde{x}_l$ s.t. q_i is the global minimum of $d_{\tilde{x}_l} \circ w \mid_{Q\setminus\{q_j\}}$ and q_j is the global minimum of $d_{\tilde{x}_l} \circ w \mid_{Q\setminus\{q_i\}}$.

Clearly, $\tilde{N}_I$ so obtained is in $\mathcal{N}$ and independent of the numbering of the neurons and the order of the training patterns presented.

□

Thinking with some examples the reader may find that in most cases $\tilde{N}_I = N_I$. However, it is not easy to feel the theoretical correspondence between $\tilde{N}_I$ and N_I. We try to explain this by reversing the process. That is, we translate Practice 2 into its theoretical version by replacing $\exists \tilde{x}_l$ by $\exists x \in X$, d by $\breve{d}$ and $\tilde{B}_i$ by B_i. Hence we obtain a theoretical version $\hat{N}_I$ of $\tilde{N}_I$ and then compare it with the induced mapping N_I.

The reader may notice that we have formulated Practice 2 in terms of global minimum. Actually in practice it is more natural to find out the (global or local) minimum directly, but theoretically it becomes more convenient to formulate in terms of Voronoi regions. So we rewrite Practice 2 in terms of Voronoi regions. For any $q_j \in Q$, $q_j \in \tilde{N}_I(q_i)$ iff

[$\tilde{\mathbf{N}}$1] $q_j = q_i$, or

[$\tilde{\mathbf{N}}$2] $\exists \tilde{x}_l$ s.t. $\tilde{x}_l \in \tilde{B}_i(Q) \cap \tilde{B}_j(Q\setminus\{q_i\})$ or $\tilde{x}_l \in \tilde{B}_j(Q) \cap \tilde{B}_i(Q\setminus\{q_j\})$.

However, we should not forget Assumption 2, which points out that

$$\tilde{x}_l \in \tilde{B}_i(Q) \Rightarrow \tilde{x}_l \notin \tilde{B}_j(Q)$$

since $q_j \neq q_i$. So we should rewrite **[$\tilde{\mathbf{N}}$2]** as

[$\tilde{\mathbf{N}}$2] $\exists \tilde{x}_l$ s.t. $\tilde{x}_l \in [\tilde{B}_i(Q) \backslash \tilde{B}_j(Q)] \cap \tilde{B}_j(Q \backslash \{q_i\})$ or $\tilde{x}_l \in [\tilde{B}_j(Q) \backslash \tilde{B}_i(Q)] \cap \tilde{B}_i(Q \backslash \{q_j\})$.

Now we may translate **[$\tilde{\mathbf{N}}$1]** and **[$\tilde{\mathbf{N}}$2]** into their theoretical versions. Let $\hat{N}_I$ denote the theoretical version of $\tilde{N}_I$. Then for any $q_j \in Q$, $q_j \in \hat{N}_I(q_i)$ iff

[$\hat{\mathbf{N}}$1] $q_j = q_i$, or

[$\hat{\mathbf{N}}$2] $B_j(Q)$ is strictly contained in $B_j(Q \backslash \{q_i\})$ or $B_i(Q)$ is strictly contained in $B_i(Q \backslash \{q_j\})$.

It is clear that the requirement $B_i(Q \backslash \{q_j\}) = B_i(Q)$ is weaker than that of "not weakly contribute", which requires $B_i(O \backslash \{q_j\}) = B_i(O)$ for any $O \subset Q$. Therefore $\hat{N}_I(q_i)$ may discard some neurons which *do* weakly contribute, and as a result we cannot guarantee from Proposition 4.10 that $N_I \leq \hat{N}_I$. In fact, it may even happen that $\hat{N}_I \not\leq N_Q$, i.e. $B_i(\hat{N}_I(q_i)) \neq B_i(Q)$ for some $q_i \in Q$. Therefore, the reader may be disappointed. Surprisingly, $\tilde{N}_I$ often produces results much better than we expect. It is because practically we must use direct distance instead of geodesic distance. As mentioned before, there is a problem of non-connectedness of $\tilde{B}_i(O)$. Practically we can increase the number of neurons, i.e. the cardinality of Q, to make $\tilde{B}_i(Q)$ path-connected. Therefore, it is not difficult to have $\tilde{B}_i(Q)$ and $\tilde{B}_i(Q \backslash \{q_j\})$ path-connected. However, *we cannot make $\tilde{B}_i(O)$ path-connected for arbitrary O.* In particular, increasing the cardinality of Q has little effect in increasing the cardinality of the 1-neighborhoods.

(It is natural since the 1-neighborhoods represent the "structure" of the input space which should not be affected by the number of the neurons representing it.) If we consider weak contribution, i.e. $\tilde{B}_i(O)$ and $\tilde{B}_i(O\backslash\{q_j\})$ for arbitrary O, then using direct distance may eventually produce incorrect information as the cardinality of O becomes smaller and smaller, and as a result a "not weakly contributing" neuron may be wrongly classified as a "weakly contributing" neuron. However, since we now consider $\tilde{B}_i(Q)$ and $\tilde{B}_i(Q\backslash\{q_j\})$, such mistakes are seldom made. In other words, $\tilde{N}_I$ is very robust in the choice between geodesic distance and direct distance. If we say that $\tilde{N}_I$ discards some neurons which *do* weakly contribute, it also discards some neurons which *do not* weakly contribute but would otherwise be wrongly classified as weakly contributing neurons when direct distance is used, and its positive effect is in general much greater than its negative effect. Therefore it is a good choice to use $\tilde{N}_I$ as a practical approximation of N_I in order to avoid errors caused by using direct distance.

4.3 Some examples

At last we give some examples of $\tilde{N}_I$. We have already seen some in Example 3.12, 3.13, 3.14 and 4.4. Here we give some more. Consider the examples we discussed in Chapter 2. $w(Q, \tilde{N}_I)$ for w obtained in Example 2.1, 2.2, 2.3, 2.4 and 2.5 are shown in Figure 4.5, 4.6(a), 4.6(b), 4.7(a) and 4.7(b) respectively. In these examples w were obtained using Kohonen's algorithm, but in fact the induced graph has no relation to how w is obtained. For any w there is an induced graph. To obtain a good representation of X, we would more often use competitive learning instead of Kohonen's algorithm to obtain w. However,

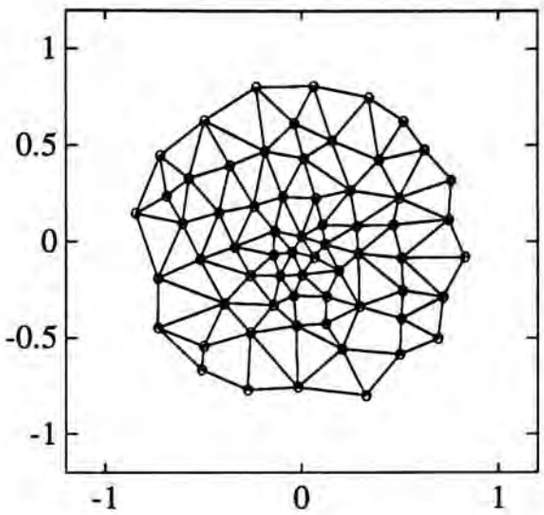

Figure 4.5: $w(Q, \tilde{N}_I)$ in Example 2.1.

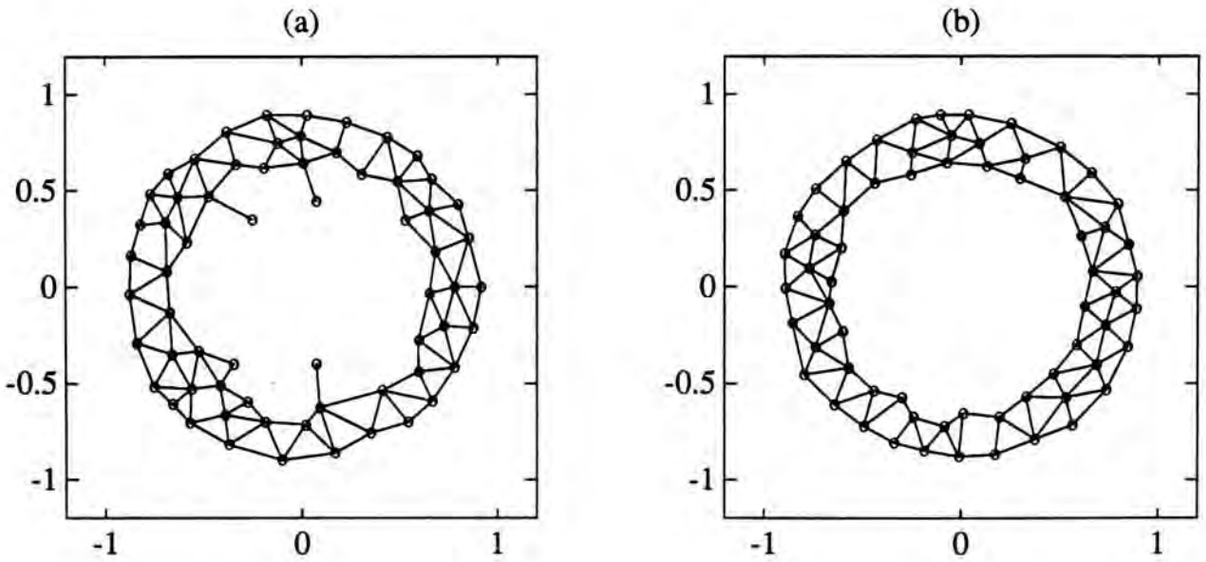

Figure 4.6: (a) $w(Q, \tilde{N}_I)$ in Example 2.2. (b) $w(Q, \tilde{N}_I)$ in Example 2.3.

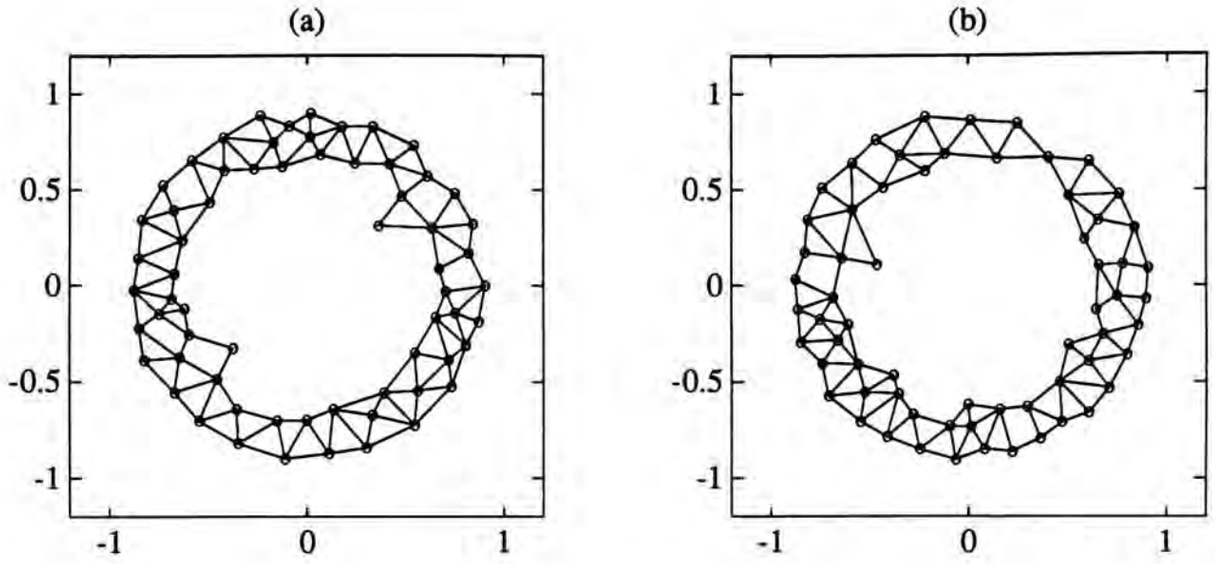

Figure 4.7: (a) $w(Q, \tilde{N}_I)$ in Example 2.4. (b) $w(Q, \tilde{N}_I)$ in Example 2.5.

under special circumstances competitive learning may not work, as we see from the following example.

Example 4.5 Here we want to give an example which is similar to the one given in [9, pages 96–97]. The input space

$$X = \left[\left[-\frac{1}{3}, \frac{1}{3}\right] \times \left[-1, -\frac{1}{3}\right] \right] \cup \left[\left[-\frac{1}{3}, \frac{1}{3}\right] \times \left[\frac{1}{3}, 1\right] \right]$$

consists of two components. The training patterns are first drawn from the lower component of X only, and as a result, all the neurons are placed in the lower component of X. Then the training patterns are drawn from both components of X, and we see how the neurons drift from one component to both. To achieve this, we cannot simply use competitive learning, otherwise only one neuron will be attracted to the upper component. (This is similar to the well-known "dead units" problem for simple competitive learning. See [8, page 221].) One way to solve this problem is to

establish "links" between neurons. Hence if one neuron is attracted by the upper component of X, it will pull other neurons to go. It just means to provide a graph (Q, N) for Kohonen's algorithm to use. In [9], two examples were given. In the first one (Q, N) was specified initially. In the second one (Q, N) was established dynamically based on the minimal spanning tree. Here, if we are not given any graph initially, we can simply establish (Q, N) dynamically based on the induced mapping. Similar to the minimal spanning tree in [9], we need not calculate the induced graph frequently. In fact, we think that we *should not* update (Q, N) to $(Q, \tilde{N}_I)$ so frequently. It is because the induced graph, unlike the minimal spanning tree, is not necessarily connected[4]. It just reflects the structure of the input space. The disconnectedness of the input space will make the neurons in the upper component of X disconnected from the neurons in the lower component of X. If (Q, N) is updated to $(Q, \tilde{N}_I)$ so frequently, the links may disappear before sufficient neurons have drifted to the upper component of X. (Of course this problem can also be solved by many other simple methods.)

In this example the total number of training patterns $M = 50000$. The first 10000 training patterns are drawn from the lower component of X only, and the following 40000 training patterns are drawn from both components of X. There are totally 32 neurons. The initial weights are simply set to the first 32 training patterns. Hence they are all in the lower component of X. w is then trained using Kohonen's algorithm,

[4]For a *symmetrical* graph to be connected we mean that for any two distinct neurons $q_i, q_j \in Q$, there exists a path (of any finite length) from q_i to q_j.

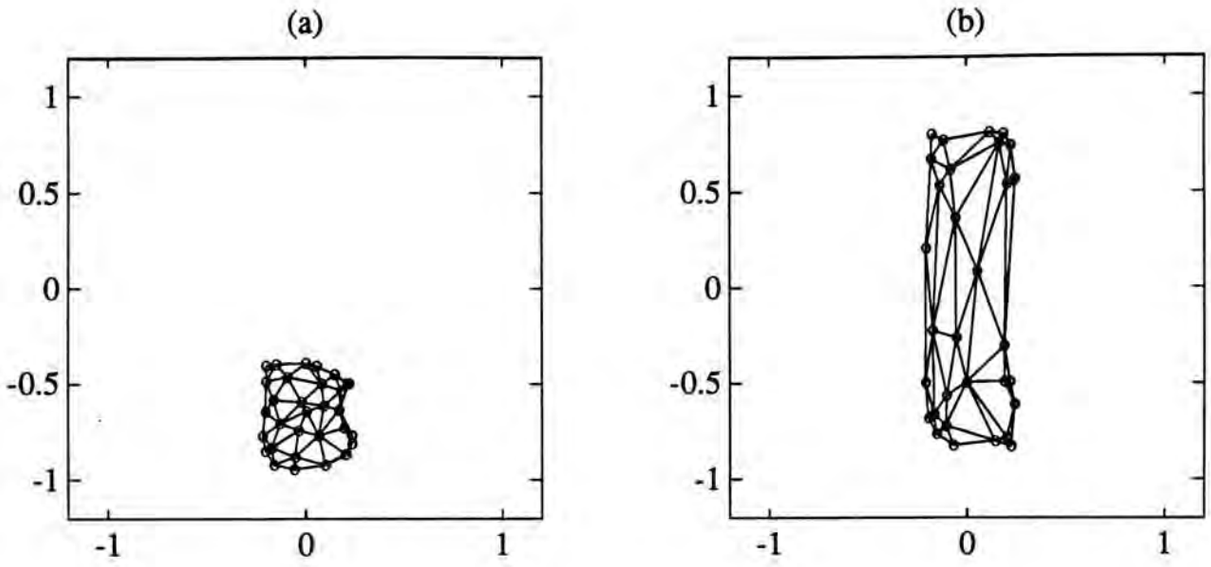

Figure 4.8: (a) $w(Q, N)$ in Example 4.5 at $l = 10000$ after updated. (b) $w(Q, N)$ in Example 4.5 at $l = 15000$ before updated.

whose parameters are set as follows:

$$r = \begin{cases} \lfloor \frac{40000-l}{40000-1} \times 2 \rfloor & \text{if} \quad l \leq 40000 \\ 0 & \text{if} \quad l > 40000 \end{cases}$$

$$\eta = \begin{cases} 0.15 & \text{if} \quad r > 0 \\ 0.015 & \text{if} \quad r = 0 \end{cases}$$

Note that the graph (Q, N) used by Kohonen's algorithm is now the last calculated induced graph instead of any given graph. The induced graph is calculated at the beginning and just after

$$l = 5000, 10000, 15000, \ldots, 50000$$

Figures 4.8(a)–4.9(b) show $w(Q, N)$ at $l = 10000$ just after updated to the induced graph, $w(Q, N)$ at $l = 15000$ just before updated to the induced graph, $w(Q, N)$ at $l = 15000$ just after updated to the induced graph and $w(Q, N)$ at $l = 20000$ just after updated to the induced graph.

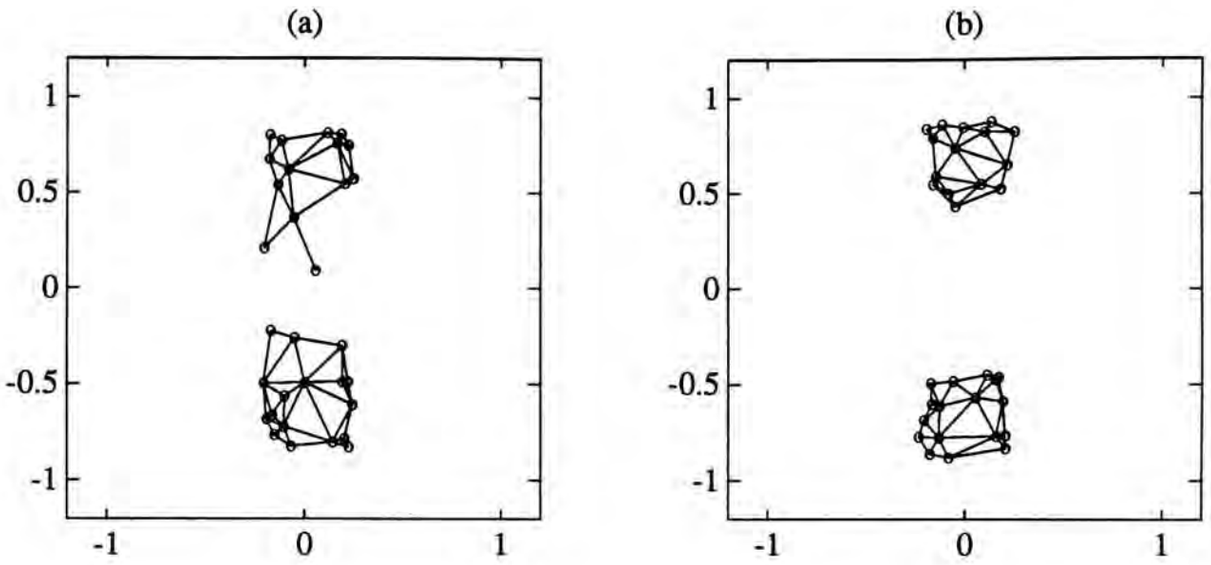

Figure 4.9: (a) $w(Q, N)$ in Example 4.5 at $l = 15000$ after updated. (b) $w(Q, N)$ in Example 4.5 at $l = 20000$ after updated.

Figure 4.10 shows the final induced graph after training. We see that there are 15 neurons in the upper component of X and 17 neurons in the lower component of X. The result is better than that of using the minimal spanning tree in [9]. It is intuitively reasonable since the induced graph is much better than the minimal spanning tree in reflecting the structure of the input space. However, we should still be aware that the disconnectedness of the induced graph is after all a potential problem in this application. As we have mentioned before the links actually reflect information flows. In other words, two neurons in different connected components do not communicate with each other, and thus being disconnected is an irreversible process. Of course the original purpose of the induced graph is to reflect the structure of the input space only, but not to be used in this specific application. Moreover, we think that it is not hard to find out some heuristics such that the graph is not disconnected

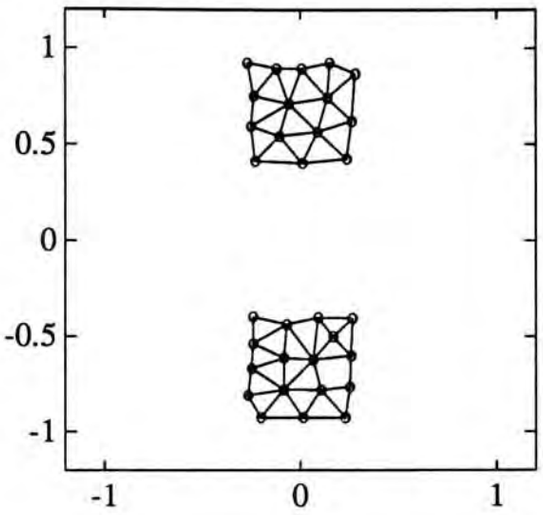

Figure 4.10: The final induced graph in Example 4.5.

until its final stage.

□□□

In the next chapter we shall present some examples of other kinds.

Chapter 5

Given mapping vs induced mapping

5.1 Comparison between given mapping and induced mapping

Suppose we are given a mapping N_G. We would like to find a mapping w_G such that w_G satisfies **[R3]** and (Q, N_G) with w_G preserves the topological order of X. However, such a w_G may not exist. Even if it exists, it cannot always be found because of the lack of a perfect "topological order preserving" algorithm. On the other hand, given a mapping w_I which satisfies **[R3]**, the induced mapping N_I can always be found provided that $w_I(Q) \subset X$. However, since N_I has no pre-specified structure, (Q, N_I) with w_I may be less valuable than (Q, N_G) with w_G if w_G can be found. (We say "may be" since N_I is surely minimal in the sense of Proposition 4.3, but N_G may not. So which one is more valuable depends on

the context.) For example, suppose $N_G = \mathbf{R}^2_{100}$. Then we know that (Q, N_G) with some w_G preserves the topological order of $[0,1] \times [0,1]$. It is difficult to imagine a w_I obtained using competitive learning produces an induced mapping N_I which has a similar periodic structure as N_G. Therefore (Q, N_G) with w_G is more valuable than (Q, N_I) with w_I. From this we see that, if we are given N_G, it is still useful if we can

[G1] compare N_I with N_G, and

[G2] match N_I to N_G.

We study **[G1]** in this section and leave **[G2]** to the next section.

There are many methods to compare two sets. Here we measure

$$\begin{aligned} K\overline{\text{card}(N_G \backslash N_I)} &= \sum_{i=1}^{K} \text{card}(N_G(q_i) \backslash N_I(q_i)) \\ K\overline{\text{card}(N_I \backslash N_G)} &= \sum_{i=1}^{K} \text{card}(N_I(q_i) \backslash N_G(q_i)) \end{aligned}$$

In the remainder of this section we investigate whether Kohonen's algorithm would match N_I to N_G. We monitor $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ at each iteration to see whether they are in general decreasing and eventually able to reach 0. The task is quite similar to monitoring J_1 as we did in Section 3.3. In fact we shall just repeat the examples in Section 3.3, but this time we measure $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ instead of J_1.

(The reader may notice that J_1 is a norm-induced metric while neither $K\overline{\text{card}(N_G \backslash N_I)}$ nor $K\overline{\text{card}(N_I \backslash N_G)}$ is a metric. However,

$$\sqrt{K\overline{\text{card}(N_G \backslash N_I)} + K\overline{\text{card}(N_I \backslash N_G)}} = \|\mathbf{N}_I - \mathbf{N}_G\|_{\text{F}} \tag{5.1}$$

where $\|\mathbf{N}_I - \mathbf{N}_G\|_{\mathrm{F}}$ is the F-norm of $\mathbf{N}_I - \mathbf{N}_G$ and by definition is

$$\sqrt{\sum_{i,j} |(\mathbf{N}_I)_{ij} - (\mathbf{N}_G)_{ij}|^2}$$

Equation 5.1 can be easily obtained by noting that $|(\mathbf{N}_I)_{ij} - (\mathbf{N}_G)_{ij}|^2 = |(\mathbf{N}_I)_{ij} - (\mathbf{N}_G)_{ij}|$ since $(\mathbf{N}_I)_{ij} - (\mathbf{N}_G)_{ij}$ is either -1, 0 or 1.

Therefore, we now in fact consider the negative part and the positive part of $\mathbf{N}_I - \mathbf{N}_G$ separately. It can be seen from the following examples that we should separate $K\overline{\mathrm{card}(N_G \backslash N_I)}$ and $K\overline{\mathrm{card}(N_I \backslash N_G)}$ into two observables because they give different useful information.)

Example 5.1 Here we repeat Example 3.2 in which $X = [-1, 1]$ and $(Q, N_G) = \mathbf{R}_{16}$. $K\overline{\mathrm{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\mathrm{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.1 as a solid line and a dashed line respectively.

□□□

Example 5.2 Here we repeat Example 3.3 in which $X = \mathbf{S}^1 \subset \mathbf{R}^2$ and $(Q, N_G) = \mathbf{S}^1_{16}$. $K\overline{\mathrm{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\mathrm{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.2 as a solid line and a dashed line respectively.

□□□

Example 5.3 Here we repeat Example 3.4. The configuration is the same as that of Example 3.3, except that this time 1-Kohonen's algorithm is used and η is fixed to 0.15. $K\overline{\mathrm{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\mathrm{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.3 as a solid line and a dashed line respectively.

□□□

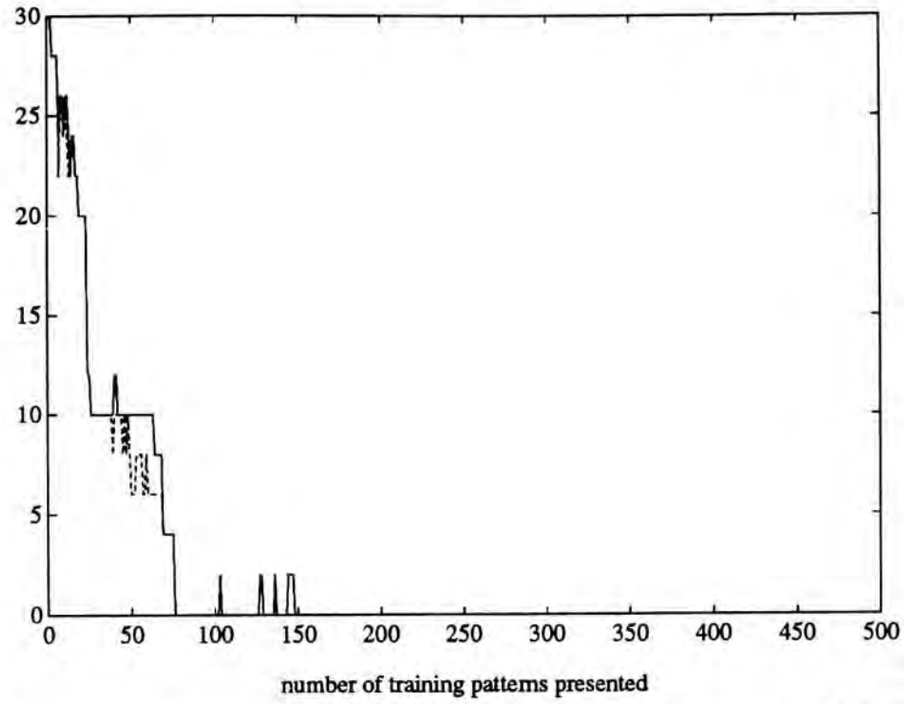

Figure 5.1: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.1.

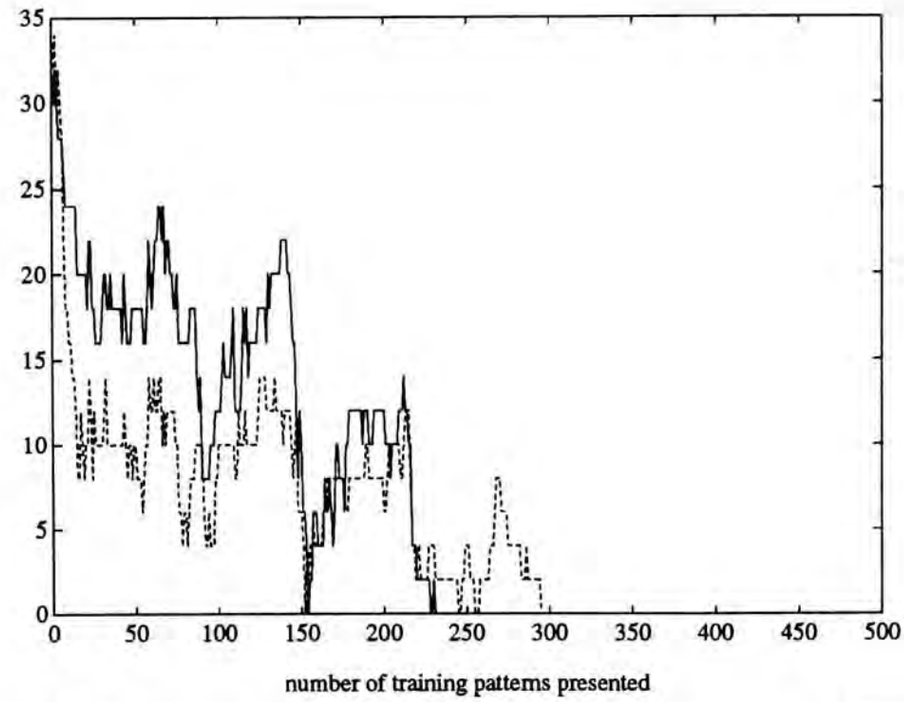

Figure 5.2: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.2.

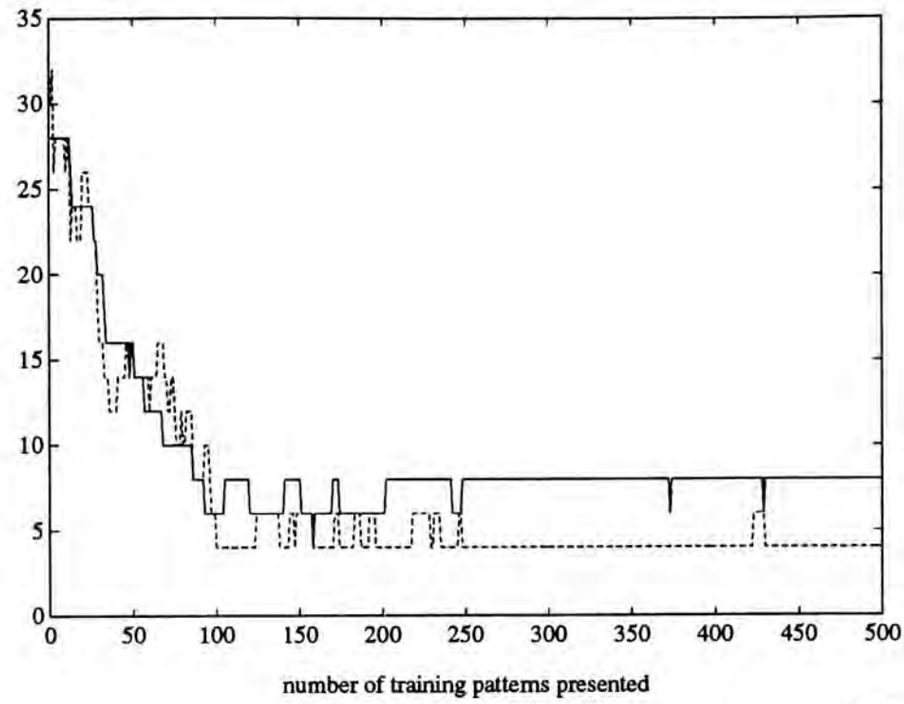

Figure 5.3: The evolutions of $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.3.

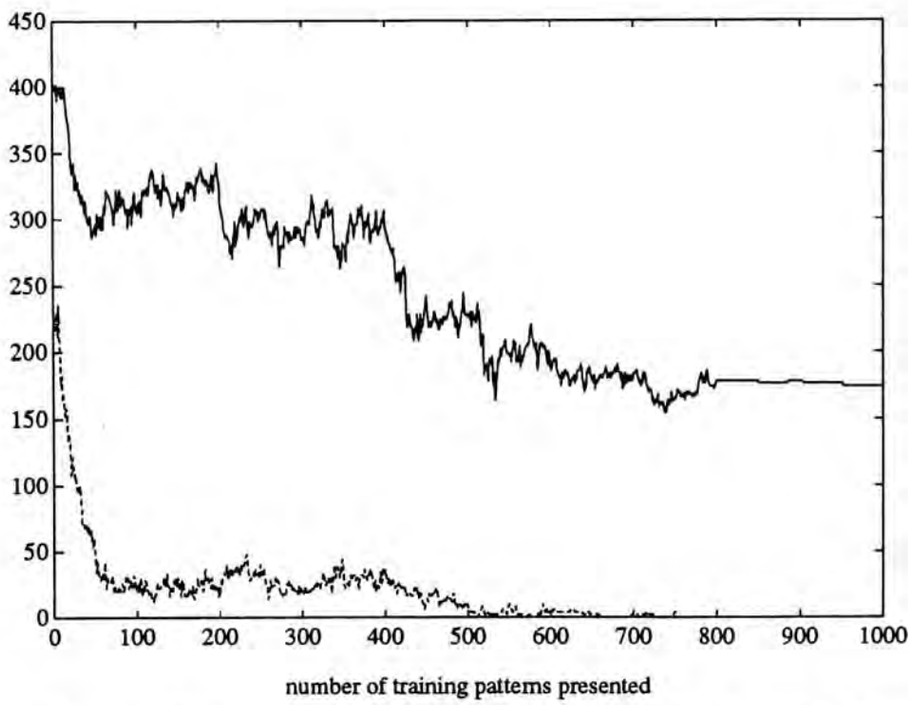

Figure 5.4: The evolutions of $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.4.

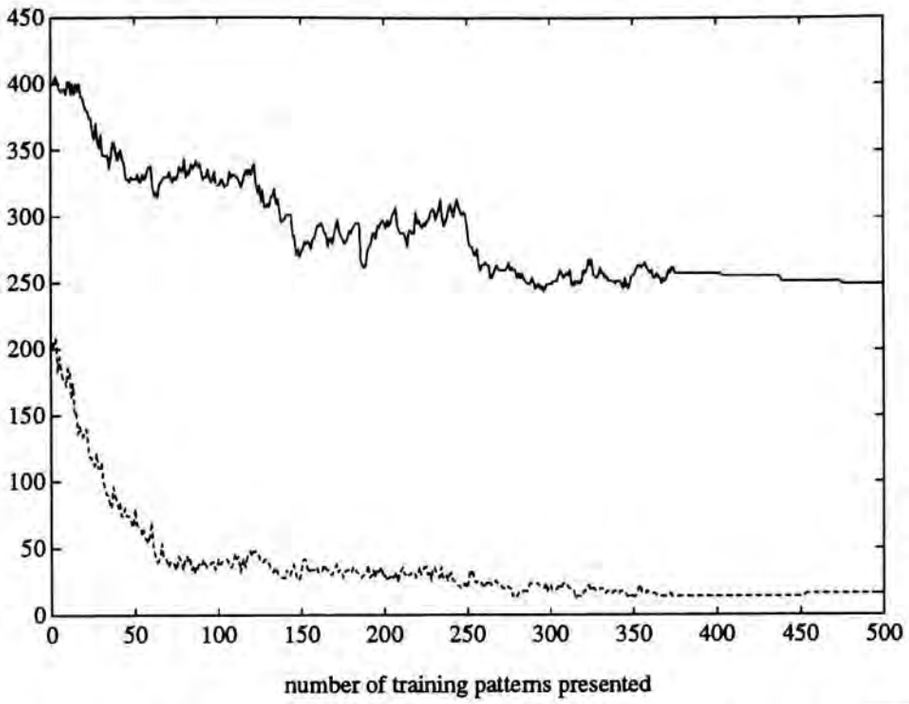

Figure 5.5: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.5.

Example 5.4 Here we repeat Example 3.5 in which $X = [-1, 1]^2$ and $(Q, N_G) = \mathbf{R}^2_{64}$. $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.4 as a solid line and a dashed line respectively.

□□□

Example 5.5 Here we repeat Example 3.6. The configuration is the same as that of Example 3.5 except that $M = 500$ and

$$r = \lfloor \frac{M - l}{M - 1} \times 4 \rfloor$$

$K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.5 as a solid line and a dashed line respectively.

□□□

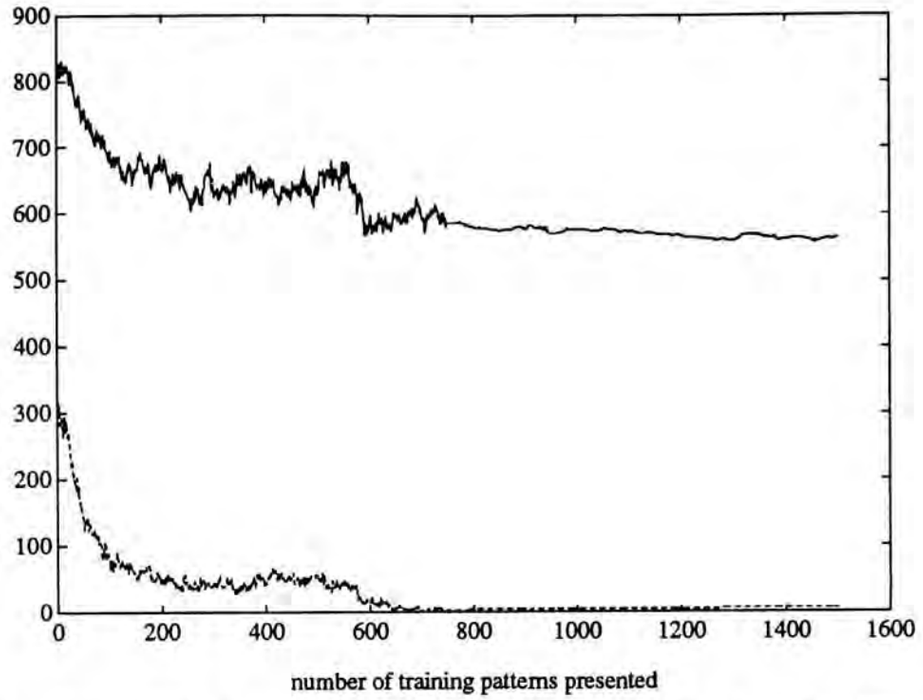

Figure 5.6: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.6.

Example 5.6 Here we repeat Example 3.7 in which $X = [-1, 1]^3$ and $(Q, N_G) = \mathbf{R}^3_{64}$. $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.6 as a solid line and a dashed line respectively.

□□□

Example 5.7 Here we repeat Example 3.8. The configuration is the same as that of Example 3.7 except that

$$r = \lfloor \frac{M - l}{M - 1} \times 3 \rfloor$$

$K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.7 as a solid line and a dashed line respectively.

□□□

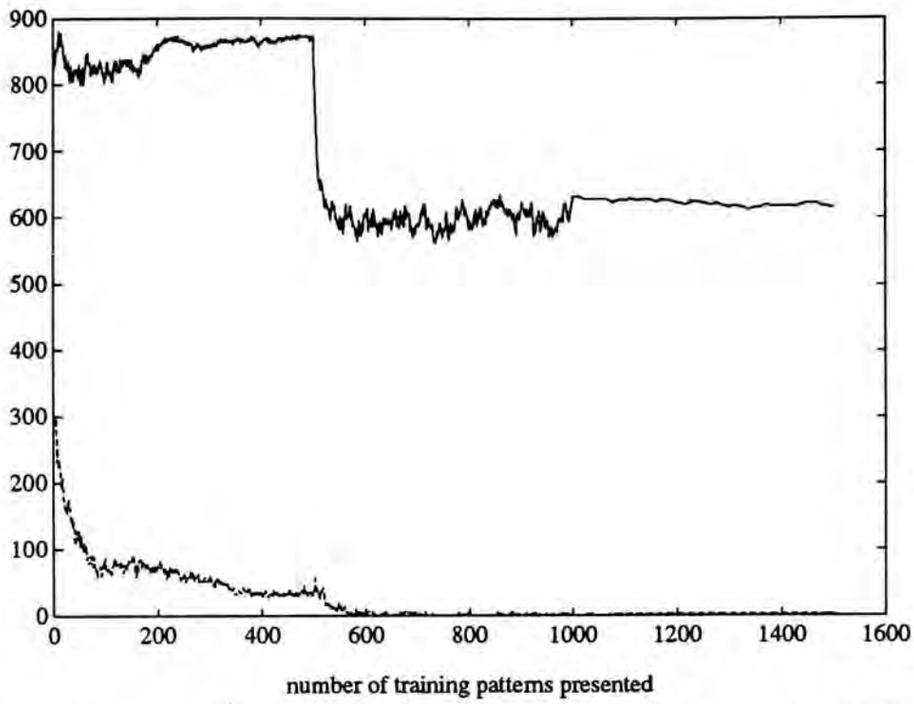

Figure 5.7: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.7.

Example 5.8 Here we repeat Example 3.9 in which $X = \mathbf{S}^1 \subset \mathbf{R}^2$ and $(Q, N_G) = \mathbf{R}_{16}$. $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.8 as a solid line and a dashed line respectively. We see from the final results that $K\overline{\text{card}(N_G \backslash \tilde{N}_I)} = 0$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)} = 2$, that means exactly one link is missed.

□□□

Example 5.9 Here we repeat Example 3.10 in which $X = [-1, 1]$ and $(Q, N_G) = \mathbf{S}^1_{16}$. $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.9 as a solid line and a dashed line respectively. Note that this time the final values of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$, unlike J_1, are completely different from those of Example 5.8.

□□□

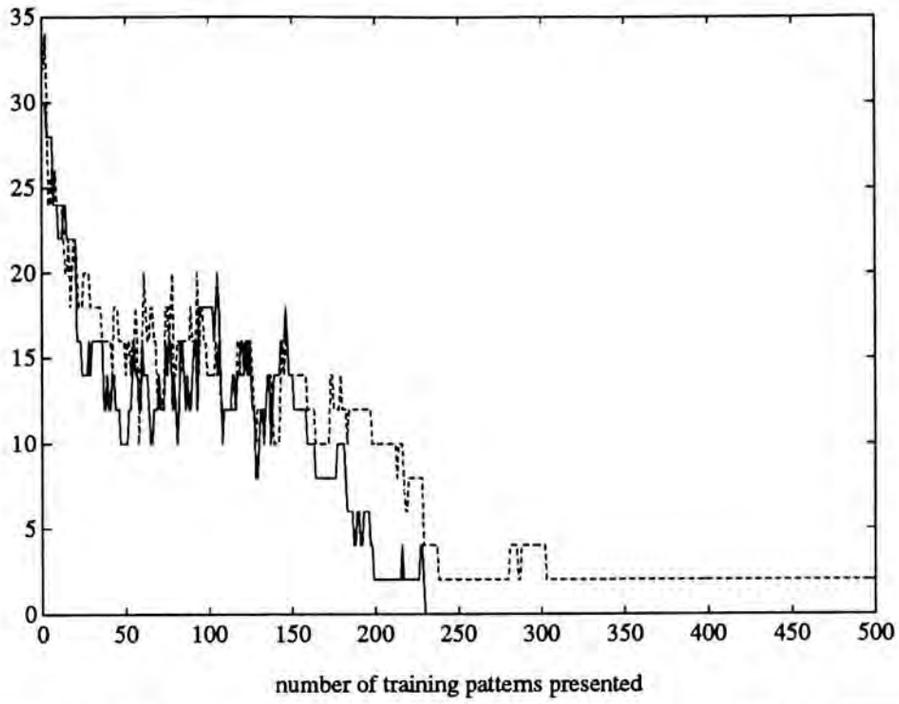

Figure 5.8: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.8.

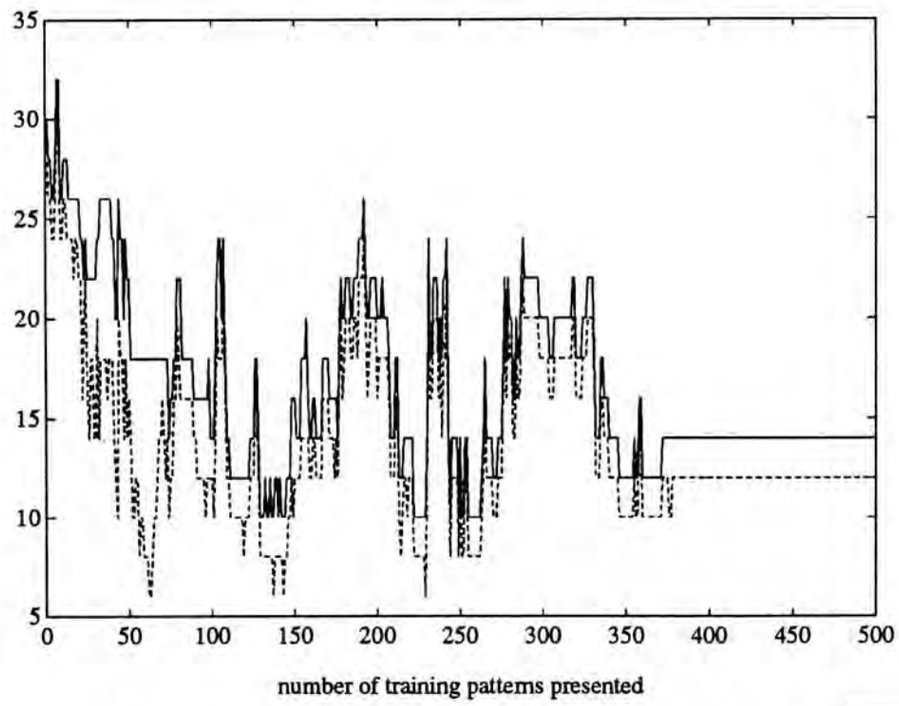

Figure 5.9: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.9.

Example 5.10 Here we repeat Example 3.11 in which $X = [-1, 1]^2$ and $(Q, N_G) = \mathbf{S}_{64}^1$. The final values of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are 8 and 186 respectively.

□□□

As we just mentioned in Remark 3.5, the N_G we have chosen is never minimal when $n > 1$. As a result, even when Kohonen's algorithm succeeds, $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ is never 0 in the above examples where $n > 1$. However, when Kohonen's algorithm succeeds, $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ is always zero, i.e.

$$\tilde{N}_I \subset N_G$$

When Kohonen's algorithm fails, the situation becomes complicated. From Example 5.9 we see that both $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ may not be 0, i.e. N_G may not be a partial graph of $\tilde{N}_I$ and $\tilde{N}_I$ may not be a partial graph of N_G. In Example 5.10, the situation is even more complicated. It seems that if the convergence phase (i.e. when $r = 0$) is longer, it is less likely that $K\overline{\text{card}(N_G \backslash \tilde{N}_I)} = 0$, i.e. it is less likely that $N_G \subset \tilde{N}_I$. The reader may go back to Figure 3.22(a) and 3.23 to make a comparison.

From the results of the above examples, it seems that the tendency of J_1 is clearer than that of $K\overline{\text{card}(N_G \backslash N_I)}$ and $K\overline{\text{card}(N_I \backslash N_G)}$. So, if our only purpose is to check whether Kohonen's algorithm has been successful or not, and if we make sure that $X \in \mathcal{C}$, then we prefer to measure J_1 rather than $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$. However, if $X \notin \mathcal{C}$, then we must measure $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$.

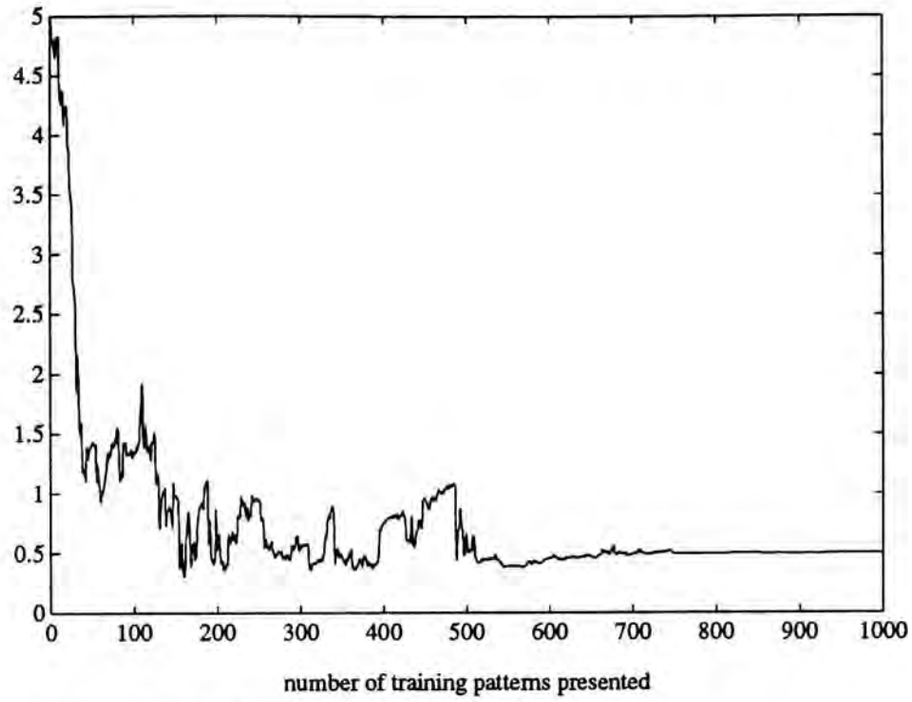

Figure 5.10: The evolution of J_1 in Example 5.11.

Example 5.11 In Example 3.12 we saw that since $X = \mathbf{P}(1,3) \notin \mathcal{C}$, $J_1 \neq 0$ although Kohonen's algorithm succeeded (with $(Q, N_G) = \mathbf{S}^1_{16}$). In this example we examine this situation in more details. The configuration is the same as that of Example 3.12 except that $M = 1000$ instead of 10000. J_1, $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are measured after each training pattern is presented. J_1 is plotted in Figure 5.10 and $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.11. We see that the final value of $J_1 \approx 0.5$. So we would draw a wrong conclusion if we used J_1. However, the final values of both $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are 0. So we would draw a correct conclusion if we used $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$.

□□□

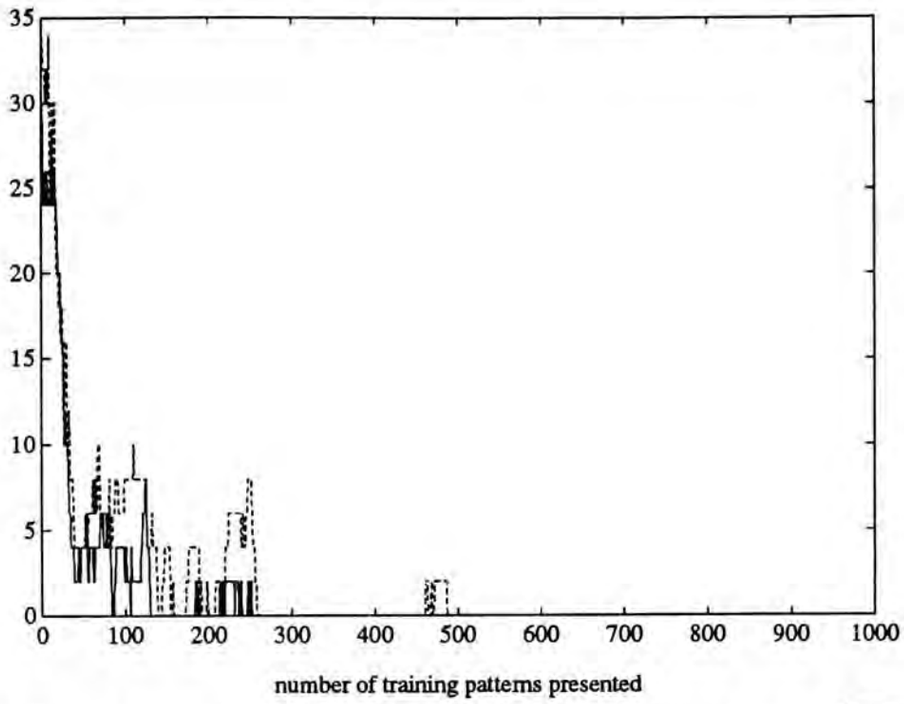

Figure 5.11: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.11.

Example 5.12 In this example we consider again $X = \mathbf{P}(0,1) \notin \mathcal{C}$ and $(Q, N_G) = \mathbf{S}^1_{16}$. The configuration is the same as that of Example 3.13 except that $M = 1000$ instead of 10000. Similarly we monitor J_1, $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ at each iteration. J_1 is plotted in Figure 5.12 and $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ are plotted in Figure 5.13. We see from Figure 5.13 that if we used $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$, we would find that Kohonen's algorithm failed (simply because (Q, N_G) with no w can preserve the topological order of X). Hence we can distinguish between $\mathbf{P}(1,3)$ and $\mathbf{P}(0,1)$.

□□□

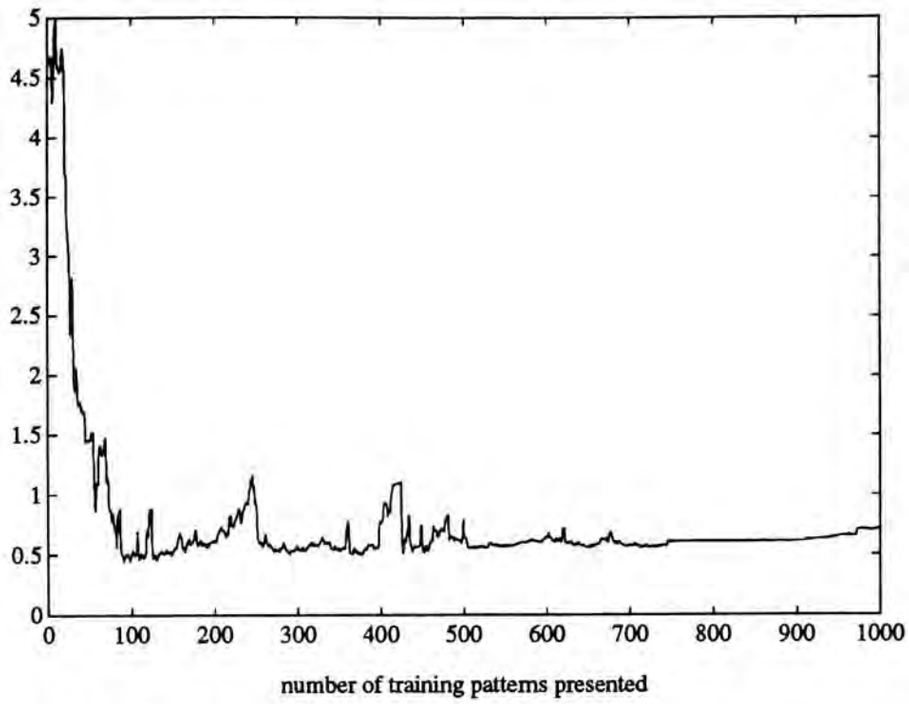

Figure 5.12: The evolution of J_1 in Example 5.1.

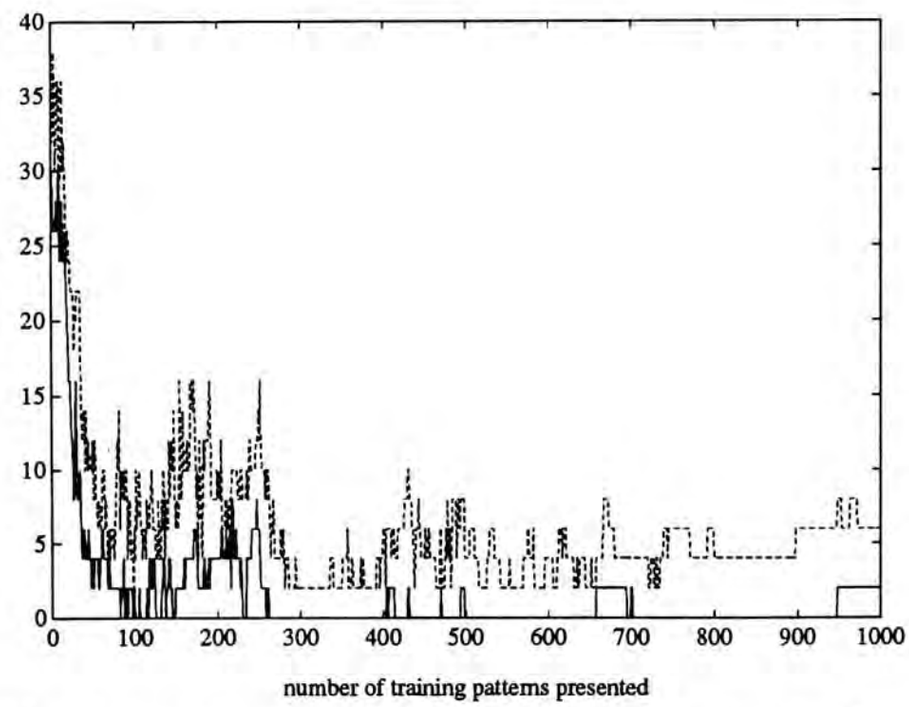

Figure 5.13: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.12.

5.2 Matching the induced mapping to given mapping

In this section we try to solve

Problem 3 Find a mapping w such that the induced mapping N_I is equal to the given mapping N_G.

□

Clearly Problem 3 is not solvable unless N_G is reasonably given, in the sense that it is the induced mapping[1] of some w. In fact, it is not easy to propose a given mapping N_G if we have no information on X. Under these circumstances we may find a w using competitive learning and the corresponding induced mapping N_I would provide valuable information for us to propose a given mapping N_G. Then we may try to find another mapping w which matches N_I to N_G.

In the previous section we see that Kohonen's algorithm can often solve Problem 3 partially by making $N_I \subset N_G$. In this section we give two more examples which may help us to gain more insights into Problem 3.

Example 5.13 Let us recall the 1-Kohonen's algorithm. Suppose $\tilde{x}_l$ is the current training pattern and q_i is the global minimum of $d_{\tilde{x}} \circ w$. Then

$$w^+(q_j) = w(q_j) + \eta(\mathbf{N}_G)_{ji}[\tilde{x}_l - w(q_j)]$$

i.e. the term $\tilde{x}_l - w(q_j)$ is included iff $q_j \in N_G(q_i)$.

Intuitively we can make two neurons in the 1-neighborhood of each other by making them sufficiently close. This motivates us to modify

[1]Therefore, $n\mathbf{R}_K$ is a qualified candidate but $\mathbf{R}_K^n$ is not.

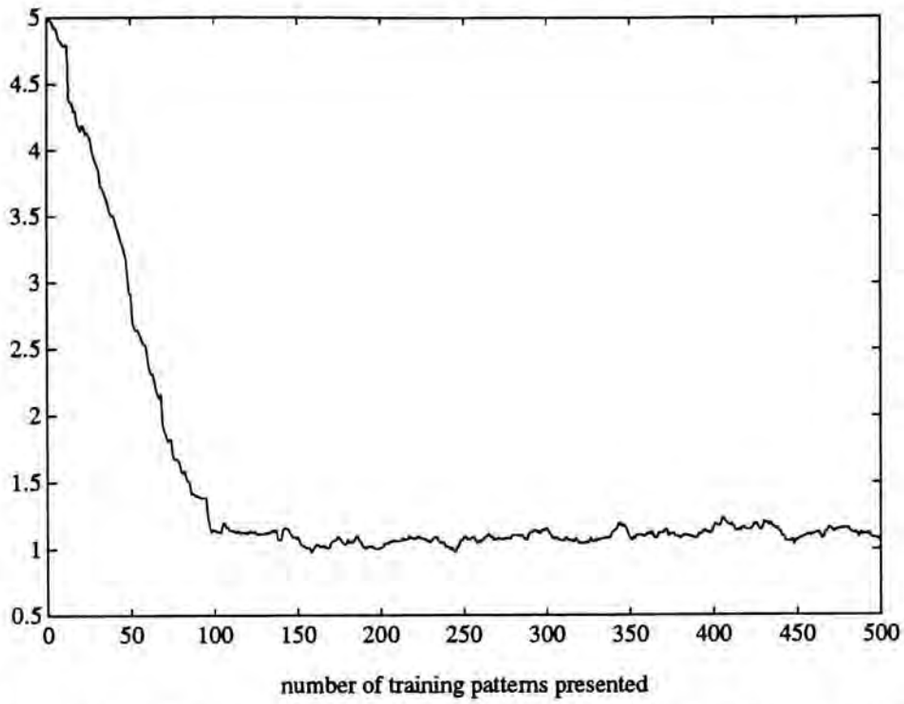

Figure 5.14: The evolution of J_1 in Example 5.13 when 1-Kohonen's algorithm is used.

1-Kohonen's algorithm to

$$w^+(q_j) = w(q_j) + \eta \left[(\mathbf{N}_G)_{ji}[\tilde{x}_l - w(q_j)] + \sum_{k=1}^{K} (\mathbf{N}_G)_{jk}[1 - (\tilde{\mathbf{N}}_I)_{jk}][w(q_k) - w(q_j)] \right]$$

i.e. we include an additional term $w(q_k) - w(q_j)$ if $q_j \in N_G(q_k) \backslash \tilde{N}_I(q_k)$. In other words, if q_j should be in the 1-neighborhood of q_k ($q_j \in N_G(q_k)$) but q_j is now not in the 1-neighborhood of q_k ($q_j \notin N_I(q_k)$), then q_j should be closer to q_k. We hope that such a modification can improve the original 1- Kohonen's algorithm.

We have already seen a failed case of 1-Kohonen's algorithm in Example 3.4 and Example 5.3. For ease of comparison we plot the results again in Figures 5.14–5.16. Let all the parameters remain the same, but this time we use our modified 1-Kohonen's algorithm. The results

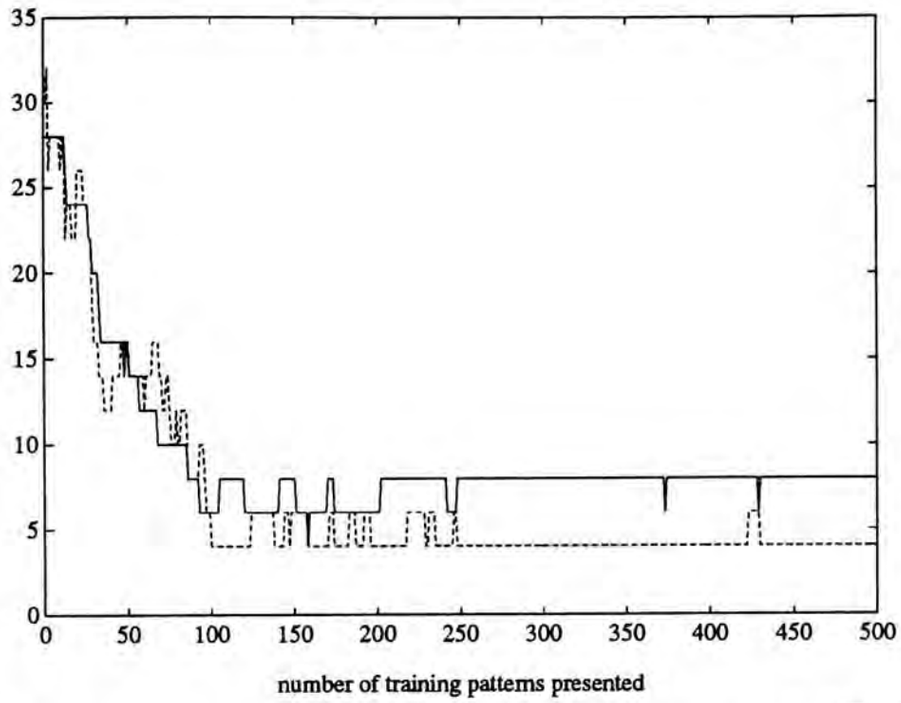

Figure 5.15: The evolutions of $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.13 when 1-Kohonen's algorithm is used.

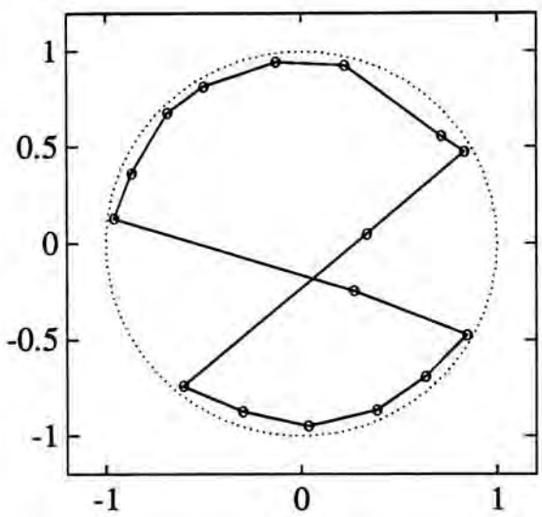

Figure 5.16: $w(Q, N_G)$ in Example 5.13 when 1-Kohonen's algorithm is used.

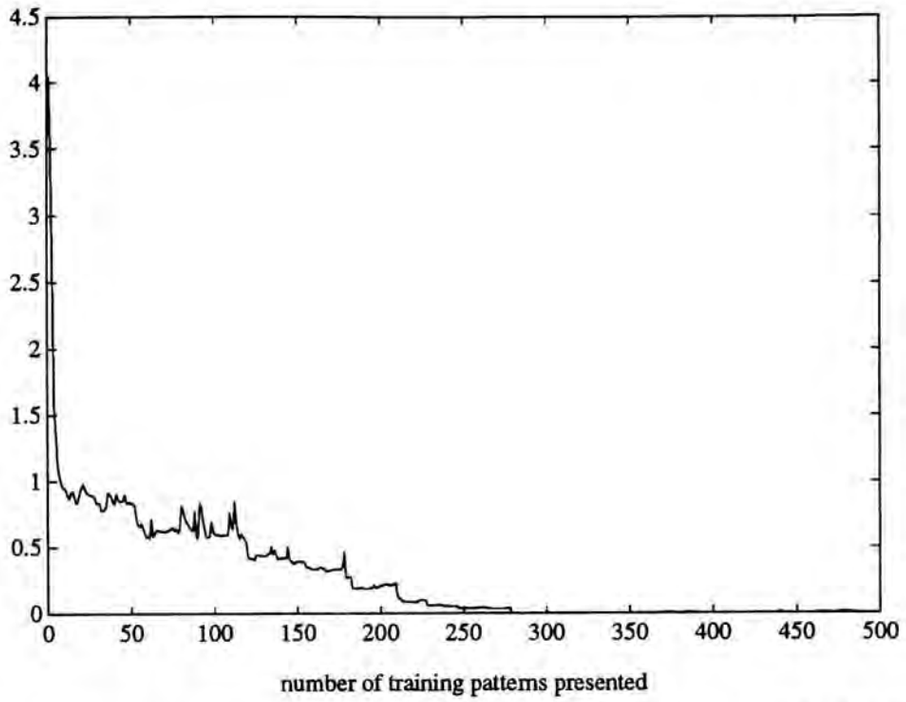

Figure 5.17: The evolution of J_1 in Example 5.13 when modified 1-Kohonen's algorithm is used.

including J_1, $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$, $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ and $w(Q, N_G)$ are plotted in Figures 5.17–5.19. We see that the topological order is successfully preserved. Of course our modified 1-Kohonen's algorithm is not a replacement of traditional Kohonen's algorithm. For example, if we use Kohonen's algorithm with

$$r = \lfloor \frac{M-l}{M-1} \times \frac{K}{3} \rfloor$$

and other parameters unchanged, we can still preserve the topological order (see the results in Figures 5.20–5.22), and clearly the computational complexity of Kohonen's algorithm is much lower than that of our modified 1-Kohonen's algorithm. Our modified 1-Kohonen's algorithm is for illustration purpose only.

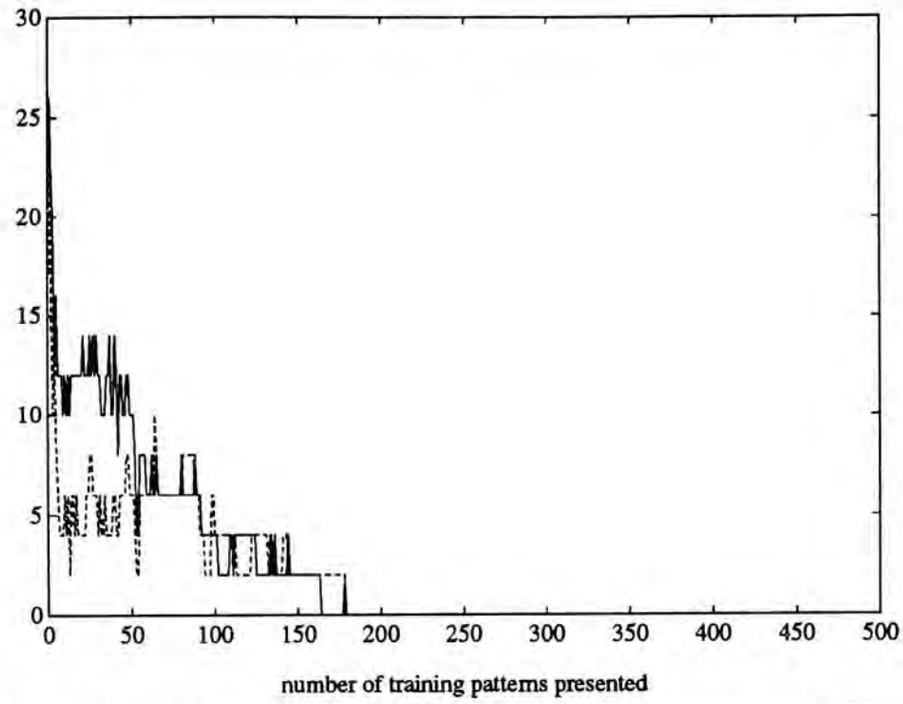

Figure 5.18: The evolutions of $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.13 when modified 1-Kohonen's algorithm is used.

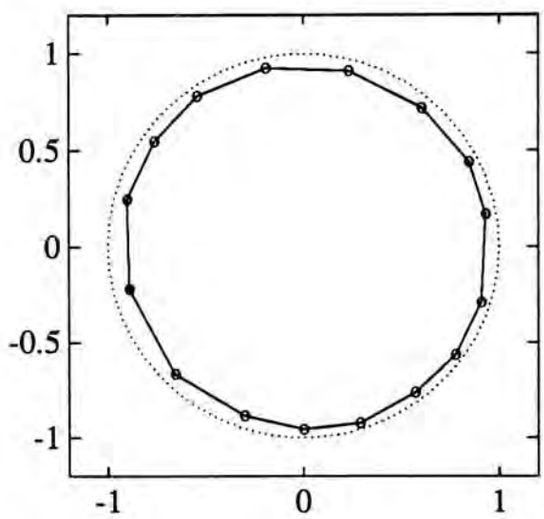

Figure 5.19: $w(Q, N_G)$ in Example 5.13 when modified 1-Kohonen's algorithm is used.

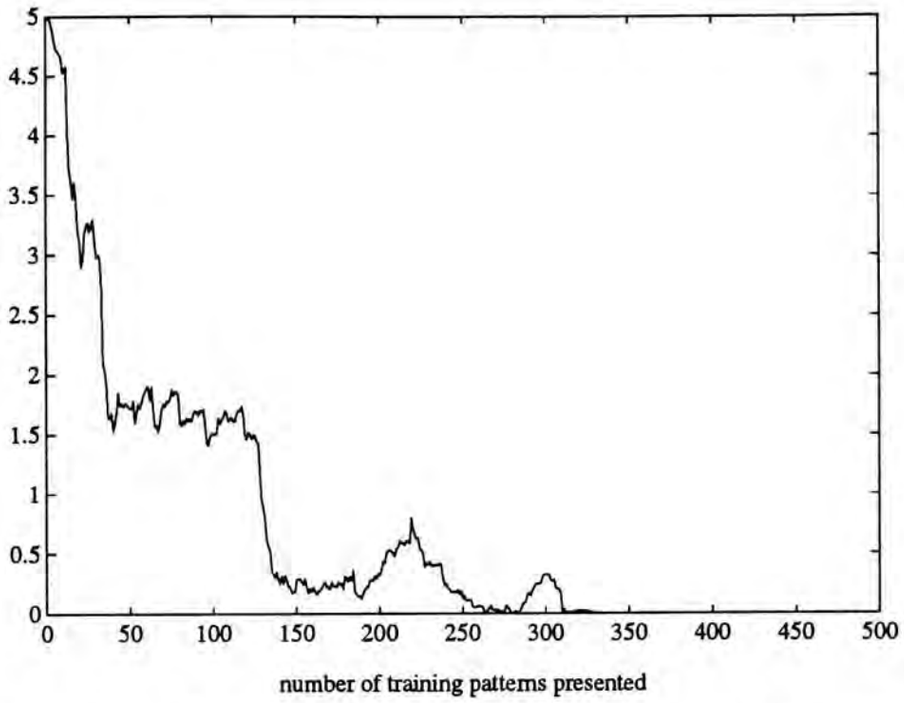

Figure 5.20: The evolution of J_1 in Example 5.13 when Kohonen's algorithm is used.

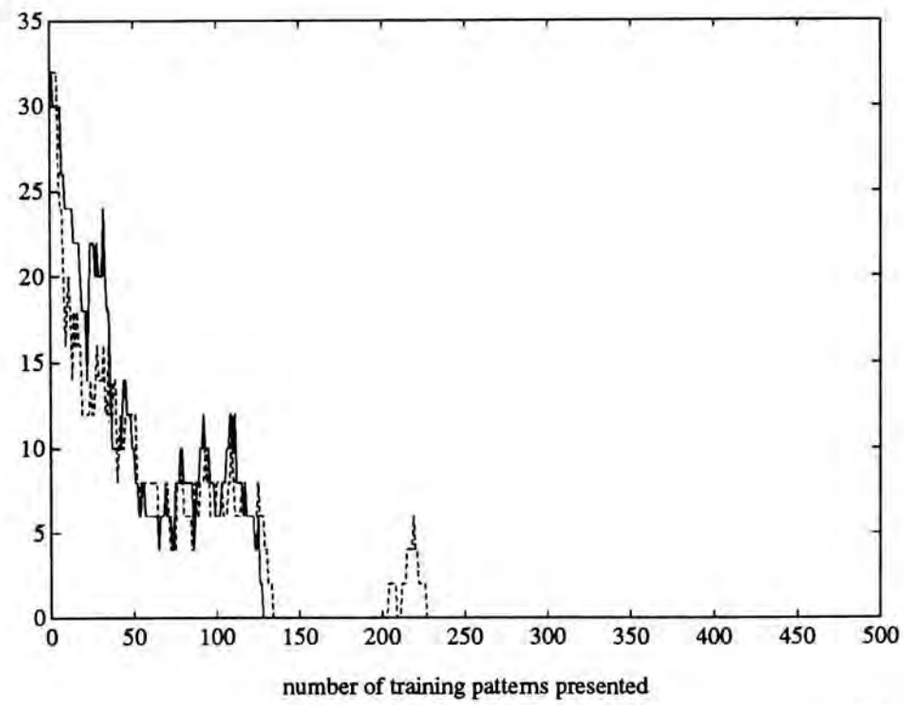

Figure 5.21: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.13 when Kohonen's algorithm is used.

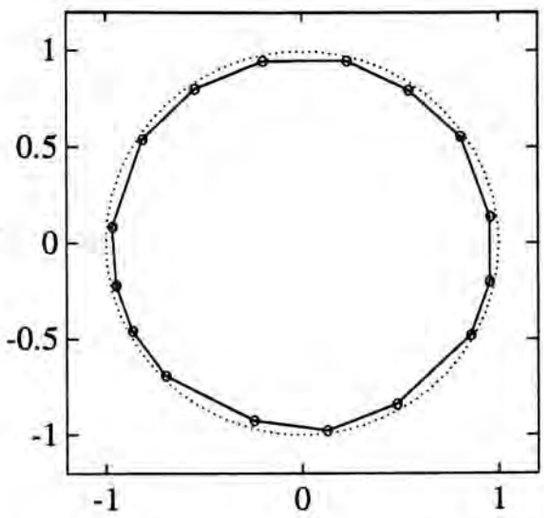

Figure 5.22: $w(Q, N_G)$ in Example 5.13 when Kohonen's algorithm is used.

The reader may wonder, if q_j should be attracted to q_k when $q_j \in N_G(q_k) \backslash N_I(q_k)$, then why q_j should not be repulsed from q_k when $q_j \in N_I(q_k) \backslash N_G(q_k)$. However, we really should not do so. Otherwise it is very difficult to guarantee that $w(Q) \subset X$.

□□□

Example 5.14 Once w changes, the Voronoi regions changes and the induced mapping N_I changes accordingly. It is in general very difficult to predict how N_I changes, except for one case. We call this case a permutation. In this case the set of weights $w(Q)$ does not change, and so does the set of Voronoi regions. Therefore we can easily predict what N_I becomes.

For convenience we use matrix notation. Let q_i be represented by a column vector $\mathbf{q}_i$ such that

$$(\mathbf{q}_i)_{j1} = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$

Then it is clear that

$$(\mathbf{N}_I)_{ij} = \mathbf{q}_i^{\mathrm{T}} \mathbf{N}_I \mathbf{q}_j$$

The matrix representation of w is a $n \times K$ matrix

$$\mathbf{w} = \begin{pmatrix} w(q_1) & w(q_2) & \cdots & w(q_K) \end{pmatrix}$$

such that each column represents a weight. Clearly

$$w(q_i) = \mathbf{w}\mathbf{q}_i$$

Let $\Gamma : Q \to Q$ be an 1–1 onto map. We call Γ a permutation. Suppose we set

$$w^+ = w \circ \Gamma$$

or in matrix form

$$\mathbf{w}^+ = \mathbf{w}\boldsymbol{\Gamma} \tag{5.2}$$

where

$$\boldsymbol{\Gamma} = \begin{pmatrix} \Gamma(q_1) & \Gamma(q_2) & \cdots & \Gamma(q_K) \end{pmatrix}$$

is the matrix representation of Γ. Clearly $\boldsymbol{\Gamma}$ is a $K \times K$ matrix and

$$\Gamma(q_i) = \boldsymbol{\Gamma}\mathbf{q}_i$$

Now it is easy to see that

$$q_i \in N_I^+(q_j) \quad \text{iff} \quad \Gamma(q_i) \in N_I(\Gamma(q_j))$$

i.e.

$$\mathbf{q}_i^{\mathrm{T}}\mathbf{N}_I^{+}\mathbf{q}_j = [\mathbf{\Gamma}\mathbf{q}_i]^{\mathrm{T}}\mathbf{N}_I[\mathbf{\Gamma}\mathbf{q}_j] = \mathbf{q}_i^{\mathrm{T}}\mathbf{\Gamma}^{\mathrm{T}}\mathbf{N}_I\mathbf{\Gamma}\mathbf{q}_j$$

Hence we have

$$\mathbf{N}_I^{+} = \mathbf{\Gamma}^{\mathrm{T}}\mathbf{N}_I\mathbf{\Gamma}$$

Conversely, suppose we can solve

$$\mathbf{N}_G = \mathbf{\Gamma}^{\mathrm{T}}\mathbf{N}_I\mathbf{\Gamma} \tag{5.3}$$

for a permutation $\mathbf{\Gamma}$ and put it into Equation 5.2, then we have a mapping w such that $N_I = N_G$. (Note that it is therefore not an iterating algorithm since we try to solve Problem 3 in one step.)

Equation 5.3 is well studied if we do not restricted $\mathbf{\Gamma}$ to a permutation. Since $\mathbf{N}_I$ is real symmetric, we can always diagonalize $\mathbf{N}_I$ using its eigenvectors, i.e.

$$\mathbf{N}_I\mathbf{E}_I = \mathbf{E}_I\mathbf{\Lambda}_I$$

where

$$(\mathbf{\Lambda}_I)_{ij} = \begin{cases} \text{an eigenvalue of } \mathbf{N}_I & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

and the i-th column of $\mathbf{E}_I$ is the eigenvector corresponding to the eigenvalue $(\mathbf{\Lambda}_I)_{ii}$. Since $\mathbf{N}_I$ is real symmetric, we can normalize the lengths of the eigenvectors such that

$$\mathbf{E}_I^{-1} = \mathbf{E}_I^{\mathrm{T}}$$

and also arrange the eigenvalues such that

$$(\mathbf{\Lambda}_I)_{11} \leq (\mathbf{\Lambda}_I)_{22} \leq \cdots \leq (\mathbf{\Lambda}_I)_{KK}$$

since all eigenvalues are real. (We order the eigenvalues since we want to compare $\mathbf{\Lambda}_I$ and $\mathbf{\Lambda}_G$.) The manipulation of $\mathbf{N}_G$ is similar. Hence we have

$$\mathbf{N}_I = \quad \mathbf{E}_I\mathbf{\Lambda}_I\mathbf{E}_I^{-1} \quad = \mathbf{E}_I\mathbf{\Lambda}_I\mathbf{E}_I^{\mathrm{T}} \tag{5.4}$$

$$\mathbf{N}_G = \mathbf{E}_G \mathbf{\Lambda}_G \mathbf{E}_G^{-1} = \mathbf{E}_G \mathbf{\Lambda}_G \mathbf{E}_G^{\mathrm{T}} \tag{5.5}$$

Putting Equation 5.4 and 5.5 into Equation 5.3 we have

$$\mathbf{E}_G \mathbf{\Lambda}_G \mathbf{E}_G^{\mathrm{T}} = \mathbf{\Gamma}^{\mathrm{T}} \mathbf{E}_I \mathbf{\Lambda}_I \mathbf{E}_I^{\mathrm{T}} \mathbf{\Gamma} = [\mathbf{E}_I^{\mathrm{T}} \mathbf{\Gamma}]^{\mathrm{T}} \mathbf{\Lambda}_I [\mathbf{E}_I^{\mathrm{T}} \mathbf{\Gamma}]$$

Therefore, if

$$\mathbf{\Lambda}_I = \mathbf{\Lambda}_G$$

then we may solve

$$\mathbf{E}_G^{\mathrm{T}} = \mathbf{E}_I^{\mathrm{T}} \mathbf{\Gamma}$$

to obtain

$$\mathbf{\Gamma} = \mathbf{E}_I \mathbf{E}_G^{\mathrm{T}}$$

Of course $\mathbf{\Gamma}$ so obtained may not be a permutation. However, if $\mathbf{\Gamma}$ is very close to a permutation, in the sense that each row and each column of $\mathbf{\Gamma}$ has one and only one dominating entry, then we may still set $\mathbf{w}^+$ according to Equation 5.2 and hope that $\mathbf{N}_I^+$ is much closer to $\mathbf{N}_G$.

We look at an example. Suppose $X = \mathbf{S}^1$, $(Q, N_G) = \mathbf{S}_6^1$ and $w(Q, N_G)$ is as shown in Figure 5.23(a). Written explicitly,

$$\mathbf{N}_G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} -0.2157 & -0.8112 & 0.0202 & 0.6927 & -0.9148 & 0.8530 \\ 0.9387 & 0.5296 & -0.9505 & 0.6599 & -0.3359 & -0.4025 \end{pmatrix}$$

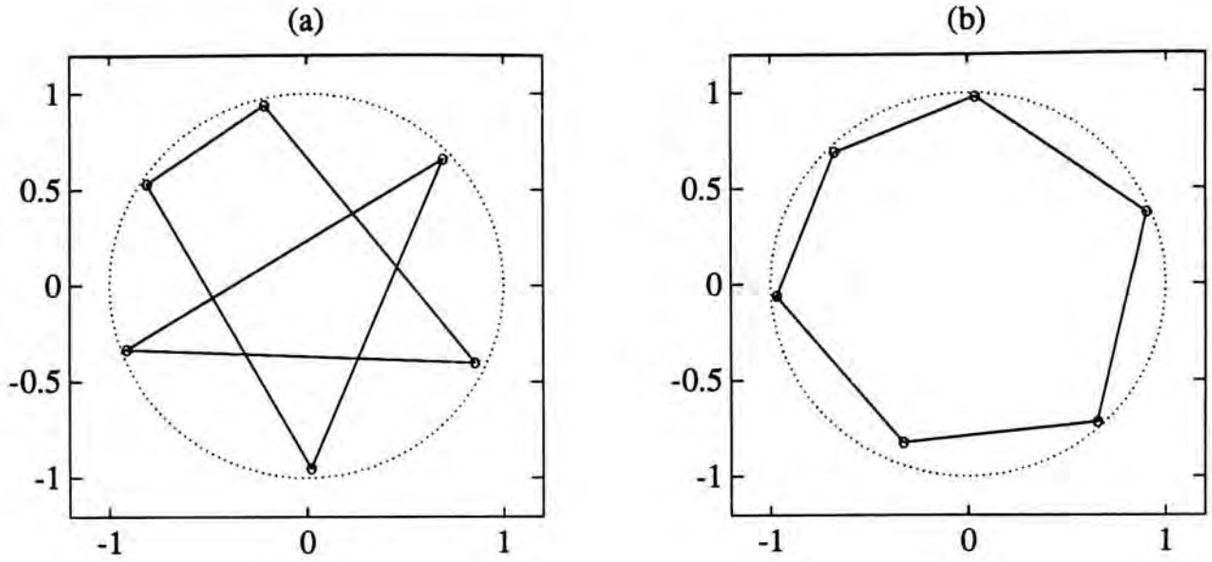

Figure 5.23: (a) $w(Q, N_G)$ in Example 5.14 ($X = \mathbf{S}^1$). (b) $w^+(Q, N_G)$ in Example 5.14 ($X = \mathbf{S}^1$).

$$\tilde{\mathbf{N}}_I = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

At this stage

$$\|\mathbf{N}_G - \tilde{\mathbf{N}}_I\|_F = 4.4721$$

After some calculations we find that

$$\mathbf{\Lambda}_G = \mathbf{\Lambda}_I = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Hence we can use the above method to obtain

$$\mathbf{\Gamma} = \begin{pmatrix} 0.3987 & -0.0673 & -0.1337 & 0.2345 & -0.2650 & 0.8327 \\ 0.8327 & 0.3987 & -0.0673 & -0.1337 & 0.2345 & -0.2650 \\ 0.2345 & -0.2650 & 0.8327 & 0.3987 & -0.0673 & 0.1337 \\ -0.0673 & -0.1337 & 0.2345 & -0.2650 & 0.8327 & 0.3987 \\ -0.2650 & 0.8327 & 0.3987 & -0.0673 & -0.1337 & 0.2345 \\ -0.1337 & 0.2345 & -0.2650 & 0.8327 & 0.3987 & -0.0673 \end{pmatrix}$$

From this we see that $\mathbf{\Gamma}$ is approximately a permutation, in the sense that each row and each column is dominated by an entry 0.8327. Then by Equation 5.2

$$\mathbf{w}^+ = \begin{pmatrix} -0.6750 & -0.9685 & -0.3281 & 0.6542 & 0.9048 & 0.0369 \\ 0.6907 & 0.0625 & -0.8251 & -0.7170 & 0.3733 & 0.9798 \end{pmatrix}$$

$w^+(Q, N_G)$ is plotted in Figure 5.23(b). As expected we have

$$\|\mathbf{N}_G - \tilde{\mathbf{N}}_I^+\|_F = 0$$

i.e. $\mathbf{N}_G = \tilde{\mathbf{N}}_I^+$.

The reader may readily see that the above usage is quite limited. Let us look at another example.

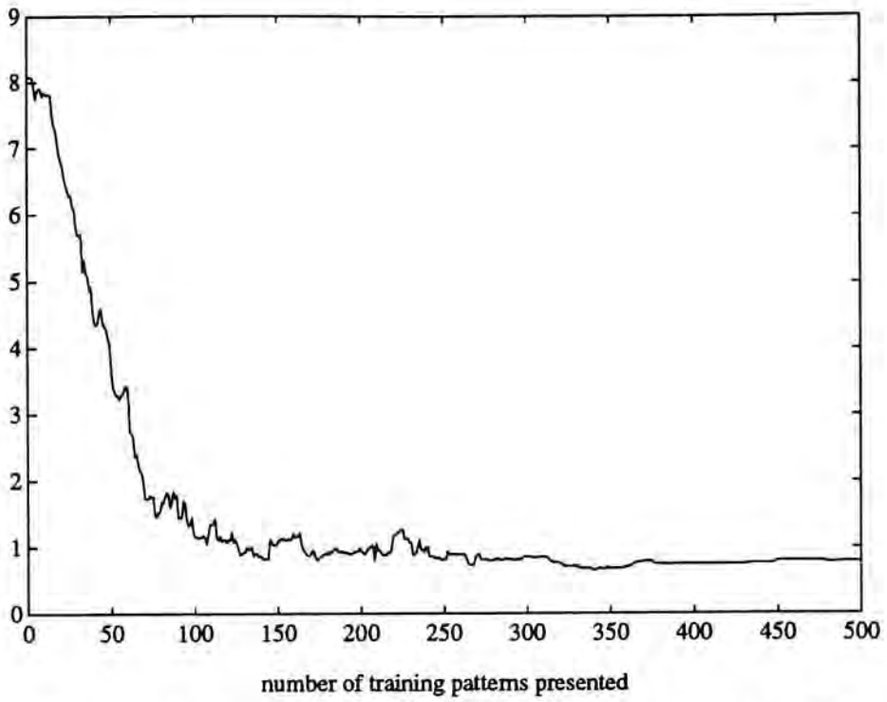

Figure 5.24: The evolution of J_1 in Example 5.14 ($X = [-1, 1]^2$) during first training.

We have already seen a classic way of failure of Kohonen's algorithm in Example 3.6 and Example 5.5. For ease of comparison we plot the results again in Figures 5.24–5.26(a). It is the usual case that further training cannot help, and we must restart the training, perhaps with new parameters. For example, if we just present the same sequence of training patterns again to further[2] train the mapping w, with all parameters remain the same, then $w(Q, N_G)$ is as shown in Figure 5.26(b). Now we modify w to w^+, by the above method we discussed, before we further the training process. Then we see that Kohonen's algorithm is successful this time. (See the results in Figures 5.27–5.29.)

□□□

[2]For "further" we mean that the weights are not initialized again.

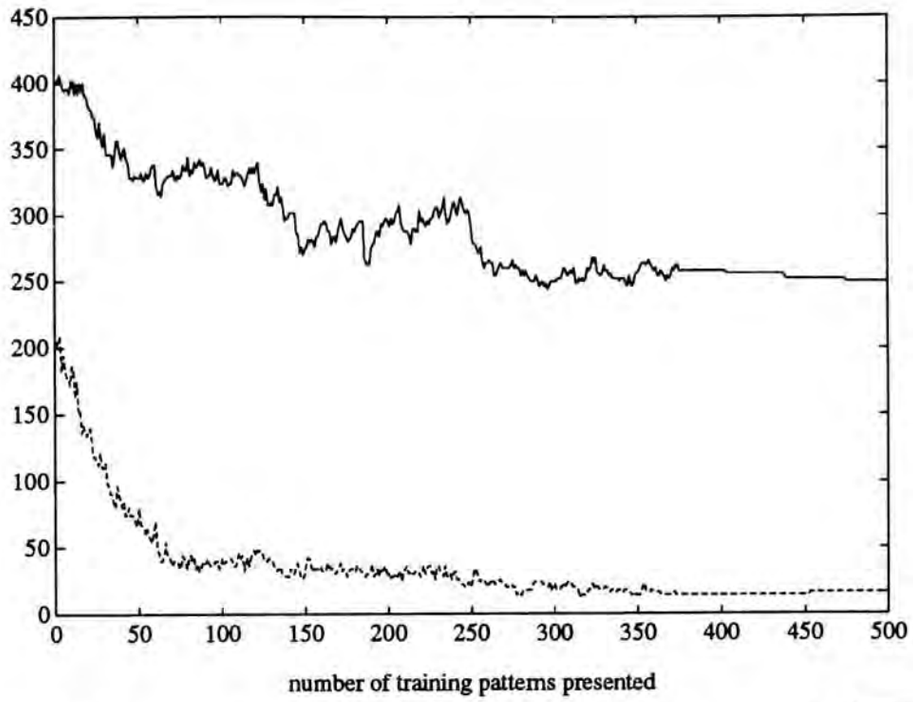

Figure 5.25: The evolutions of $K\overline{\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $K\overline{\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.14 ($X = [-1,1]^2$) during first training.

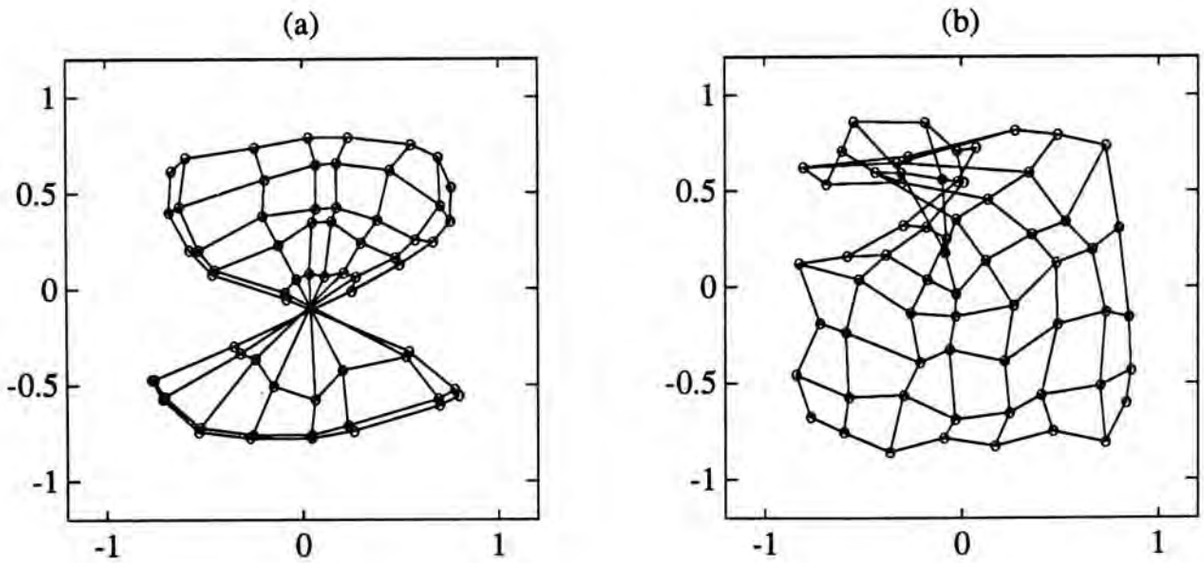

Figure 5.26: (a) $w(Q, N_G)$ in Example 5.14 ($X = [-1,1]^2$) after first training. (b) $w(Q, N_G)$ in Example 5.14 ($X = [-1,1]^2$) after further training.

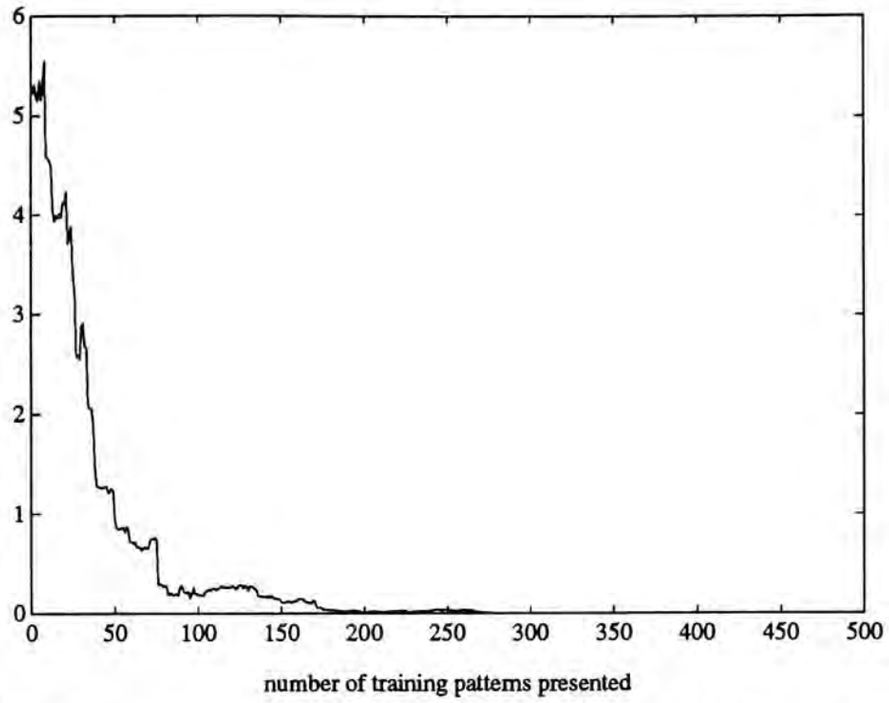

Figure 5.27: The evolution of J_1 in Example 5.14 ($X = [-1,1]^2$) during further training from w^+.

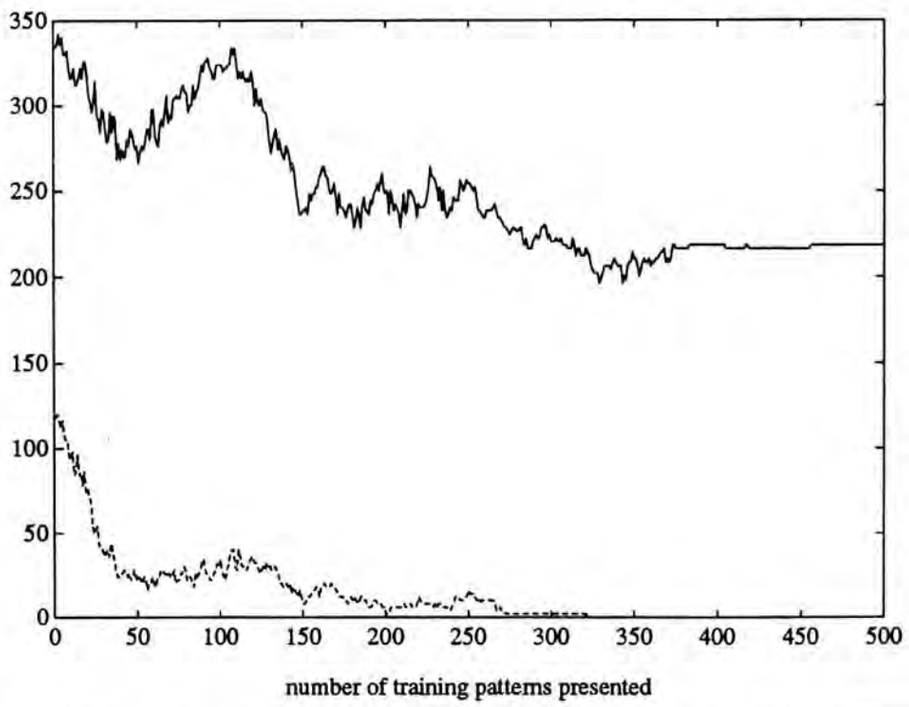

Figure 5.28: The evolutions of $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ (solid line) and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ (dashed line) in Example 5.14 ($X = [-1,1]^2$) during further training from w^+.

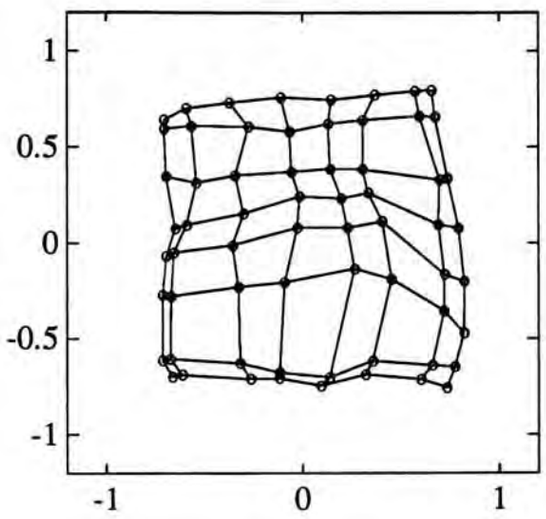

Figure 5.29: $w(Q, N_G)$ in Example 5.14 ($X = [-1, 1]^2$) after further training from w^+.

In the next chapter we study a special topic: the application of the induced graph to the determination of the dimension of the input space.

Chapter 6

A special topic: application to determination of dimension

In this chapter we talk about invariants of graphs. It is clear that for an input space X, many mappings w may satisfy **[R3]** and these mappings may generate different induced graphs. Such graphs may all be good representations of X. Then it is natural to ask what is invariant among these graphs[1]. For example, it is clear from the discussions in Example 5.14 that the matrix of eigenvalues $\mathbf{\Lambda}$ of $\mathbf{N}$ is invariant under permutations. In particular, the largest magnitude of the eigenvalues, $|(\mathbf{\Lambda})_{KK}|$ is invariant under permutations, which is the 2-norm of $\mathbf{N}$. In fact, $\|\mathbf{N}\|_1, \|\mathbf{N}\|_2, \|\mathbf{N}\|_\infty$ and $\|\mathbf{N}\|_F$ are all invariant under permutations. Therefore, for different *known* input spaces $X_{k1}, X_{k2}, X_{k3}, \ldots$, we may use competitive learning to place a set of K neurons in each of them and find out the induced mappings $N_{k1}, N_{k2}, N_{k3}, \ldots$. Then we may compute and record the matrices of eigenvalues $\mathbf{\Lambda}_{k1}, \mathbf{\Lambda}_{k2}, \mathbf{\Lambda}_{k3}, \ldots$ or the norms $\|\mathbf{N}_{k1}\|, \|\mathbf{N}_{k2}\|, \|\mathbf{N}_{k3}\|, \ldots$

[1]Since all graphs considered in this chapter are induced graphs, we shall, for ease of writing, drop the subscript I, i.e. write N_I as N, $\mathbf{N}_I$ as $\mathbf{N}$, $\tilde{N}_I$ as $\tilde{N}$ and $\tilde{\mathbf{N}}_I$ as $\tilde{\mathbf{N}}$.

of $\mathbf{N}_{k1}, \mathbf{N}_{k2}, \mathbf{N}_{k3}, \ldots$. Suppose now we are given an *unknown* input space X. We may similarly obtain an induced mapping N using competitive learning, and then compare the matrix of eigenvalues $\mathbf{\Lambda}$ or the norm $\|\mathbf{N}\|$ of $\mathbf{N}$ with the recorded data, in order to estimate the structure of X. (We remark that the complexity of computing $\|\mathbf{N}\|_1, \|\mathbf{N}\|_\infty$ or $\|\mathbf{N}\|_F$ is much lower than that of computing $\mathbf{\Lambda}$ or $\|\mathbf{N}\|_2$. However, $\mathbf{\Lambda}$ clearly provides more information. As shown in Example 5.14, $\mathbf{\Lambda}$ tells whether one $\mathbf{N}$ can ever match another through a permutation.)

More generally, we may study invariants of graphs for a set of input spaces rather than a single input space. Suppose we want to study a certain property of input spaces. Then for all input spaces $X_{p1}, X_{p2}, X_{p3}, \ldots$ which possess such a property, we may ask what is invariant among $(Q, N_{p1}), (Q, N_{p2}), (Q, N_{p3}), \ldots$. The property we are focusing on in this chapter is the dimension of the input space. Suppose we are given an input space X whose dimension is not known, but we use competitive learning to obtain a graph (Q, N) which is a good representation of X. Of course we may still compare the matrix of eigenvalues $\mathbf{\Lambda}$ with the recorded data $\mathbf{\Lambda}_{k1}, \mathbf{\Lambda}_{k2}, \mathbf{\Lambda}_{k3}, \ldots$ in order to estimate the dimension of X. However this method is relatively[2] computational intensive but not very accurate. Moreover, it seems unnecessary to compute the eigenvalues in order to determine the dimension. One may think that, for example, we may

[2]Of course the computational complexity cannot be regarded as high when the method is compared with other traditional methods which need a topological order preserving algorithm.

roughly guess the dimension by glancing at the average cardinality[3] of the 1-neighborhoods

$$\overline{\text{card}(N)} = \frac{1}{K}\sum_{i=1}^{K}\text{card}(N(q_i))$$

This is really a good start, but we want something better.

6.1 Theory

For ease of clarifying our main theme, we shall temporarily abandon our assumption that Q is finite. If X is unbounded, we shall accept Q to be countably infinite such that all Voronoi regions are bounded, provided that the essemble average

$$\overline{\text{card}(N)} = \frac{1}{K}\sum_{i=1}^{K}\text{card}(N(q_i))$$

can still be defined as a limit.

Suppose we can talk about the dimension of the input space X. It is intuitively true that as the dimension of X increases, $\text{card}(N(q_i))$ also increases. For example, if $X = \mathbf{R}^n$ and $w(Q)$ forms a cubic lattice[4] in X, then all the Voronoi regions are n-dimensional cubes. It is easy to see that $\text{card}(N(q_i)) = 2n + 1$ for all q_i, which increases with the dimension n. Of course the Voronoi regions are in general not all cubes. However, the property that the essemble average $\overline{\text{card}(N)}$ increases with the dimension is generally true.

[3]We have already mentioned different norms of $\mathbf{N}$. The relations between them and the cardinalities of the 1-neighborhoods are

$$\|\mathbf{N}\|_1 = \|\mathbf{N}\|_\infty = \max_{q_i}\text{card}(N(q_i))$$

$$\|\mathbf{N}\|_F = \sqrt{K\overline{\text{card}(N)}}$$

[4]e.g. $w(Q) = \{(i_1, i_2, \ldots, i_n) : i_1, i_2, \ldots, i_n \in \mathbf{Z}\}$.

Example 6.1 We use competitive learning to place 1000 neurons in the input space $[-1,1]^n$. 50000 training patterns are randomly drawn from the input space under uniform distribution. The initial weights are set to the first K training patterns. The learning rate η is 0.15. To ensure that **[R3]** is satisfied, the same training sequence is presented 10 times to train the mapping w. The following table summarizes the results.

n	1	2	3	4
$\overline{\text{card}(\tilde{N})}$	2.9980	6.3260	10.3680	14.6820
n	5	6	7	8
$\overline{\text{card}(\tilde{N})}$	18.6360	22.4360	26.1560	29.1600

In particular, for the case $n = 1$, we can also compute $\overline{\text{card}(N)}$ directly. We know that the only possible induced graph for $[-1,1]$ is $\mathbf{R}_K$. Hence

$$\overline{\text{card}(N)} = \frac{[K-2]\times 3 + 2\times 2}{K} = \frac{998\times 3 + 2\times 2}{1000} = 2.998$$

which matches our experimental result.

The results are plotted in Figure 6.1, from which we see that the data approximately fit into a straight line. We also plot the relative frequencies[5] for different n in Figure 6.2, from which we see that the variance increases with n. In other words, when n is large, the Voronoi regions consist of many different types of n-dimensional polygons.

□□□

Although $\overline{\text{card}(N)}$ gives good hints to the dimension of the input space, using it to determine the dimension has a practical problem and also a theoretical problem. Let us discuss the practical problem first.

[5]i.e., $f(x)$ is the portion of neurons whose 1-neighborhoods have cardinality x.

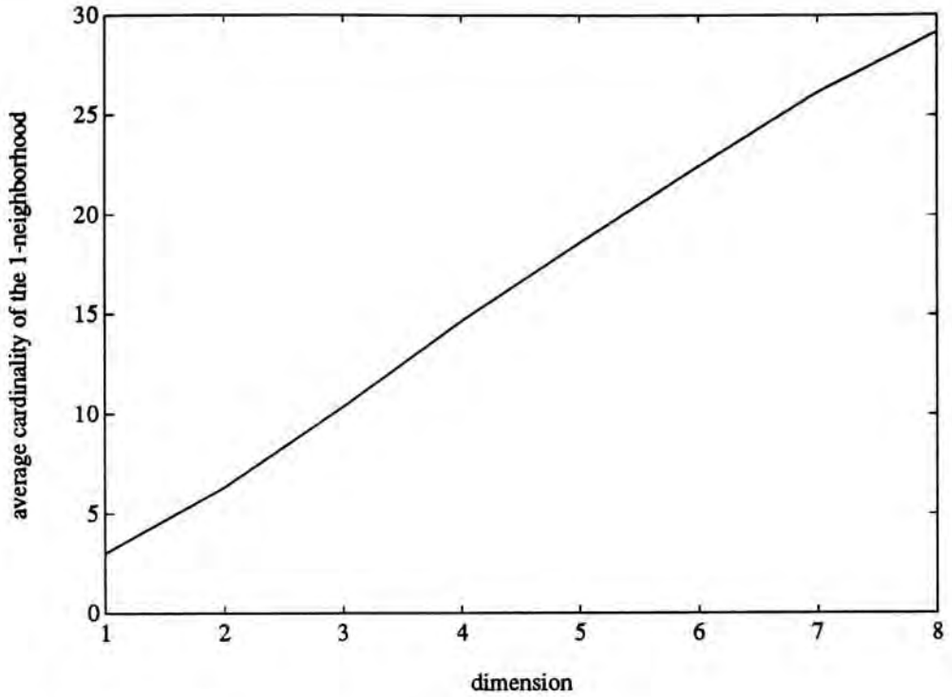

Figure 6.1: $\overline{\text{card}(\tilde{N})}$ vs n.

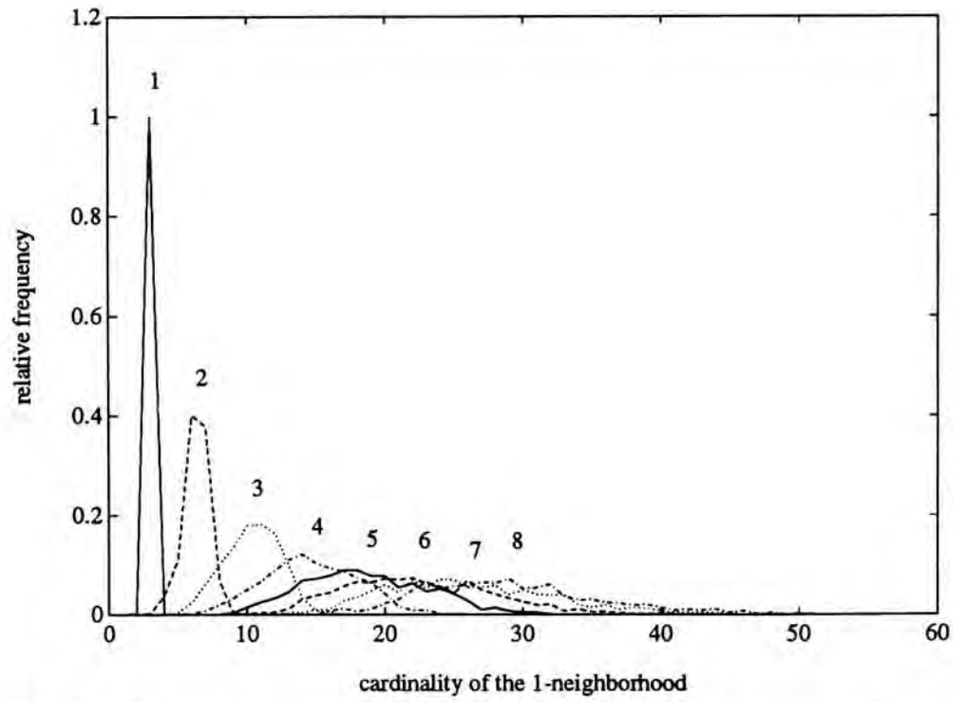

Figure 6.2: Relative frequencies for different n (marked above the means).

Figure 6.1 motivates us to assume that $\overline{\text{card}(\tilde{N})}$ is related to n by a straight line. It seems that, if we measure $\overline{\text{card}(\tilde{N})}$, we can hence know the dimension. However, it has *no* reason to believe that the slope and the intercept of the straight line are invariant under our choices of parameters, such as the number of neurons K and the number of training patterns M. Therefore we do some more experiments.

Example 6.2 All parameters are set the same as those in Example 6.1 except that n is now fixed to 2 and K is now varied from 100 to 1000. The results are summarized in the following table.

K	100	200	300	400	500
$\overline{\text{card}(\tilde{N})}$	6.1200	6.3000	6.3133	6.3850	6.4000
K	600	700	800	900	1000
$\overline{\text{card}(\tilde{N})}$	6.4000	6.4286	6.3525	6.3689	6.3260

□□□

Example 6.3 All parameters are set the same as those in Example 6.1 except that n is now fixed to 2 and M is now varied from 5000 to 50000. The results are summarized in the following table.

M	5000	10000	15000	20000	25000
$\overline{\text{card}(\tilde{N})}$	4.3840	5.2040	5.6180	5.8400	6.0060
M	30000	35000	40000	45000	50000
$\overline{\text{card}(\tilde{N})}$	6.0920	6.1800	6.1960	6.2820	6.3260

□□□

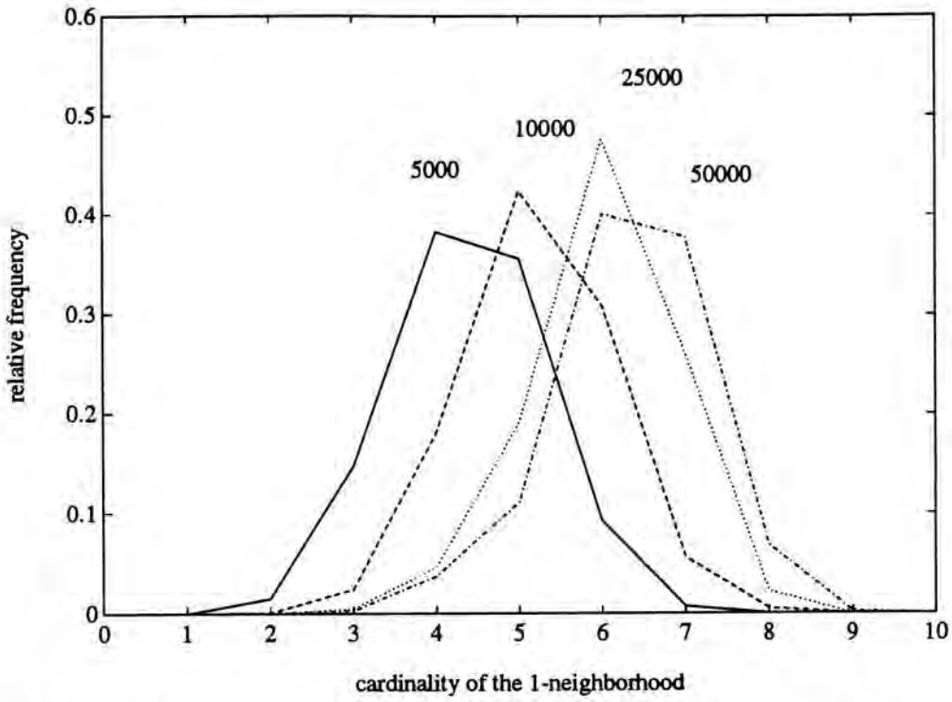

Figure 6.3: Relative frequencies for M =5000, 10000, 25000 and 50000 (with $n = 2$).

From the above two examples we see that $\overline{\text{card}(\tilde{N})}$ is basically invariant under the change of K until K is very small. However, it clearly increases with M (or perhaps the ratio M/K). Figure 6.3 shows the relative frequencies for the cases M =5000, 10000, 25000 and 50000. We think the reason of $\overline{\text{card}(\tilde{N})}$ dependent on M is that the induced graph is practically generated by the training patterns, and as the number of training patterns decreases, some links are missed. Figure 6.4 shows $w(Q, \tilde{N})$ for the case $M = 5000$. In the figure we also plot the training patterns as dots. We see that at the place where a link is missed, there is also a lack of training patterns.

Therefore, if we want to determine the dimension from $\overline{\text{card}(\tilde{N})}$, we must fix the reference. For example, if we are given an input space X of unknown dimension, we may uniformly draw 50000 training patterns from it, and use

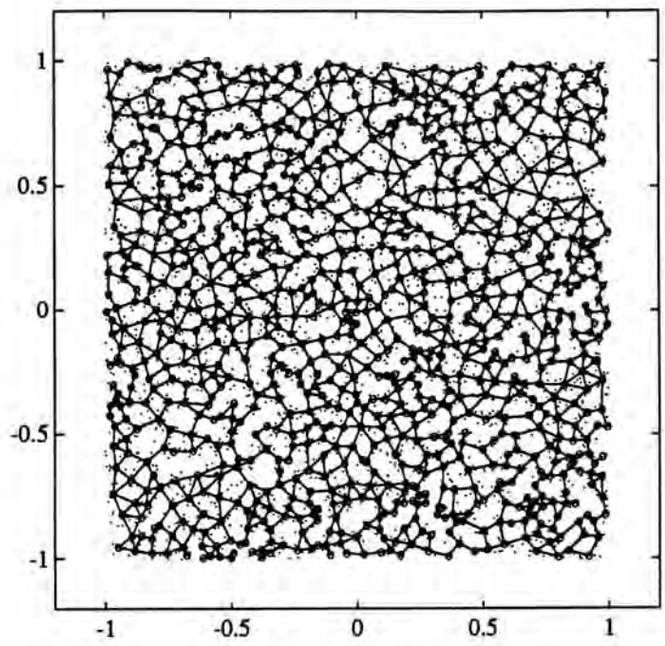

Figure 6.4: $w(Q, \tilde{N})$ for $M = 5000$ in Example 6.3. The training patterns are also shown as dots.

competitive learning to place 1000 neurons in it. Then we may measure $\overline{\text{card}(\tilde{N})}$ and look at Figure 6.1 to estimate its dimension. The reader may notice that the methodology is basically the same as that of comparing the eigenvalues, but this time we compare $\overline{\text{card}(\tilde{N})}$. In both cases we must set up a table which records the data of the known input spaces first, and then compare the data of the unknown input space with those in the table. Such an approach is traditional, although different researchers recorded different things in the table (e.g. [24]). However, we are now more ambitious. We want to determine the dimension directly without making any comparisons.

Next we proceed to the theoretical problem. For a good representation we do *not* specify the mean to obtain the mapping w, but only require w to satisfy certain conditions. Competitive learning is only a specific method to obtain

a good w, but some w which satisfy **[R3]** can (in general) never be obtained using competitive learning. Although competitive learning is extremely simple and good, and we seldom use any other methods, we still want to find out a quantity which varies with the dimension only, but does not vary with the mapping w, provided that w still satisfies **[R3]**. $\overline{\text{card}(N)}$ is clearly *not* such a quantity. For example, let us consider $X = \mathbf{R}^2$. Using competitive learning, $\overline{\text{card}(N)} \approx 6.3260$. However, if $w(Q)$ is a cubic lattice, then $\overline{\text{card}(N)} = 5$. Of course we shall (in general) never obtain a cubic lattice using competitive learning. Besides squares, regular triangles or regular hexagons can also tile the plane. Suppose the Voronoi regions are all regular triangles, then $\overline{\text{card}(N)} = 4$. If the Voronoi regions are all regular hexagons, $\overline{\text{card}(N)} = 7$. All these are good representations of X, but their $\overline{\text{card}(N)}$ are quite different. We hope that we can determine the dimension whichever graph we are given. We do not want to restrict our works to the graphs which we obtain using competitive learning.

To develop our new method, let us recall the following fact. In $\mathbf{R}^n$, the volume of the r-ball of x

$$B_r(x) = \{x' : \|x' - x\| \leq r\}$$

grows as r^n. Some researchers made use of this idea in the determination of the fractal dimension. They counted the number of samples in the r-ball and saw how it grew with the radius r. Here we similarly study how the cardinality of the r-neighborhood grows with the radius of the neighborhood r.

Suppose $X = \mathbf{R}^n$. With each w which satisfies **[R3]** we associate a quantity

$$P(n, r) = \overline{\text{card}(N^{(r)})} = \frac{1}{K} \sum_{i=1}^{K} \text{card}(N^{(r)}(q_i))$$

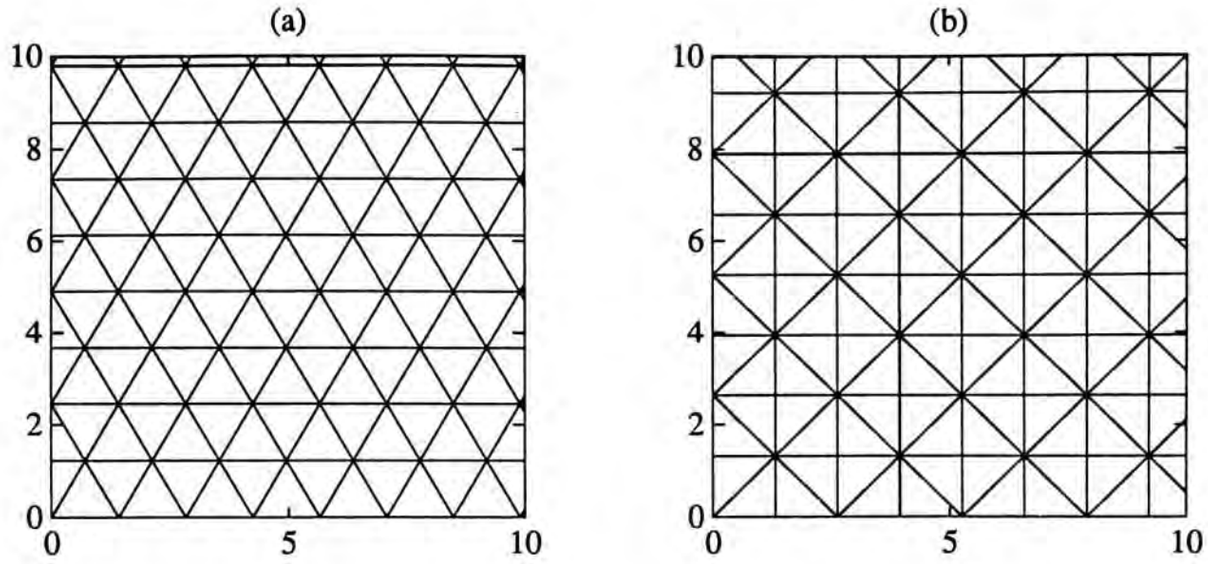

Figure 6.5: Triangles which tile the plane.

In particular, we shall use the notation $P_c(n,r)$ if $w(Q)$ is a cubic lattice, i.e. all the Voronoi regions are n-dimensional cubes.

To motivate the reader, we focus on the case $X = \mathbf{R}^2$ first. We shall soon show that

$$P_c(2,r) = 2r^2 + 2r + 1$$

Figures 6.5(a)–6.7(b) show different types of polygons which tile the plane. In fact they are all Voronoi regions for some mappings w. Then we observe that

$$P(2,r) = \frac{\kappa_2}{4}[2r^2 + 2r] + 1$$

where κ_2 is the number of sides of the polygon. This observation is really surprising. It motivates us to conjecture that for any mapping w which satisfies **[R3]**

$$P(2,r) \approx \kappa[2r^2 + 2r] + 1$$

where κ is dependent on w but independent of r. If similar conjecture is true for

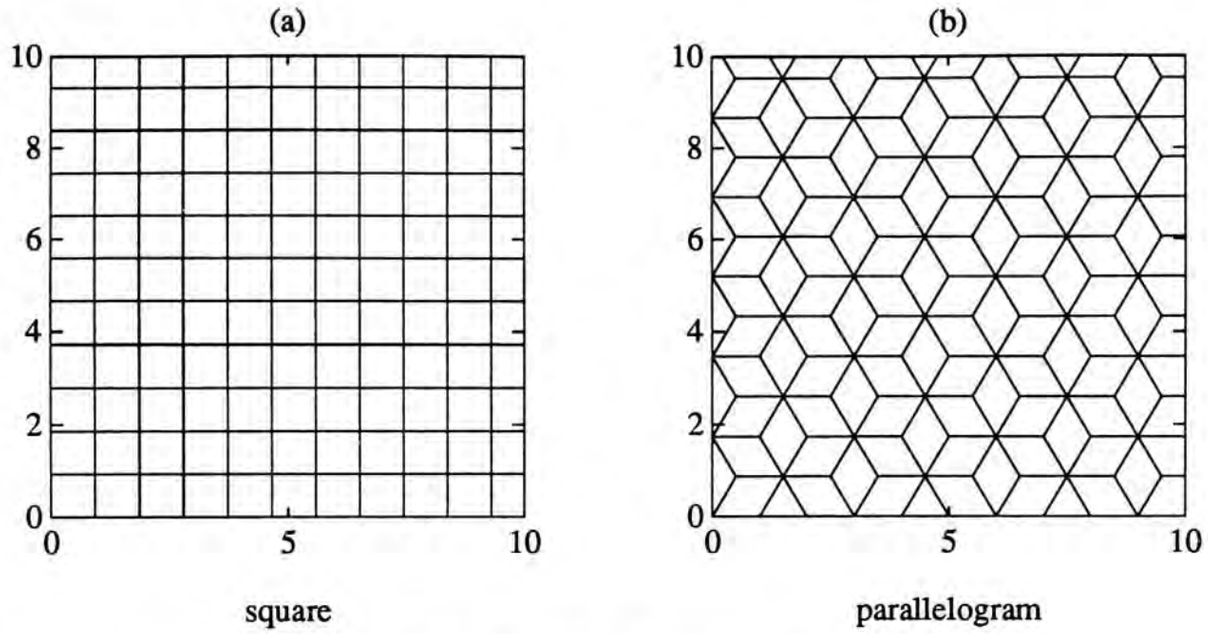

Figure 6.6: Quadrilaterals which tile the plane.

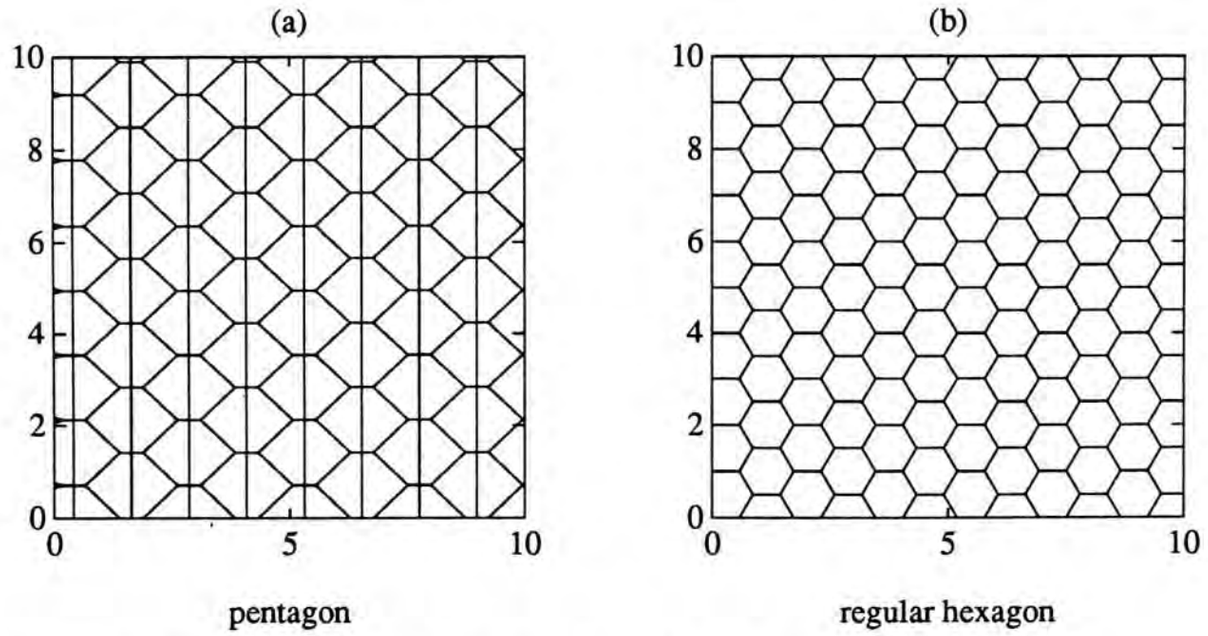

Figure 6.7: Pentagons and hexagons which tile the plane.

any dimension n, we may reduce the study of $P(n,r)$ to the study of $P_c(n,r)$, and focus our attention on the latter only. In other words, to know $P(n,r)$ for any combinations of Voronoi regions, we only need to know $P(n,r)$ for a specific combination of Voronoi regions. This is why the observation surprises us.

In fact it is not hard to calculate $P_c(n,r)$, which is equal to the number of integral n-tuples satisfying the following inequality:

$$\sum_{l=1}^{n} |x_l| \leq r$$

It is clear that

$$\text{number of integral } n\text{-tuples satisfying } \sum_{l=1}^{n} |x_l| \leq r =$$
$$\sum_{m=0}^{r} \left[\text{ number of integral } n-1\text{-tuples satisfying } \sum_{l=1}^{n-1} |x_l| \leq m \right] \times$$
$$\left[\text{ number of integral 1-tuple satisfying } |x_n| = r - m \right]$$

Hence we have proved the following proposition.

Proposition 6.1 $P_c(n,r)$ is given by the following recursive formula

$$P_c(n,r) = 2 \sum_{m=0}^{r-1} P_c(n-1,m) + P_c(n-1,r) \tag{6.1}$$

with the initial condition $P_c(1,r) = 2r+1$.

□

Example 6.4 Applying Proposition 6.1 we have calculated $P_c(n,r)$ for some n:

$$\begin{aligned} P_c(1,r) &= 2r+1 \\ P_c(2,r) &= 2r^2+2r+1 \end{aligned}$$

$$
\begin{aligned}
P_c(3,r) &= \frac{4}{3}r^3 + 2r^2 + \frac{8}{3}r + 1 \\
P_c(4,r) &= \frac{2}{3}r^4 + \frac{4}{3}r^3 + \frac{10}{3}r^2 + \frac{8}{3}r + 1 \\
P_c(5,r) &= \frac{4}{15}r^5 + \frac{2}{3}r^4 + \frac{8}{3}r^3 + \frac{10}{3}r^2 + \frac{46}{15}r + 1 \\
P_c(6,r) &= \frac{4}{45}r^6 + \frac{4}{15}r^5 + \frac{14}{9}r^4 + \frac{8}{3}r^3 + \frac{196}{45}r^2 + \frac{46}{15}r + 1
\end{aligned}
$$

We see that $P_c(n,r)$ grows with r as a polynomial of degree n. As r is very large, $P_c(n,r)$ grows approximately as r^n, similar to the volume of the r-ball. However, since the coefficient of r^n decreases rapidly as the dimension n increases, r is generally not large enough to make the highest order term dominate. For example,

$$\frac{\text{coefficient of } r^{n-1}}{\text{coefficient of } r^n} = \frac{n}{2}$$

Therefore, r must be at least much larger than $n/2$ for the highest order term to dominate all other terms, which is almost impractical. In fact, for r in its practical range, all terms in $P_c(n,r)$ have non-negligible contributions.

It is reasonable to conjecture that, similar to $P_c(n,r)$, $P(n,r)$ also grows with r as a polynomial of degree n. Thus the following method seems workable. We may measure and fit

$$\overline{\text{card}(\tilde{N}^{(1)})}, \overline{\text{card}(\tilde{N}^{(2)})}, \overline{\text{card}(\tilde{N}^{(3)})}, \ldots, \overline{\text{card}(\tilde{N}^{(n+1)})}$$

into a polynomial of degree $n+1$. If the coefficient of r^{n+1} almost vanishes (at least smaller than $2/n\times$ coefficient of r^n), then we may conclude that $P(n,r)$ is a polynomial of degree n, and hence the dimension is determined. However, this method is not good enough for the following two reasons. Firstly, to measure up to $\overline{\text{card}(\tilde{N}^{(n+1)})}$ is time consuming when n and K are large, although we have already employed those practical hints

mentioned in Chapter 2. Secondly, Q is in fact finite. Therefore $P(n,r)$ would not grow indefinitely but eventually saturate. If X is bounded, this happens when the r-neighborhood hits the boundary. As a result, $\overline{\text{card}(\tilde{N}^{(r)})}$ may not fit into the n-th degree polynomial when r is large. Since we have to measure up to $\overline{\text{card}(\tilde{N}^{(n+1)})}$, the method becomes inaccurate when n is large. Here we see that to determine the dimension, we should use local information only, i.e. $\overline{\text{card}(\tilde{N}^{(r)})}$ for small r.

□□□

Substituting r by $r-1$ in Equation 6.1 we have

$$P_c(n,r-1) = 2\sum_{m=0}^{r-2} P_c(n-1,m) + P_c(n-1,r-1) \tag{6.2}$$

Subtracting Equation 6.2 from Equation 6.1 we obtain

$$P_c(n,r) - P_c(n,r-1) = 2P_c(n-1,r-1) + P_c(n-1,r) - P_c(n-1,r-1)$$

i.e.

$$P_c(n,r) = P_c(n-1,r) + P_c(n,r-1) + P_c(n-1,r-1) \tag{6.3}$$

Although the next proposition follows immediately from Equation 6.3, it is really surprising.

Proposition 6.2 $P_c(n,r) = P_c(r,n)$

Proof. We prove it by induction. Suppose the following three are true:

$$\begin{aligned} P_c(n-1,r) &= P_c(r,n-1) \\ P_c(n,r-1) &= P_c(r-1,n) \\ P_c(n-1,r-1) &= P_c(r-1,n-1) \end{aligned}$$

Then

$$\begin{aligned} P_c(n,r) &= P_c(n-1,r) + P_c(n,r-1) + P_c(n-1,r-1) \\ &= P_c(r,n-1) + P_c(r-1,n) + P_c(r-1,n-1) \\ &= P_c(r-1,n) + P_c(r,n-1) + P_c(r-1,n-1) \\ &= P_c(r,n) \end{aligned}$$

The initial condition $P_c(1,r) = 2r+1 = P_c(r,1)$ is trivial.

□

Proposition 6.2 is an indispensable tool in studying the local behaviour of $P_c(n,r)$. To know $P_c(n,r)$ for small r, it suffices to know $P_c(n,r)$ for small n. In other words, if we know how the r-neighborhoods grow in low dimensions, we know how the r-neighborhoods locally grow in all dimensions. Note that it is much easier to obtain $P_c(n,r)$ for small n than to obtain $P_c(n,r)$ for small r. To calculate the latter we need to use the recursive formula Equation 6.1.

Now it is time to propose our conjecture.

Conjecture 1 Let $X = \mathbf{R}^n$. Then for any mapping w which satisfies **[R3]**,

$$P(n,r) \approx \kappa[P_c(n,r) - P_c(n-2,r)] + P_c(n-2,r) \tag{6.4}$$

where κ is independent of r.

□

Remark 6.1 The reader may wonder how we arrive at Conjecture 1. After looking at the case $n = 2$, one may more easily guess that

$$P(n,r) \approx \kappa[P_c(n,r) - 1] + 1$$

However, if we consider a special way to tile $X = \mathbf{R}^3$, we would see that Conjecture 1 is more likely to be true. (If Conjecture 1 is true for any mapping w which satisfies **[R3]**, then it must be true for the following special case.) Suppose we have already tiled $\mathbf{R}^2$. Let $\{V_i : i \in \mathbf{Z}\}$ be the set of Voronoi regions. We have claimed that

$$P(2,r) \approx \kappa[2r^2 + 2r] + 1 \tag{6.5}$$

Clearly $\{V_i \times [j, j+1] : i, j \in \mathbf{Z}\}$ tiles $\mathbf{R}^3$. Using an argument similar to the proof of Proposition 6.1, we see that

$$P(3,r) = 2\sum_{m=0}^{r-1} P(2,m) + P(2,r) \tag{6.6}$$

in this special case. Substituting Equation 6.5 into Equation 6.6 and then applying the following two identities

$$\begin{aligned} \sum_{m=1}^{r} m &= \frac{r[r+1]}{2} \\ \sum_{m=1}^{r} m^2 &= \frac{r[r+1][2r+1]}{6} \end{aligned}$$

we finally obtain

$$\begin{aligned} P(3,r) &\approx \kappa\left[\ \frac{4}{3}r^3 + 2r^2 + \frac{2}{3}r\ \right] + 2r + 1 \\ &= \kappa\left[\ P_c(3,r) - P_c(1,r)\ \right] + P_c(1,r) \end{aligned} \tag{6.7}$$

Similarly $\{V_i \times [j, j+1] \times [k, k+1] : i, j, k \in \mathbf{Z}\}$ tiles $\mathbf{R}^4$. Using the same argument we easily obtain

$$P(4,r) = 2\sum_{m=0}^{r-1} P(3,m) + P(3,r) \tag{6.8}$$

in this special case of special case. Substituting Equation 6.7 into Equation 6.8 and then applying the recursive formula Equation 6.1, we have

$$\begin{aligned} P(4,r) \approx & \kappa \left[\left[2 \sum_{m=0}^{r-1} P_c(3,m) + P_c(3,r) \right] - \right. \\ & \left. \left[2 \sum_{m=0}^{r-1} P_c(1,m) + P_c(1,r) \right] \right] + \left[2 \sum_{m=0}^{r-1} P_c(1,m) + P_c(1,r) \right] \\ = & \kappa \left[P_c(4,r) - P_c(2,r) \right] + P_c(2,r) \end{aligned}$$

The reader may readily notice that what we have demonstrated is in fact if Equation 6.4 is true for all w for dimension n, then it is true for a special subclass of $\{w\}$ for dimension $n+1$.

□□□

Conjecture 1 tries to say that

$$\frac{P(n,r) - P_c(n-2,r)}{P_c(n,r) - P_c(n-2,r)}$$

is independent of r. Hence we may set

$$\frac{P(n,1) - P_c(n-2,1)}{P_c(n,1) - P_c(n-2,1)} \approx \frac{P(n,2) - P_c(n-2,2)}{P_c(n,2) - P_c(n-2,2)} \tag{6.9}$$

Applying Proposition 6.2 we know that

$$\begin{aligned} P_c(n,1) = \quad & P_c(1,n) \quad && = 2n+1 \\ P_c(n-2,1) = \quad & P_c(1,n-2) \quad && = 2n-3 \\ P_c(n,2) = \quad & P_c(2,n) \quad && = 2n^2+2n+1 \\ P_c(n-2,2) = \quad & P_c(2,n-2) \quad && = 2n^2-6n+5 \end{aligned}$$

Putting them into Equation 6.9 we finally obtain the following equation

$$2n^2 + [-2P(n,1) - 2]n + [P(n,1) + P(n,2) - 2] \approx 0 \tag{6.10}$$

Hence, if we measure $\overline{\text{card}(\tilde{N})}$ and $\overline{\text{card}(\tilde{N}^{(2)})}$ and substitute them into Equation 6.10, we may solve n. From the experiments below we find that n should be the smaller root of Equation 6.10, i.e.

$$n \approx \frac{1 + \overline{\text{card}(\tilde{N})} - \sqrt{5 + \overline{\text{card}(\tilde{N})}^2 - 2\overline{\text{card}(\tilde{N}^{(2)})}}}{2} \tag{6.11}$$

For convenience we shall denote the right hand side of Equation 6.11 as $\tilde{n}$. Clearly $\tilde{n}$ is not necessarily an integer.

Here we have used $\overline{\text{card}(\tilde{N})}$ and $\overline{\text{card}(\tilde{N}^{(2)})}$ only, satisfying our hope that only the local information is used. Besides the advantages we have mentioned earlier, this also allows us to solve a quadratic equation (Equation 6.10) only instead of a higher order equation. Practically, if we only need to calculate up to $\overline{\text{card}(\tilde{N}^{(2)})}$, we may make some further special treatments in the computer program in order to reduce the time required to calculate $\text{card}(\tilde{N}^{(r)}(q_i))$, $r = 1, 2$. Firstly, we can combine the calculations of $\text{card}(\tilde{N}(q_i))$ and $\text{card}(\tilde{N}^{(2)}(q_i))$ in only one loop. Secondly, we can calculate $\text{card}(\tilde{N}^{(2)}(q_i))$ without actually calculating $\tilde{N}^{(2)}(q_i)$. (In other words, we can obtain the column sums of nonzero($\tilde{\mathbf{N}}^2$) without having computed the whole matrix nonzero($\tilde{\mathbf{N}}^2$).) This also saves computer memory such that a larger K is allowed.

Example 6.5 The configuration is the same as that of Example 6.1. The results are summarized in the following table.

n	1	2	3	4
$\tilde{n}$	0.9990	2.0139	2.9846	3.9534
n	5	6	7	8
$\tilde{n}$	4.8842	5.8117	6.5589	7.2453

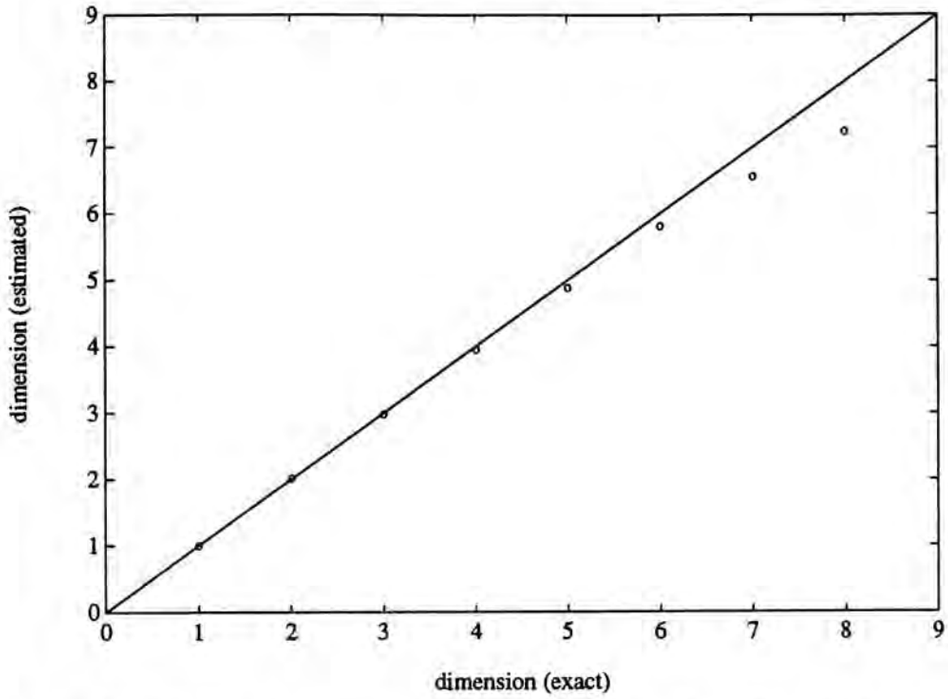

Figure 6.8: $(n, \tilde{n})$ are plotted as circles to see whether they fit into the straight line $\tilde{n} = n$.

For ease of visualization we also plot the results in Figure 6.8 to see whether they fit into the straight line $\tilde{n} = n$. It can be seen that the error increases with the dimension. We think that it is owing to the effect of "hitting the boundary" we mentioned before. As the dimension increases, a larger portion of neurons are placed near the boundary of the input space, especially when the total number of neurons is kept fixed. Note that Conjecture 1 is not true when the r-neighborhoods cannot expand freely. If a neuron is too near the boundary of the input space, its 2-neighborhood may be "clipped", and error is introduced to $\tilde{n}$. We expect that the error can be reduced if we use more neurons in the cases of high dimensions. Anyway, our experimental results can be accurate up to dimension 6, which is really satisfactory.

□□□

Example 6.6 The configuration is the same as that of Example 6.2. The results are summarized in the following table.

K	100	200	300	400	500
$\tilde{n}$	1.8385	1.9201	1.9465	1.9620	1.9784
K	600	700	800	900	1000
$\tilde{n}$	1.9844	1.9896	2.0049	2.0079	2.0139

□□□

Example 6.7 The configuration is the same as that of Example 6.3. The results are summarized in the following table.

M	5000	10000	15000	20000	25000
$\tilde{n}$	1.8507	2.0299	2.0372	2.0360	2.0323
M	30000	35000	40000	45000	50000
$\tilde{n}$	2.0268	2.0244	2.0167	2.0212	2.0139

□□□

From the above two examples we see that $\tilde{n}$ is invariant under the changes of K and M until they become very small. When K is very small, the effect of hitting the boundary becomes significant. When M is small, some links are missed as mentioned before. However, we see that $\tilde{n}$ is much more robust than $\overline{\text{card}(\tilde{N})}$ under the miss of links. This can also be seen from Figure 6.9, which plots the normalized $\tilde{n}$ (solid line) and the normalized $\overline{\text{card}(\tilde{N})}$ (dashed line) as M varies from 5000 to 50000, with $\tilde{n}$ and $\overline{\text{card}(\tilde{N})}$ normalized to 1 at $M = 50000$.

In the last example of this section we want to point out that 10 cycles of presentation of the training patterns may be in fact not needed provided that the number of training patterns is sufficiently large.

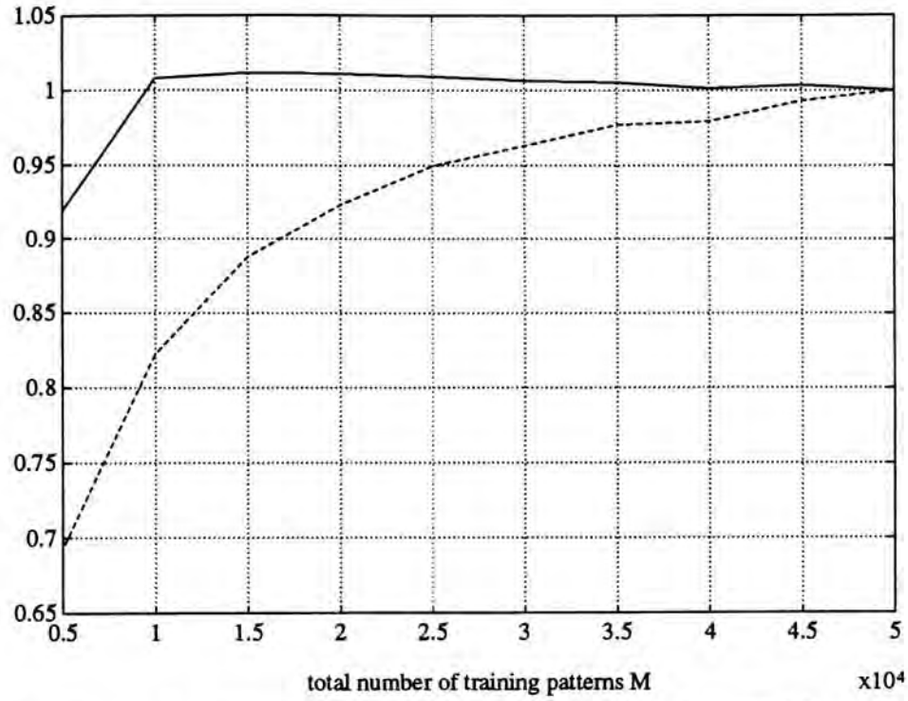

Figure 6.9: Normalized $\tilde{n}$ (solid line) and normalized $\overline{\text{card}(\tilde{N})}$ (dashed line) vs M (reference point at $M = 50000$).

Example 6.8 The configuration is the same as that of Example 6.1 except that the 50000 training patterns are only presented once instead of 10 times. The results are summarized in the following table.

n	1	2	3	4
$\tilde{n}$	0.9990	2.0351	3.0053	3.9699
n	5	6	7	8
$\tilde{n}$	4.9539	5.7988	6.5463	7.2545

□□□

6.2 Advanced examples

In this section we give some more advanced examples.

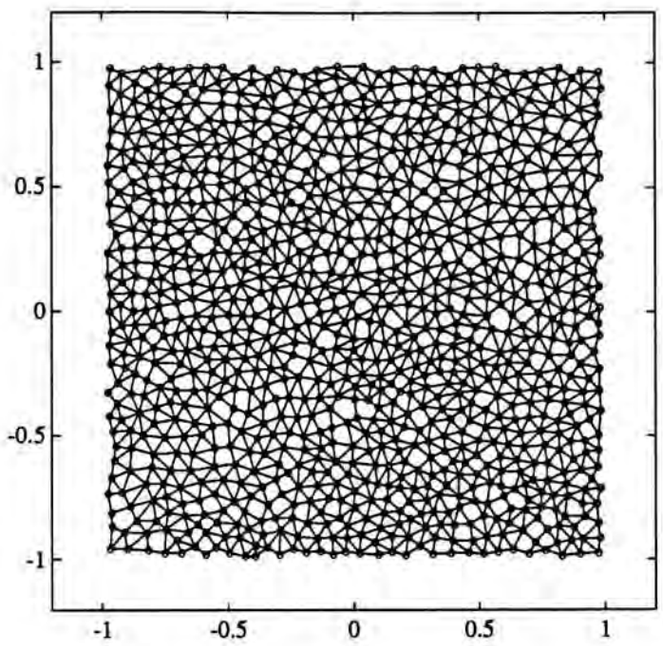

Figure 6.10: $w(Q, \tilde{N})$ for the case $n = 2$ in Example 6.5.

Example 6.9 In this example we want to demonstrate that $\tilde{n}$ is accurate even if the probability distribution defined on X is non-uniform, since w obtained using competitive learning can compensate such an effect. Figure 6.10 plots $w(Q, \tilde{N})$ for the case $n = 2$ in Example 6.5. Suppose now we obtain a new sequence of training patterns from the old one by the following component-wise transformation:

$$\tilde{x}_l^+ = \tilde{x}_l^3$$

Then the training patterns are no more uniformly distributed on X. We present this new sequence of training patterns, with all other parameters remain the same, to train a w and then measure $\tilde{n}$. The result $w(Q, \tilde{N})$ is plotted in Figure 6.11, and $\tilde{n}$ is measured to be 2.0800.

□□□

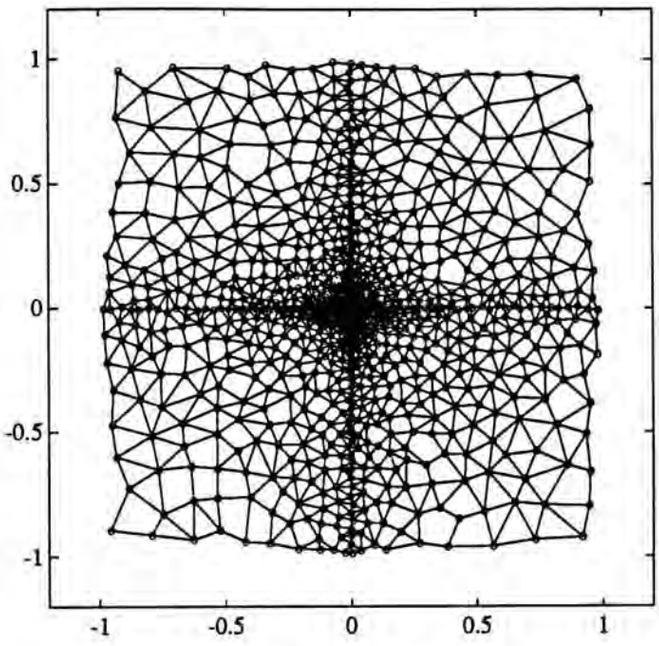

Figure 6.11: $w(Q, \tilde{N})$ in Example 6.9, with the training patterns not uniformly distributed.

Example 6.10 We have mentioned that X is more often bounded instead of the whole $\mathbf{R}^n$. Moreover, X is not necessarily rectangular. Here we consider a special bounded subset of $\mathbf{R}^2$. Let us recall Example 2.3. (In fact we just arbitrarily pick up a better one among Example 2.2, 2.3, 2.4 and 2.5). We have shown at the end of Chapter 4 the induced graph of this w. Here we plot it again in Figure 6.12. For this induced graph $\tilde{n}$ is measured to be 1.5148. We see that a thin 2-dimensional band tends to be classified as an 1-dimensional object. It is caused by the effect of hitting the boundary we have mentioned in the previous section.

□□□

Example 6.11 Although our theoretical work is based on linear input spaces, we hope that it can still be applied to non-linear input spaces.

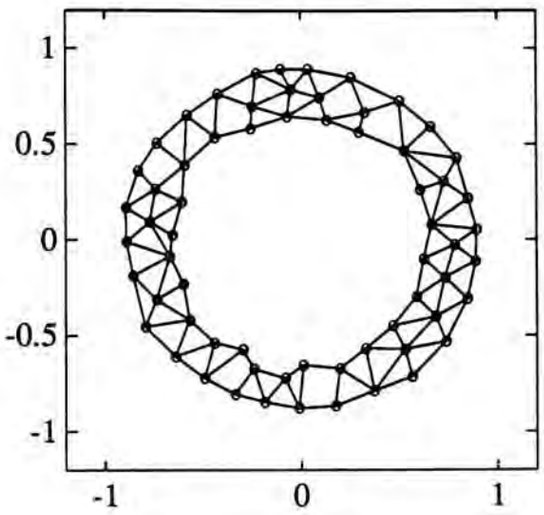

Figure 6.12: $w(Q, \tilde{N})$ in Example 6.10.

Suppose X is non-linear, but at each point of X we can define the tangent plane. Then the dimension at a point x of X can be viewed as the dimension of the tangent plane at this point. Locally at x the behaviour of X is very close to the behaviour of the tangent plane, and as $\tilde{n}$ makes use of local information only, it should still be accurate if the curvature at x is not very large. Here we consider two simple non-linear input spaces:

$$\mathbf{S}^1 = \left\{ (\cos\phi, \sin\phi) : \phi \in [0, 2\pi) \right\}$$

$$\mathbf{S}^1 \times \mathbf{S}^1 = \left\{ \begin{array}{l} \Big([3 + 2\cos\phi_1]\cos\phi_2, [3 + 2\cos\phi_1]\sin\phi_2, 2\sin\phi_1 \Big) : \\ \phi_1, \phi_2 \in [0, 2\pi) \end{array} \right\}$$

Clearly $\mathbf{S}^1$ is a circle in $\mathbf{R}^2$ and $\mathbf{S}^1 \times \mathbf{S}^1$ is a torus in $\mathbf{R}^3$. To obtain the sequences of training patterns, ϕ, ϕ_1 and ϕ_2 are randomly drawn from a uniform distribution on $[0, 2\pi)$, and as a result the training patterns of $\mathbf{S}^1 \times \mathbf{S}^1$ are not uniformly distributed on it.

If all the other parameters are set the same as those in Example 6.1 and Example 6.5, then we have the following results:

X	$\overline{\text{card}(\tilde{N})}$	$\tilde{n}$
$\mathbf{S}^1$	3.0000	1.0000
$\mathbf{S}^1 \times \mathbf{S}^1$	6.5120	2.0736

We see that $\tilde{n}$ is accurate.

If we compare the relative frequencies of the cardinalities of the 1-neighborhoods with those obtained in Example 6.1 for the cases $n = 1$ and $n = 2$,

cardinality of the 1-neighborhood	relative frequency			
	$X = [-1,1]$	$X = \mathbf{S}^1$	$X = [-1,1]^2$	$X = \mathbf{S}^1 \times \mathbf{S}^1$
1	0.0000	0.0000	0.0000	0.0000
2	0.0020	0.0000	0.0000	0.0000
3	0.9980	1.0000	0.0030	0.0000
4	0.0000	0.0000	0.0400	0.0000
5	0.0000	0.0000	0.1440	0.0910
6	0.0000	0.0000	0.4050	0.4630
7	0.0000	0.0000	0.3300	0.3670
8	0.0000	0.0000	0.0770	0.0790
9	0.0000	0.0000	0.0010	0.0000
10	0.0000	0.0000	0.0000	0.0000

then we can easily observe the effect of hitting the boundary. The low end of the relative frequencies for an input space with boundary (such as $X = [-1,1]$ or $X = [-1,1]^2$) is much lower than that for an input space without boundary (such as $X = \mathbf{S}^1$ or $X = \mathbf{S}^1 \times \mathbf{S}^1$) of the same dimension,

while their high ends still coincide. It is because the 1-neighborhoods of the neurons near the boundary have lower cardinalities.

□□□

Example 6.12 In this last example of this section we consider an arbitrary input space. Let

$$X = \Big\{ \Big([t_3^2 + t_4^2 + 1]\cos(t_1), [t_3^2 + t_4^2 + 1]\sin(t_1), [t_3^2 + t_4^2 + 1]\cos(t_2), \\ [t_3^2 + t_4^2 + 1]\sin(t_2), t_3, t_4 \Big) : t_1, t_2 \in [0, 2\pi), t_3, t_4 \in [-1, 1] \Big\}$$

Clearly $X \subset \mathbf{R}^6$ has dimension 4. Moreover, the cross section of X at fixed t_3 and t_4 is in fact a torus in $\mathbf{R}^4$, and thus X is non-linear. To obtain the sequence of training patterns, t_1, t_2 are randomly drawn from a uniform distribution on $[0, 2\pi)$ and t_3, t_4 are randomly drawn from a uniform distribution on $[0, 1)$. The total number of training patterns $M = 50000$. We use competitive learning to place 1000 neurons in X. The learning rate η is 0.15 and the same training sequence is presented 10 times. At last $\overline{\text{card}(\tilde{N})}$ is found to be 14.8780 and $\tilde{n}$ is measured to be 4.1880.

□□□

6.3 Special applications

In this last section we study some interesting applications.

Example 6.13 In this example we consider the dimension of attractor of a chaotic time series. Consider the chaotic Mackey-Glass differential delay equation [16]:

$$\frac{dx}{dt}(t) = -0.1x(t) + 0.2\frac{x(t-\tau)}{1 + [x(t-\tau)]^{10}}$$

where τ is a parameter. (The reader may also see [19, 14, 5].) This system has low dimensional attractors whose dimensions increase with τ [21, 2]. As in [19], the equation is integrated using a fourth order Runge-Kutta method to provide values of x at discrete time steps[6]. The initial condition is a constant function at $x = 0.8$. The single time series is then embedded in a state space by creating state vectors [3, 17, 22, 4, 14]

$$(x(t), x(t-6), x(t-12), \ldots, x(t-6[n-1]))$$

where n is the dimension of the state vectors and must be $\geq$ the dimension of the attractor. We have totally 50000 such state vectors, which are shuffled and then used as training patterns. The total number of neurons $K = 1000$ and the learning rate $\eta = 0.15$. The same training sequence is presented 10 times to train the mapping w. The result is summarized in the following table.

τ	dimension reported in [21, 2]	$\overline{\text{card}(\tilde{N})}$	$\tilde{n}$
17	2.1	5.8640	1.9129
30	3.6	10.0160	3.1885

We see that $\tilde{n}$ is reasonable even for the dimension of attractor of a chaotic time series.

□□□

Example 6.14 In this example we consider the dimension of speech space. (For the importance of finding the dimension of speech, see [1, 24].) In particular, we study the space of LPC coefficients of speech. (However,

[6]We thank Mr. Jones Chui sincerely for providing a C program which integrates the Mackey-Glass differential delay equation and creates the state vectors.

as mentioned in [1, 24] the dimension is basically independent of the representation of speech.) The raw speech data[7] are obtained by sampling at 10 kHz using 12 bit quantization. The duration of each speech frame is 25.6ms, and the starts of consecutive frames are spaced by 10ms. 12 LPC coefficients are obtained for each frame using Durbin's Recursive algorithm (autocorrelation method, [18]). These 12 LPC coefficients form a training pattern in $\mathbf{R}^{12}$. We have totally 43227 such training patterns, which are shuffled before used. The total number of neurons $K = 1000$ and the learning rate $\eta = 0.15$. The same training sequence is presented 10 times to train the mapping w. At last we find that $\overline{\text{card}(\tilde{N})} = 19.5500$ and $\tilde{n} = 4.5410$. The dimension we have estimated is quite consistent with the results in [24, 23], in which other methods were employed to obtain the dimension.

□□□

[7]We thank Dr. Edmund Lai sincerely for providing the raw speech data and a C program which calculates the LPC coefficients.

Chapter 7

Conclusion

In this thesis we have considered the set of neurons Q as the underlying set of a fully looped symmetrical directed graph (Q, N). In other words, we have superimposed a graph structure on the set of neurons. This rigorous graph structure replaced the vague "spatial relationships" of the neurons which had been traditionally used. Then we defined the local minimum of a real-valued function defined on this graph. By considering the distance function as a real-valued function, we defined primarily what preserving the topological order means. The definition of preserving the topological order was generalized gradually, from X being convex, to X being path-connected, to at last each component of X being path-connected.

Accompanied the definition of preserving the topological order was a quantity J_1 which is the average number of local minima of $d_{\bar{x}_l} \circ w$ minus 1. J_1 is expected to be almost 0 when the topological order is preserved. Hence we can measure J_1 to detect whether Kohonen's algorithm succeeded in high dimensions. This formed one of the most important applications of J_1 in this thesis.

One main problem of using J_1 is that practically it only works when $X \in \mathcal{C}$. Of course we needed not confine the class of input spaces to $\mathcal{C}$ in our theory since we can consider geodesic distance, but practically we can only measure the direct distance. On the other hand, the induced graph, which was obtained by formulating the theory in terms of Voronoi regions instead of local minima, has a practical approximation which is surprisingly good even when direct distance is used. Of course it was a good reason for us to direct our attention to the induced graph rather than the local minima, but the main motivation to do so was that the concept of induced graph is so natural that every one can easily understand.

The purpose of the induced graph is to reflect the structure of the input space, and hence we can study the structure of the input space by studying the induced graph instead. Roughly speaking, the induced graph is the "minimal" graph (in the sense of Proposition 4.3) which preserves the topological order of the input space. We had added some remarks to clarify the idea of the induced graph. With the concepts of preserving the topological order and the induced graph, we discussed the vague idea of when (Q, N) with w is a good representation of X.

If one is familiar with the minimal spanning tree[1] in [9], one may readily notice that there are some common points between the induced graph and the minimal spanning tree. (Nevertheless, in fact we had not noticed any similarities between them until the final stage of our work.) Firstly, both were motivated by the limitations of the original Kohonen's algorithm. Secondly, in both cases no pre-specified graph structure is given, and the induced graph and the minimal

[1]It seems more common to call it the minimum spanning tree. However, since in [9] it was called the minimal spanning tree, we had better follow [9].

spanning tree are induced from the mapping w. However, there are also essential differences between them. In [9], the minimal spanning tree was computed using direct distance, and its relation to the input space was only through the mapping w, which was trained by the training patterns. Of course using geodesic distance may reflect more about the structure of the input space, but practically we never measure geodesic distance. The induced graph, on the other hand, is closely related to the Voronoi regions. Even when direct distance is used, whether there is a link is completely determined by whether there is a training pattern nearby. Therefore the induced graph strongly reflects the structure of the input space. In fact it was the original purpose of the induced graph, and the minimal spanning tree was not originally for this purpose. As a result, the induced graph is much better than the minimal spanning tree in reflecting the structure of the input space. For example, we have seen that if the input space is disconnected, the minimal spanning tree is never disconnected but the induced graph would be disconnected according to the input space. Moreover, as we have seen in Chapter 6 the dimension of the input space can be easily extracted from the induced graph, but we do not think that the same is true for the minimal spanning tree.

If we say that the emergence of the induced graph was driven by the maturity of the theory, then the development of the theory for the studies of the dimension was completely driven by the observations in the experiments. The appearance of the induced graph was so natural that almost everyone was ready to accept it, while the observations in the experiments about the dimension were indeed amazing. For example, although Conjecture 1 may not be true for non-linear spaces, it was a sufficient surprise to us. If we do more researches in this area

in future, we should not be surprised to have more and more surprises.

At last we want to point out some possible directions of further researches. We have applied J_1, $\overline{K\text{card}(N_G \backslash \tilde{N}_I)}$ and $\overline{K\text{card}(\tilde{N}_I \backslash N_G)}$ to detect whether Kohonen's algorithm succeeded or not. It is useful especially for high dimensions. We have also tried to make use of the information of local minima or induced graph to develop new algorithms or modify Kohonen's algorithm, but none of them was a significant success. Although the induced graph successfully reflects the structure of the input space, as mentioned in Chapter 5 it does not mean that we do not need a topological order preserving algorithm anymore. On the contrary, from the experimental results of Kohonen's algorithm in high dimensions (e.g. torus in $\mathbf{R}^3$), we find that we really need a better topological order preserving algorithm. Kohonen's algorithm has some limitations as mentioned in Chapter 2. To develop a better topological order preserving algorithm is important but extremely difficult. Another possible direction of further researches is to develop more applications of the induced graph to the studies of the structure of the input space. In this direction there are plenty of opportunities. Even in the determination of dimension, our work has only been a start rather than an end. Our results for non-linear spaces were not good enough, and should have some improvements if we know more about the nature of non-linear spaces. In fact our understanding in linear space is still far from complete. The researches in this area are colourful, and the findings are always non-trivial. As we have a good start, there is no reason to stop our work in this direction.

Bibliography

[1] M. D. Alder, R. Togneri, and Y. Attikiouzel. Dimension of the speech space. *IEE Proceedings I (Communications, Speech and Vision)*, 138(3):207–214, June 1991.

[2] J. D. Farmer. Chaotic attractors of an infinite dimensional system. *Physica D*, 4:366–393, 1982.

[3] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59(8):845–848, August 1987.

[4] J. D. Farmer and J. J. Sidorowich. Exploiting chaos to predict the future and reduce noise. Technical Report LA–UR–88–901, Los Alamos National Laboratory, 1988.

[5] J. D. Farmer and J. J. Sidorowich. Predicting chaotic dynamics. In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic patterns in complex systems*. World Scientific, 1988.

[6] I. Grabec. Self-organization based on the second maximum entropy principle. In *First IEE International Conference on Artificial Neural Networks (Conference Publication Number 313)*, London, October 1989.

[7] I. Grabec. Self-organization of neurons described by the maximum-entropy principle. *Biological Cybernetics*, 63(5):403–409, 1990.

[8] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley, 1991.

[9] J. A. Kangas, T. K. Kohonen, and J. T. Laaksonen. Variants of self-organizing maps. *IEEE Transactions on Neural Networks*, 1(1):93–99, March 1990.

[10] A. Kaufmann. *Graphs, dynamic programming, and finite games.* Academic Press, 1967.

[11] T. Kohonen. *Self-organization and associative memory.* Springer Series in Information Sciences. Springer-Verlag, second edition, 1988.

[12] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[13] E. Kreyszig. *Introductory functional analysis with applications.* John Wiley & Sons, 1978.

[14] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: prediction and system modeling. Technical Report LA–UR–87–2662, Los Alamos National Laboratory, 1987.

[15] R. P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, April 1987.

[16] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, July 1977.

[17] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Physical Review Letters*, 45:712–716, 1980.

[18] L. R. Rabiner and R. W. Schafer. *Digital processing of speech signals.* Prentice-Hall, 1978.

[19] T. D. Sanger. A tree-structured adaptive network for function approximation in high-dimensional spaces. *IEEE Transactions on Neural Networks*, 2(2):285–293, March 1991.

[20] G. F. Simmons. *Introduction to topology and modern analysis.* McGraw-Hill, 1963.

[21] K. Stokbro, D. K. Umberger, and J. A. Hertz. Exploiting neurons with localized receptive fields to learn chaos. *Complex Systems*, 4(6):603–622, December 1990.

[22] F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L.-S. Young, editors, *Dynamical systems and turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80.* Springer-Verlag, Berlin, 1981.

[23] N. Tishby. A dynamical systems approach to speech processing. In *International Conference on Acoustics, Speech and Signal Processing*, 1990.

[24] R. Togneri, M. D. Alder, and Y. Attikiouzel. Dimension and structure of the speech space. *IEE Proceedings I (Communications, Speech and Vision)*, 139(2):123–127, April 1992.