STATISTICAL PATTERN RECOGNITION BASED ON LVQ ARTIFICIAL NEURAL NETWORKS: APPLICATION TO TATA BOX MOTIF

By

Haiyan Wang.

December 2000

STATISTICAL PATTERN RECOGNITION BASED ON LVQ ARTIFICIAL NEURAL NETWORKS: APPLICATION TO TATA BOX MOTIF

By

Haiyan Wang.

Dissertation submitted in compliance with the requirements for Masters Degree in Technology in the Department of Electrical Engineering (Light Current) at Technikon Natal.

This dissertation represents my own work

Haiyan Wang

APPROVED FOR FINAL SUBMISSION

15/12/2000

Professor Vladimir B. Bajic, Pr. Eng., D. Eng. Sc. (EE)
Supervisor

Date

December 2000

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor V. B. Bajić for suggesting me the problem I worked on in this study. I am grateful his guidance, patience and support. He has provided more encouragement for me than I could imagined. Without his help and knowledge this study would never have been finished.

I also wish to thank Dr. Kevin Duffy for being co-supervisor, for his help and guidance during the course of this study. Special thanks are due to Catherine Radakishun who has helped me concentrated on my thesis. Special thanks are also due to Linda Seaman who typeset and proof read the manuscript, putting in a lot of hard work and time.

My parents deserves many thanks for their encouragement and supporting me. Many thanks and apologies to my little son for many nights and weekends devoting to this thesis rather than being with him.

Finally, I would like to thank my husband for his understanding, inspiration and love.

Abstract

The computational analysis of eukaryotic promoters are among the most important and complex research domains that may contribute to complete gene identification. The current methods for promoter recognition are not sufficiently developed. Eukaryotic promoters contain a number of short motifs that may be used in promoter recognition. Having good computational models for these motifs can be crucial for increased efficiency of promoter recognition programs. This study proposes a combined statistical and LVQ neural network system as a computational model of the $TATA$ box motif of eukaryotic promoters. The methodology used is universal and applicable to any short functional motif in DNA.

The statistical analysis of the core $TATA$ motif hexamer and its neighboring haxamers show strong regularities that can be used in motif recognition. Moreover, the positional distribution of the $TATA$ motif in terms of its distance from the transcription start site is very regular and is used in the statistical modeling. Furthermore, the matching score of the position weight matrix for the motif was used as a part of the model. Based on these statistical properties, a novel LVQ classifier for $TATA$ motif recognition is developed. The characteristics of the method are that the genetic algorithm was used for finding good initial weights of the LVQ system, while fine tuning of two LVQ networks was done by the lvq2 algorithm. The final computational model is developed for a recognition level of 67.8% correct recognition on the test set with less than 1% false recognition. This model is evaluated in the task of promoter recognition on an independent test set. The results in promoter recognition outperform three other promoter recognition programs. It is shown that the recognition of promoters based on the recognition of the $TATA$ motifs using this new model is superior to the recognition based on the currently used position weight matrix description of this motif.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, developments of broad fields of science and technology have urged the formation and progress of an new multidisciplinary scientific field named bioinformatics. Bioinformatics is a branch of science that uses biological data stored in computer databases and with the aid of computer science, attempts to derive new biological knowledge [1]. One of the important goals of bioinformatics is to contribute to the reduction of the number of necessary laboratory experiments. It is achieved by analyzing biological data by different computer algorithms, and making predictions of where the most relevant data is located, either on the DNA or RNA sequences, or in the protein sequences. Bioinformatics relates to many research fields such as molecular biology, biochemistry, genetics, biotechnology, parallel computing, internet technology, artificial intelligence, information systems, etc.

One of the major tasks in bioinformatics is mapping of the whole genome of model organisms. In June 2000, it was announced that the human genome is virtually sequenced. This means that most of the human genome content is found and stored in computer databases. This aspect of bioinformatics, aimed at gathering the genetic data is based on physical experiments and provides the raw information that can later be analyzed by

computers.

It should be mentioned that in addition to the collection of raw genetic and other biological data, the core of bioinformatics relates to the computerized analysis of biological data contained in biological databases, data integration, and utilization of information contained in databases. Comprehensibly, these are regarded as the essence of bioinformatics. An important aspect of computer analysis of DNA data is the computational detection of genes embedded in long DNA strings and deciphering their function. A problem related to complete gene recognition is promoter recognition which more precisely determines the starting endpoint of genes ([2], [32], [33], [43]).

1.1.1 Promoter recognition

Gene identification problems relate in most cases to identification of protein coding genes in DNA and its function ([17], [18], [24], [31], [42], [81]). Although it has been announced that the human genome has been basically sequenced, our understanding of how regulatory information is encoded by a DNA sequence and how it becomes functional is still very fragmentary. Thus, both experimental and computational techniques contribute to gene recognition. Experimental techniques provide more certain information, but it are too slow and too expensive to perform the huge amount of necessary experiments to locate the genes. Thus, computational methods are becoming one the most important approaches to achieve gene recognition, and also can offer relative concentrated information to reduce necessary experiments.

Gene expression is regulated in numerous ways and at many levels in the biochemistry of the cell. One of the most important points of regulation is generally transcription initiation. The region of DNA before the respective gene in eukaryotes that is involved intimately in this transcription machinery is called a promoter. It is thus possible to use promoter prediction to achieve a more precise location of the start of the genes in eukaryotes. By knowing the location of a promoter one knows at least the approximate start of the transcript, and eventually this may help in finding the beginning of the gene.

Another reason we are interested in searching for a promoter is that a promoter itself plays an important role in gene and is also a part of the DNA mapping. Moreover, finding different structures of promoters may help in clustering numerous genes.

In fact, the relative abundance of information contained in a promoter makes it a focus of much research, (see [7], [32], [87]). Current knowledge of promoter structure and its activities relies on extensive experimental work and the information obtained in this way allows at a least partial background for the establishment of rules that can be used for promoter recognition.

Our interest is in eukaryotic promoters. These are far more complex than the prokaryotic ones, and possess very individual structure specialized for different conditions of gene expression and different timings. Estimates are that nearly 70%~80% eukaryotic promoters have the $TATA$-box (see [11], [15], [27], [44], [83], [88], [114], [129]), an important functional motif in promoters. With this in mind one can assert that most of the eukaryotic promoter predictions can benefit heavily by finding good matches to the $TATA$-box or $TATA$-like motifs. The $TATA$-box is a hexamer sequence with a consensus given by $TATAAA$, though it is more accurate to describe it with the Position Weight Matrix (PWM) (cf. [45], [119], [121], [122], [123]). It should be mentioned that an orientation of a DNA strand is determined by direction from the so-called 5′ end to the 3′ end of the strand. Location of specific motifs in DNA or RNA is normally expressed relative to some 'fixed' position in DNA which in the case of $TATA$-box is the transcription start site (TSS). Then, if a motif is located 20 bp to the 5′ end it is said that it is located upstream of the reference point, or if it is located 20 bp towards the 3′ end it is said it is positioned downstream of the reference point. 'bp' stands for base pairs on the double stranded DNA. The $TATA$-box is found in a majority of protein-coding genes in eukaryotes centered about 25 to 30 bp upstream of the TSS, and this information can be employed to enhance promoter recognition.

A number of computer programs exist which aid in promoter identification ([2], [7], [8], [15], [22], [26], [32], [36], [49], [58], [62], [67], [71], [72], [74], [75], [84], [85], [93],

[95], [101], [102], [106], [105], [110], [116], [120], [130], [132]). One of the frequently used characterizations of $TATA$-motif in promoter recognition programs is its PWM description ([15], [93], [116]). But PWMs of other promoter motifs are also used [20], [22], [37], [38], [62], [67], [74], [92], [93], [94], [97], [100], [106], [105], [116]. Also, a number of methods for promoter recognition are based on artificial neural networks (ANNs) ([5], [7], [8], [49], [62], [71], [72], [74], [75], [104], [105], [106]). However, despite significant progress made during recent years, and evaluation studies of publicly available computer programs in 1997 [32], indicated that promoter recognition programs are insufficiently developed and that their performance is far from satisfactory. For the independent test set of Fickett and Hatzigeorgiou [32], only $13\% - 54\%$ of the true promoters were found. False predictions ranged from as bad as one false prediction for every 460 bp to up to one in over 5000 bp.

With this in mind, it is possible to formulate the research goals of this project.

1.2 Research goals

The general research goal is directed toward the improvement of promoter recognition. As we mentioned above, the prediction accuracy of current computer programs aimed at promoter identification is not so satisfactory. Though our current knowledge of promoter structure and functionality is still not complete, it is clear that there are grounds to employ the raw information contained in the data sequenced by experimental methods and develop different learning algorithms and neural networks to attempt recognition of eukaryotic promoters.

In this study we will attempt to:

1. find some regularity for the DNA segments around the $TATA$-box and the distribution of the putative $TATA$-motifs based on the PWM scores;

2. propose a system for promoter recognition combining the learning vector quantization neural networks (shortly LVQ) with statistical analysis based on 1, and the

analysis of the effects of different parameter values to recognition quality;

3. apply genetic algorithms to optimize the initial weights of LVQ in order to improve prediction accuracy.

Thus the chapters are organized so as to describe the system structures and their performances. Chapter 2 covers the biological background necessary for the development of algorithms, demonstrates some statistical features of the neighboring DNA segments around the $TATA$-box and the distribution of $TATA$-boxes. Chapter 3 describes how to use the LVQ network to recognize promoters based on recognition of the $TATA$-box motif and discusses in detail the effect of parameter changes of LVQ against promoter data. A genetic algorithm is introduced in Chapter 4 to overcome the effect of initial weight determination in order to accelerate convergency and improve recognition accuracy. A novel $GLVQ$ classifier system with multistage processing is developed. Finally, Chapter 5 presents conclusions.

Chapter 2

Statistical Analysis of Putative TATA Motifs

In this chapter a basic model of $TATA$ motifs is developed as a combination of PWM, some statistical properties of the motif and its immediate neighboring regions, and position information. This model will be used in Chapter 4 to further support promoter recognition using LVQ networks. We first proceed with the necessary molecular biology background. DNA and eukaryotic promoter structure are briefly introduced in Section 2.1. In Section 2.2, the basic problem we are dealing with, the data source, etc. are described. Section 2.3 discusses some statistical and biological regularities related to the $TATA$ motif, as well as position distribution information based on PWM scores of the motif. Finally a simple program is developed to test the efficiency of the basic statistical model of the $TATA$ motif.

2.1 Molecular Biology Background

In this section, we present some necessary molecular biology knowledge required for understanding the algorithms that we will develop for promoter recognition.

Information governing the characteristics of every organism is stored in its DNA

(deoxyribonucleic acid). DNA molecules contain specific regions, genes, that via the biochemical activities of the cell, produce different proteins. Protein molecules in turn control cellular chemistry and contribute to cell structure. The whole human organism is a collection of trillions of cells working together, with each cell having its own identity and function which is completely controlled by the DNA. Thus, DNA influences all physical characteristics for every living organism on earth.

2.1.1 DNA & RNA

In every living organism DNA is the basic information-contained unit. The normal DNA structure is a right-handed double helix, which was first discovered by James D. Watson and Francis Crick in 1953 (Fig. 2-1). The DNA molecule actually consists of two strands twisted about each other in the shape of a spiral staircase (double helix). Each spiraling strand consists of the fundamental DNA building blocks made up of nitrogenous bases called nucleotides. A nucleotide consists of three parts: (1) a deoxyribose sugar, (2) a phosphate group, and (3) a nitrogen-containing pyrimidine or purine **base**. Each deoxyribose sugar unit contains five carbon atoms joined in a ring structure with an oxygen atom. The carbon atoms of the deoxyribose sugar are designated by numbering them sequentially from one to five. The first carbon atom, the 1′ carbon, is by definition the carbon atom covalently attached to one of the organic bases. Phosphate groups are attached to the third (3′) and fifth (5′) carbon atoms.

There are four different types of nucleotides and their sequential ordering in DNA is the most responsible for organisms inheritable characteristics. These nucleotides are adenine (A), thymine (T), cytosine (C) and guanine (G). The bases from each strand bind weakly to each other, holding the molecule together. Each type of base will only bind with another specific type and the only possible pairing is A-T and C-G. Thus, the two strands are complementary and information contained in one is also contained in the

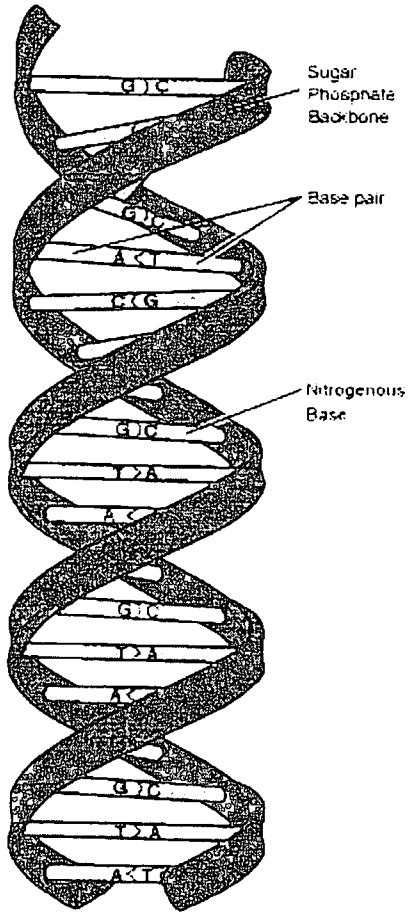

Figure 2-1: A DNA molecule consists of two strands that wrap around each other to resemble a twisted ladder (taken from http://www.accessexcellence.org/AB/GG/dna.html)

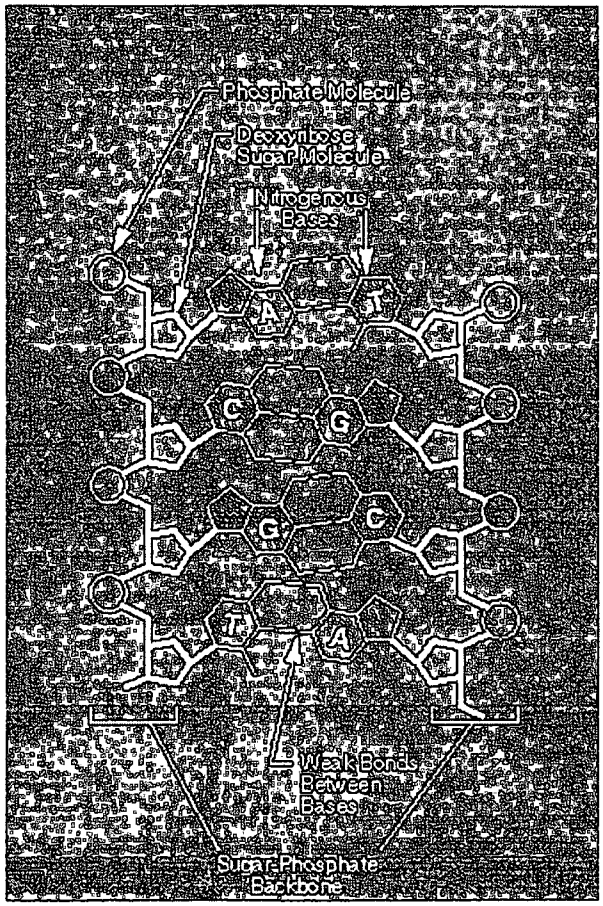

Figure 2-2: The DNA structure (taken from http://esg-www.edu:8001/esgbio/lm/nucleicacids/dna.html)

other. Two bases that bind together are referred to as a **base pairs (bp)**. A and T are connected by two hydrogen bonds. G and C are connected by three hydrogen bonds (Fig. 2-2).

Another nucleic acid called ribonucleic acid (RNA) is also intimately involved in the biochemistry of the cell. The chemical composition of RNA is similar to DNA, and the only difference is that in RNA, thymine (T) is replaced by uracil (U). In principle, RNA may be double stranded, but most cellular RNAs consists of a single strand. The major role of RNA is related to protein synthesis. This requires three classes of RNA: messenger RNA (mRNA), transfer RNA (tRNA) and ribosomal RNA (rRNA). Other classes of RNA include ribosomes and small RNA molecules.

2.1.2 Genes and Gene Expression

Genes are numerous 'coding units' in a DNA molecule that have specific functions and contain information necessary for making specific products, which in most cases are proteins. One can define a gene more precisely by considering features important for converting information from DNA into protein. Information in an eukaryotic gene is arranged in sections called **exons**, which are separated by other sections called **introns**. Only exons contain the information code for protein synthesis. Normally, three consecutive nucleotides in exons called **codons** contain information for producing an amino acid. In total 20 types of amino acids can be produced in a cell. The final product, the protein, consists of chains of amino acids. In the total DNA of eukaryotes, the coding region accounts for only about 3%–5%. Figure 2-3 shows the general organization of the eukaryotic DNA sequence, which consists of genes, pseudogenes and extragenic region.

Cells make thousands of different proteins. Gene expression is the process by which the information in DNA is converted to protein. In this process information is first transcribed into RNA. The 4-based nucleotide code is then translated from RNA to the 20 types of amino acids which make proteins. There are several steps as shown in Fig.

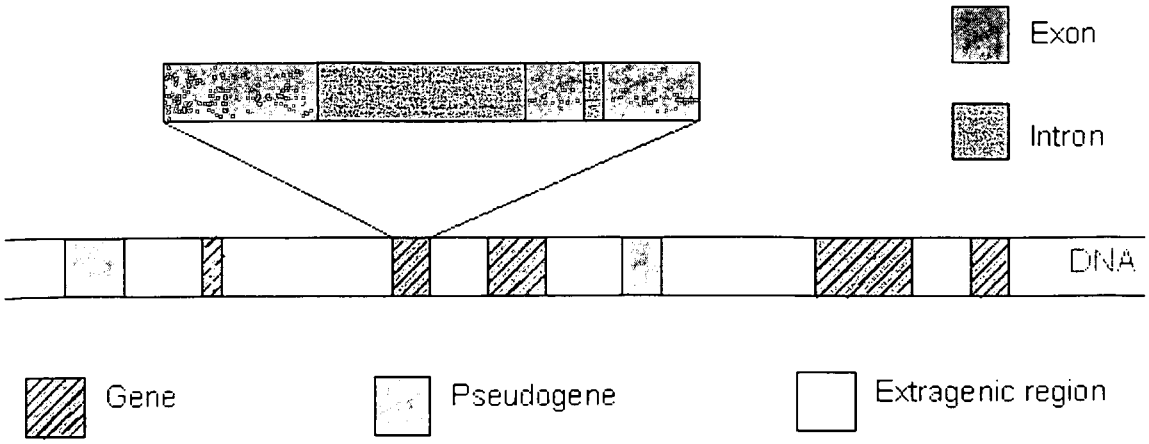

Figure 2-3: General organization of the DNA sequence

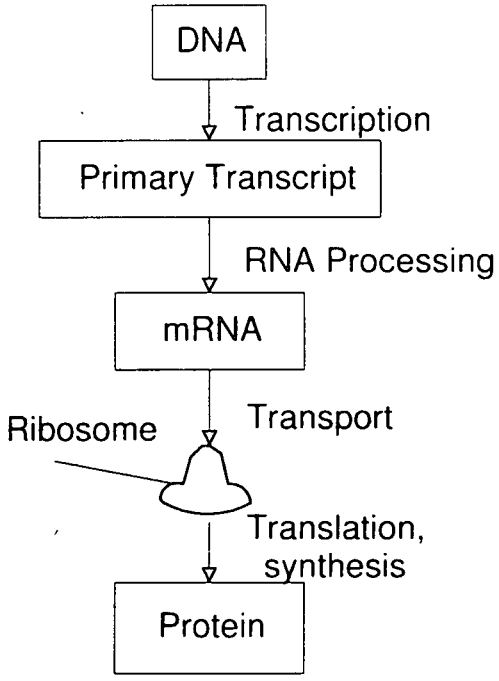

Figure 2-4: Process of conversion of information in DNA to protein

2-4 (note that this process is slightly different for prokaryotic organisms which do not have the second step) that lead to the final protein product:

Step 1. Transcription. Transcription is the first step in protein synthesis. RNA nucleotides are added to the DNA and they form a RNA strand, the so-called primary transcript. Transcription begins when a specific enzyme, RNA polymerase, recognizes and binds to the so-called preinitiation transcription complex that is formed on the DNA. This preinitiation complex is formed just before the beginning of the gene and it is located in a region called a **promoter.** Prokaryotic polymerases can recognize the promoter and bind to it directly (they do not require the preinitiation complex), but eukaryotic polymerases have to rely on other proteins called transcription factors (TFs) that participate in formation of the preinitiation transcription complex. After binding, the interaction of the RNA polymerase and DNA causes the DNA strands to separate over a short distance, and the polymerase molecule moves into the gene. It begins adding RNA nucleotides to form a new chain which is based on the sequence of the DNA template strand. Transcription continues until RNA polymerase reaches a termination site on the DNA strand, and then the new RNA sequence is released. This finishes the formation of the primary transcript. In prokaryotic organisms, the new RNA sequence is the so-called messenger RNA (mRNA), while in the eukaryotic organisms the result of transcription is the so-called pre-mRNA.

Step 2. RNA processing. This step is characteristic only for eukaryotic organisms and involves alterations of the primary transcript to generate a functional mRNA. In a process known as RNA splicing the noncoding regions, introns, are cut out by splicesomes and only the functional coding regions, exons, are left in the mRNA molecule.

Step 3. Translation and protein synthesis. mRNA binds to subcellular structures called **ribosome.** Then, by means of the transfer RNAs (tRNA) information in

mRNA translates to amino acids. During protein synthesis, the tRNA moves amino acids into the proper position along mRNA, and the a protein chain is formed. The process stops when a specific codon called the stop codon is reached.

In simplified terms, the above process can be described as

DNA -> RNA -> Protein

2.1.3 Promoters

As mentioned above in the process of protein synthesis, a promoter plays an important role in the initiation of the transcription process. Promoters are regions to which different TFs bind to form the preinitiation transcription complex that enables RNA polymerase to position properly on the correct DNA strand and points it in the right direction, so that the transcription process can start from the correct position. Promoters generally indicate and contain the starting point of transcription (transcription start site) (TSS). and they regulate the rate of initiation of transcription.

A promoter in eukaryotes is a region located at or near the TSS. The promoters of eukaryotes may comprise different subregions such as $TATA$-box, $CCAAT$-box, initiator element (Inr), GC-box etc. These subregions are not always present and they are not located at the same distances with respect to the TSS. Also, they may appear in different combinations in different promoters. Thus it is difficult to create a unique model of general eukaryotic promoters for a larger group of promoters, because the structure of eukaryotic promoters are very individual to the promoter. This fact is one of the reasons why presently there is no adequate computer tool to accurately detect different types of eukaryotic promoters in a large-scale search of DNA databases. A possible promoter-gene structure in eukaryotes is depicted in Fig.2-5. Also, a possible structure of promoter region is shown in the lower figure in Fig. 2-5. The boundaries for the $TATA$-box element with respect to the TSS are also given.

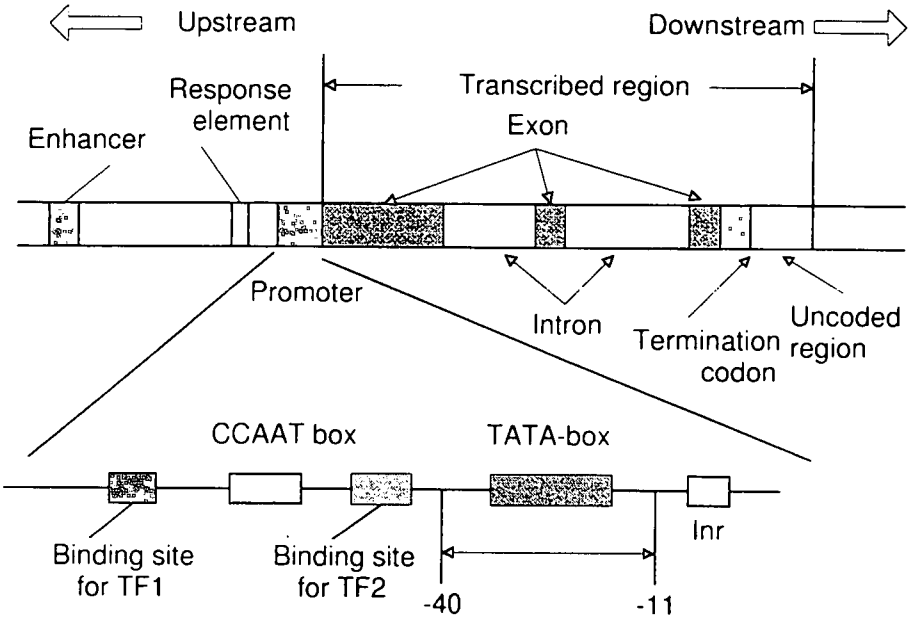

Figure 2-5: Possible promoter-gene structure for mRNA eukaryotic gene. The enhancer region in the upper figure has similar functionallity as the promoter region and it always cooperate with the promoter in transcription initiation. However, it is usually distances very far from the TSS. The lower figure shows possible structure of an eukaryotic promoter that contains different TF binding sites. The indicated boundaries of the $TATA$-box elements are given only for computational purposes.

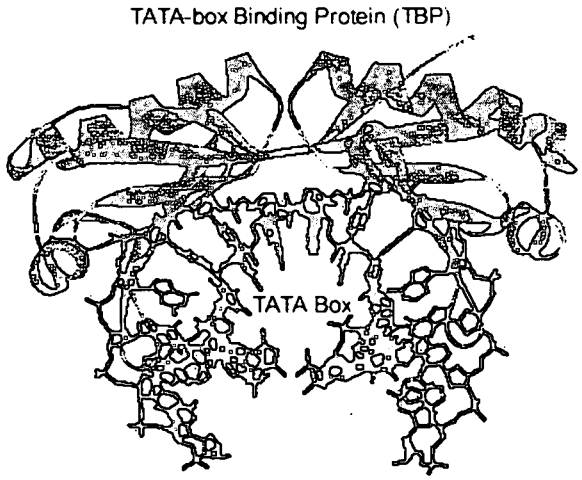

Figure 2-6: TATA-box binding protein - graphical presentation (taken from www.biochem.ucl.ac.uk/bsm/xtal).

Eukaryotic genes are transcribed by three classes of RNA polymerases: I, II and III. Our attention focuses on the RNA polymerase II (Pol II), which is involved in the transcription of all protein coding genes.

Many eukaryotic Pol II promoters have some specific subregions that have reasonably high consensus. The most common promoter element is the $TATA$ box, and for computational purposes we can consider it located at -40 bp to -11 bp upstream of TSS. It is a hexamer rich with thymine (T) and adenine (A). The $TATA$ box together with the initiator belongs to the "core promoter elements". The protein which interacts with the $TATA$ box is known as the $TATA$-box binding protein (TBP), is shown in Fig. 2-6.

The $TATA$ box exists in nearly $70\% - 80\%$ of vertebrate Pol II promoters, and it is relatively characteristic. So finding the $TATA$ box or $TATA$ -like components could be an important ingredient of the methods for promoter recognition. It also helps to detect more precisely the location of TSS since its distance from the TSS is very regular.

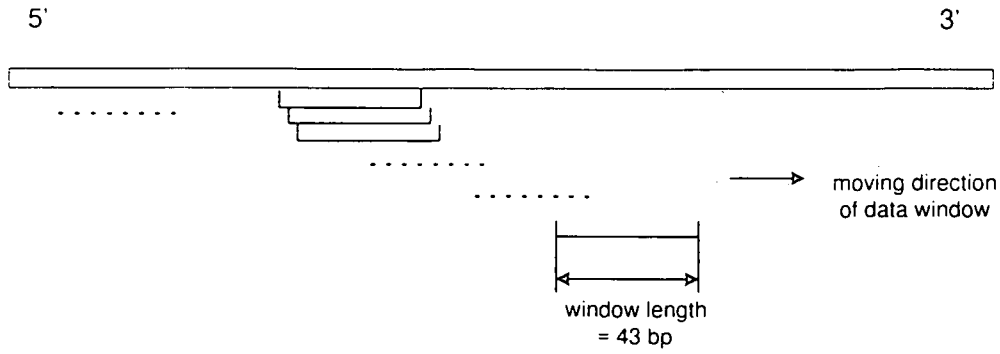

Figure 2-7: A window with the length of 43 positions slides along the sequence to form the input data

2.2 Necessary Background for Promoter Recognition

2.2.1 Introduction

Suppose we have a long DNA sequence which contains some promoters. The goal is to locate these promoters possibly without any false detection. We set a data window to slide along the sequence from its 5′ end toward the 3′ end as shown in Fig. 2-7. Our task is try to find the $TATA$-box in the sequence within the window, then the promoter or TSS can be considered recognized.

The reasons for using the length of window as 43 positions are as follows. Our analysis is based on the consideration of the $TATA$ box hexamer and two neighboring hexamers, one before the $TATA$ box and another after it. For this we need to analyze a segment 18 nucleotides in length. However, we also use the PWM description of the $TATA$ box [15], that uses a window length of 15 nucleotides, where the $TATA$ box hexamer starts at position 2. The union of these two windows gives a new window length of 20 nucleotides, where the $TATA$ box hexamer will start at position 7. Now, the distribution of the $TATA$ box hexamer expressed as the location of its 5′ end nucleotide can be, for computational purposes, considered to be from -39 up to -16 relative to the TSS. Consequently, we need to analyze segments from -45 up to -3 relative to the TSS in order to accomodate

all possible locations of the 20 nucleotide long window in the analysis of location of the $TATA$ box hexamer. When the window of 20 bp slides along the segment of -45 to -3, the location of a hypothetical $TATA$ hexamer will slide from position -39 to position -16 relative to the TSS.

2.2.2 Data Source

To build the respective model of the $TATA$ box motif we need some relatively accurate data. For core promoter elements the best data source is considered to be the Eukaryotic Promoter Database (EPD) ([16], [89]) which contains a collection of annotated experimentally mapped TSSs and surrounding sequences. EPD is structured in a way that facilitates dynamic extraction of biologically meaningful promoter subsets for comparative sequence analysis. It can be accessed at http://www.epd.isb-sib.ch.

The promoter data used in this study is taken from the EPD. We extracted 878 vertebrate promoter sequences that are -45 to -3 relative to the TSS. They are divided into training set $pAtr$ and test set $pAtst$. The training set has 640 randomly selected vertebrate promoters, while the test set has 238 promoter sequences. The non-promoter sequences were taken from the exon and intron regions of simple vertebrate genes (genes without multiple TSSs and without multiple splicing) from the GenBank database. These were divided into non-overlapping segments of 43 bp. From the set of exon sequences we randomly selected 8000 sequences and this set is denoted as S_{cds}. Analogously, from the set of intron sequences, another 8000 sequences are randomly selected to form the set S_{int}. We used the whole S_{cds} set as the negative training set, and the S_{int} set as a negative test set.

2.2.3 Numerical Representation of a Sequence

The DNA sequence is a sequence of 4 different bases denoted by A, T, C and G. For computational analysis it is frequently convenient to represented a sequence by a binary code if no biological or physical properties of nucleotides are taken into consideration.

Base		Code
A		1000
C		0100
G		0010
T		0001

Table 2.1: Binary code for DNA sequence representation

Base		EIIP
A		0.1260
C		0.1340
G		0.0806
T		0.1335

Table 2.2: EIIP used to represent DNA

The code is given in Table 2.1

These codes have the same Hamming distance between any two-nucleotide representation, which is considered desirable for not contributing to biased learning [3]. This code representation has been widely used in numerous ANN based DNA and protein analyses.

Another representation is also successfully used in promoter recognition algorithms [7], which is based on the so-called electron-ion interaction potential (EIIP) [125], [126]. It incorporates some of the physical properties of the nucleotides rather than merely representing them. Our analysis is based on EIIP. The EIIP values are given in Table 2.2.

2.2.4 Recognition Quality

There are several measures that can be used to express the quality of recognition (see [6]). The four basic measures called true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are frequently utilized to partially express the recognition

quality. We will consider a promoter as correctly recognized if the TSS is located within some pre-specified bound. The bounds in this study are taken as -200 and $+100$ relative to the actual TSS location. Thus, if a TSS is predicted to be within the $[-200, +100]$ relative to the actual TSS counts as correctly recognized. Otherwise it counts as falsely recognized. Here in order to avoid ambiguities we define TP, TN, FP and FN in the context of promoter recognition:

$$\begin{aligned} TP &= \%\, of\ correctly\ predicted\ promoters \\ FP &= \%\, of\ predicted\ promoters\ in\ nonpromoters \end{aligned} \tag{2.1}$$

The remaining part of promoters that are not predicted by the program designates false negative (FN). They are simply represented as:

$$FN = all\ promoters - TP \tag{2.2}$$

Also, all other non-promoter locations which are correctly treated by the predictor program are called true negatives (TN):

$$TN = all\ nonpromoters - FP \tag{2.3}$$

The higher the TP is and the lower FP is, the better the recognition result. It is however difficult to make a comparison of the recognition quality based solely on the TP, FP, TN and FN. It is more convenient if the quality of prediction is based on a single measure. One such good measure is given in [6] and is called averaged score measure (ASM). We will use for the preliminary analysis, the so-called correlation coefficient (CC) which is simple for calculation, but for the final results we will use the ASM. The CC is

defined as

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}, \quad -1 \leq CC \leq 1 \tag{2.4}$$

The closer the value of CC is to 1, the better the overall predictor performance. When it reaches 1 it means a completely correct classification is available; when it is zero it means arbitrary classification, and when all the classifications are wrong, the CC value is equal to -1. This measure is very popular in bioinformatics.

2.3 Statistical Analysis

A good model of $TATA$-box motifs is derived from 502 unrelated eukaryotic promoters in a form of a PWM of length 15 nucleotides [15]. We will use this model as a part of our modeling of $TATA$ motifs. Additionally, in order to obtain more information of the motif, we will use some features of the hexamers neighboring the core $TATA$ hexamer. These will be combined with the positional information of the hypothetical $TATA$ motifs and will serve as input data for a neural network based model of this motif.

2.3.1 PWM

Modeling biological signals is an important part of recognition of important regions in biological sequences. One of the most influential approaches to modeling biological signals is based on the PWM introduced by Staden [119]. PWM is a motif descriptor. It attempts to capture the intrinsic variability characteristics of sequence patterns. It is usually derived from a set of aligned sequences that are functionally related. In the case of DNA sequences, a PWM is obtained from the base-frequency matrices as probabilities of a given nucleotide occurring at a given position in a functional site.

	A	C	G	T
-3	15.6812	37.2751	39.0746	7.9692
-2	4.1131	11.8252	4.6272	79.4344
-1	90.4884	0	0.5141	8.9974
0	0.7712	2.5707	0.5141	96.1440
1	91.0026	0	1.2853	7.7121
2	68.8946	0	0	31.1054
3	92.5450	0.7712	5.1414	1.5424
4	57.0694	0.5141	11.3111	31.1054
5	39.8458	11.3111	40.3599	8.4833
6	14.3959	34.7044	38.5604	12.3393
7	21.3368	37.7892	32.9049	7.9692
8	21.0797	32.6478	32.9049	13.3676
9	21.0797	30.3342	32.9049	15.6812
10	17.4807	27.5064	35.7326	19.2802
11	19.7943	25.9640	35.9897	18.2519

Table 2.3: PWM for $TATA$ box according to Bucher (1990)

PWMs are usually used when there is enough information that enables building PWMs accurately. Bucher [15] generalized the basic PWM algorithm and derived PWM models for four promoter elements: $TATA$ box, *cap* site, $CCAAT$ box, and GC box. Bucher's algorithm is conventional in that PWMs are calculated from base frequencies of nucleotides. However, the crucial difference to the other PWM algorithms is that the use of a quantitative measure of local over-representation as an optimization criterion. This is used in order to estimate some parameters that cannot be derived directly from the sequence data such as the cut-off matching score to the PWM, and the width and location of the preferred region of motif occurrence, which are expressed in a mathematical formula. The algorithm is of a synthetic nature and integrates several basic concepts and techniques of nucleotide sequence analysis. The details on Bucher's algorithm can be found in [15].

The PWM of the $TATA$-box is given in Table 2.3, in which position 0 refers to the

second T in the $TATA$-box motif; the columns correspond to the four bases of DNA, while the rows represent the positions of the nucleotides in a DNA sequence. When evaluating a sequence for the presence of $TATA$-like motifs, the query sequence is scanned for the presence of potential sites. Usually a window the length of the PWM is run along the sequence, and the coefficients from the matrix corresponding to each nucleotide in each position on the window are summed. In the $TATA$-box case, the length of the window is normally defined to be 15 nucleotides in length, the length of the PWM. The matching score for a sequence in the examined window is given by:

$$x = \sum_{i=1}^{15} \omega_{b_i,i} \tag{2.5}$$

where b_i is the i-th base of the sequence in a window and $\omega_{b_i,i}$ is the weight of base b_i at the i-th position of the motif.

Let τ represent a selected threshold. If the matching score x is high enough, above a given threshold (which is usually experimentally or statistically determined), say, $x > \tau$, then the sequence is considered to be a matched one and we can regard it as one that corresponds to a hypothetical $TATA$ motif. In our case, the basic data window used to scan along a DNA sequence is 43 nucleotides in length, and within that window another window of length 15 nucleotides slides starting from position 6 (this corresponds to -40 relative to TSS) up to position 30 (which corresponds to -17 nucleotides relative to the TSS). In this process, only the maximum value of all the matching scores in a larger window are compared with the threshold to find out whether it can be considered as the matched one. Other window positions are neglected.

In the case of promoter recognition, PWMs have several advantages over the use of consensus sequences and specific binding sites. PWMs capture more information than a consensus sequence, and have a sound foundation in both statistics (representing likelihood ratios) and thermodynamics (representing binding energies) [100]. However,

in practice, we need a sufficiently large data set to produce a quality PWM. This is frequently a big problem. Fortunately for the $TATA$ box motif the available data are sufficient for a good PWM formulation.

2.3.2 Some statistical representation of $TATA$-box motifs

If a functional $TATA$-box motif exists, one can assume that it will be somehow well separated from its local environment in order to allow more efficient recognition by the TBP. For this reason, we may expect that the segments immediately before and after the core $TATA$ motif will have some different properties that will enhance recognition of the core motif. This also means that we expect that properties of segments around functional motifs will be different from the properties of similar segments around non-functional motifs, such as around pseudo-$TATA$ motifs in non-promoter sequences.

With this idea in mind we focus our attention on the distribution of the bases A, C, G, and T. We use the EIIP values to represent the four bases. To describe the potential feature of the segments around the $TATA$-box, let S_1 represent the neighboring hexamer before the $TATA$-box, let S_2 indicate the $TATA$-box itself, and S_3 the hexamer after the $TATA$-box. The terms 'before' and 'after' are used relative to the direction of the DNA strand from 5′ towards the 3′ end. Let AV_1, AV_2, and AV_3 denote the average EIIP values of the hexamers S_1, S_2, and S_3, respectively.

Let e_j, $j = -40, -39, ..., -3$, represents the EIIP value of each nucleotides in a sequence. Let us assume that the beginning of a $TATA$-box hexamer is in the position i. Then AV_1, AV_2 and AV_3 are given by

$$AV_1 = \frac{1}{6}\sum_{j=1}^{6} e_{i-j},\ AV_2 = \frac{1}{6}\sum_{j=0}^{5} e_{i+j},\ AV_3 = \frac{1}{6}\sum_{j=6}^{11} e_{i+j} \tag{2.6}$$

Initially, the sequences in promoter training set $pAtr$ containing hypothetical $TATA$-box motif are selected by means of calculating the PWM matching score according to

(2.5), when the threshold is selected as $\tau = 0.45$ which allows about 75% (in total 475) sequences to be regarded as containing the $TATA$ motif. Then these sequences are sorted into different groups according to the start position of the $TATA$ motif hexamer. We assume that the 3′ ends of $TATA$ hexamers distribute from $-40\,bp$ to $-11\,bp$, so its starting point (5′ end) is from $-39\,bp$ to $-16\,bp$. Thus, by means of the PWM matching score we determine the positions of hypothetical $TATA$ motifs and we can determine their distributions according to the location of the 5′ end of the motif relative to the TSS. Hence, in total, there are 24 possible positions. Let G_i, $i = -39, -38, ..., -16$, denotes a group of found $TATA$ motifs. The sequences in G_i contain the $TATA$ motifs that all start at position i and found by the PWM matching score in $pAtr$. The number of the sequences in a different G_i are not the same. The total number of predicted $TATA$ motifs N_{tata} is given by

$$\sum_{i=-39}^{-16} N(G_i) = N_{tata} \tag{2.7}$$

where $N(G_i)$ is the number of sequences in G_i. In our case, $N_{tata} = 475$. The distribution of the $TATA$ motifs found in $pAtr$ is shown in Fig. 2-8.

The lower axis represents distance from the TSS. It can be seen that distribution of $TATA$ motifs in $pAtr$ concentrate on -30. For each position i, we have a set of sequences G_i.

The next step is to calculate the average EIIP values AV_1, AV_2, and AV_3, of the three hexamers S_1, S_2, and S_3, for each sequence in G_i. Figure 2-9 depicts AV_1, AV_2, and AV_3 of 87 sequences that belong toG_{-30}. In what follows we will refer to S_1 as the segment before, and to S_3 as the segment after the $TATA$ motif hexamer.

Although the values of AV_2 are generally above the values of AV_1 and AV_3, there are some overlaps. We would like the values of AV_2 to be well separated from the values of

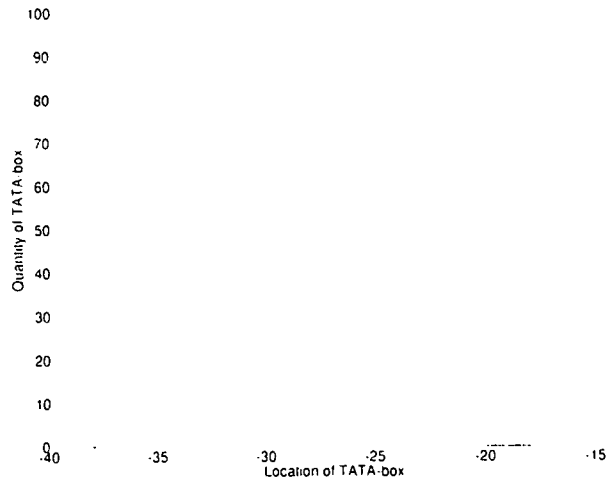

Figure 2-8: Distribution of $TATA$ motifs found in $pAtr$ set.

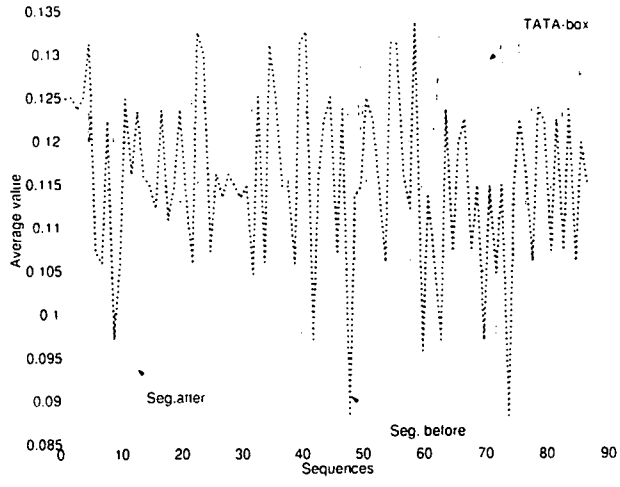

Figure 2-9: The average EIIP value of sequences in G_{-30}.

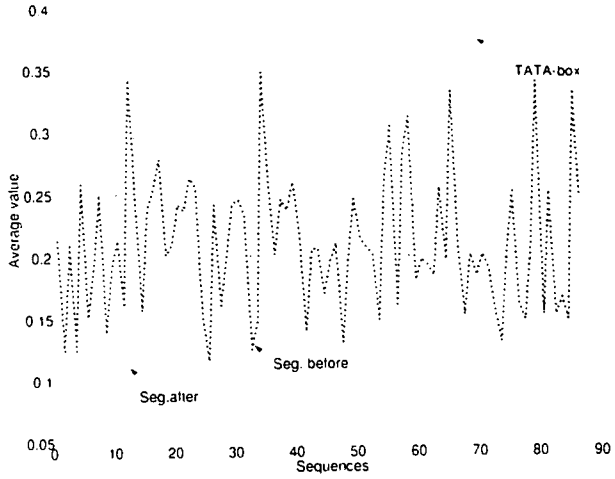

Figure 2-10: Average EIIP values for the three hexamers when the EIIP values for A and T are multiplied by 3.

AV_1 and AV_3. To achieve this we attempt to modify the EIIP values for A and T (those that are the main constituents of the $TATA$ motifs) and multiply them by some factor, expecting that this may provide better separation of the AV_2, AV_1 and AV_3 values. The factors chosen were 3, 5, and 10. The results of the three situations are given in Fig. 2-10, Fig. 2-11 and Fig. 2-12.

As expected, the average values of S_2 increase a lot, and the range of values of AV_1 and AV_3 is relatively compressed. One can notice that the overlapping of AV_1 and AV_3 values with AV_2 is considerably reduced. Therefore, for further analysis we will use the EIIP values of A and T multiplied by 10.

We are now in a position to further analyze the distribution of the values of AV_1, AV_2 and AV_3 and on that basis draw some conclusions. We applied the threshold $\tau = 0.45$ for the PWM matching score to select $TATA$ motifs from the $pAtr$ set, as well as the pseudo-$TATA$ motifs from the S_{cds}. In this way we obtained about 13% (exactly 1040) sequences from S_{cds} that passed this threshold filtering of the matching score.

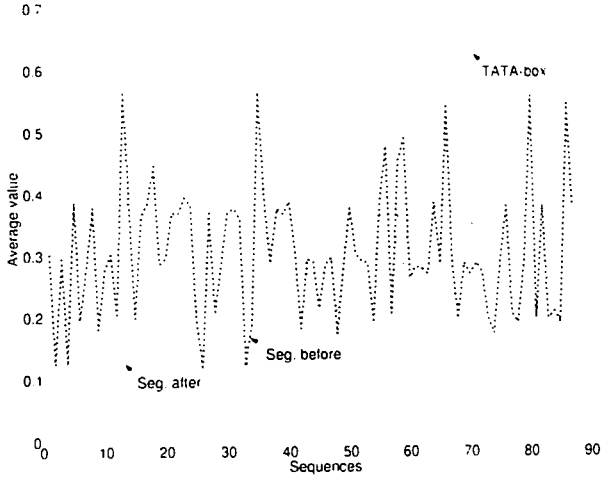

Figure 2-11: Average EIIP values for the three hexamers when the EIIP values for A and T are multiplied by 5.

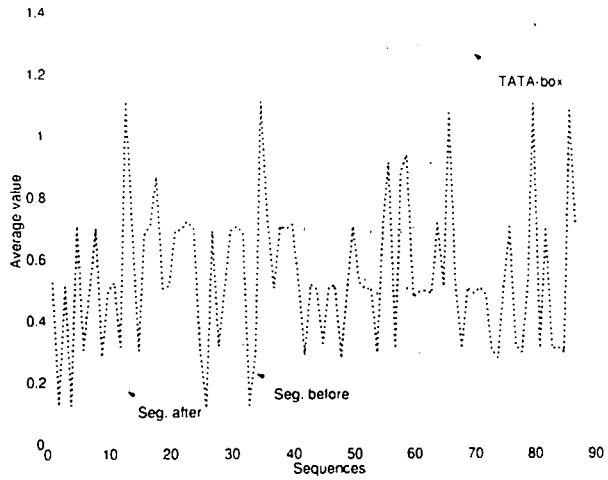

Figure 2-12: Average EIIP values for the three hexamers when the EIIP values for A and T are multiplied by 10.

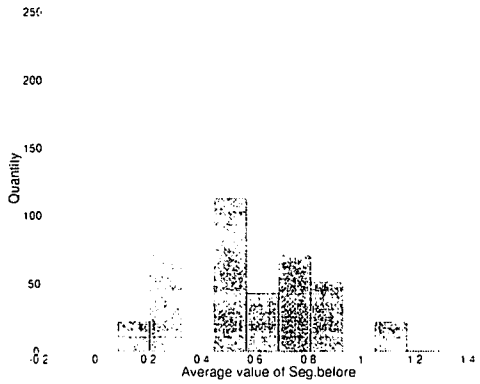

Figure 2-13: Distribution of AV_1 for sequences from $pAtr$.

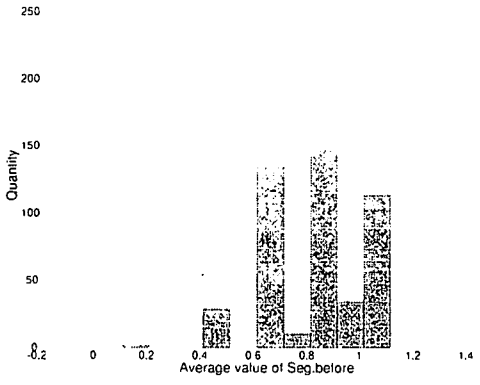

Figure 2-14: Distribution of AV_1 for sequences from S_{cds}.

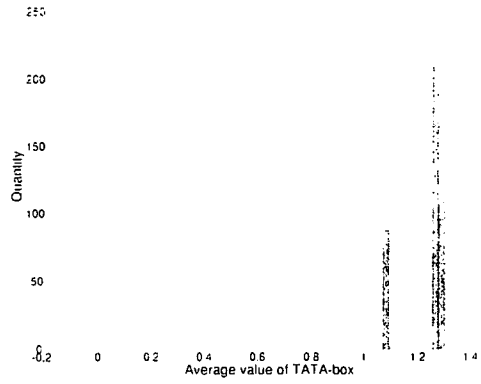

Figure 2-15: Distribution of AV_2 for sequences from $pAtr$.

Several interesting observations could be made based on Figs. 2-13-2-22.

1. There exist some regions of averaged values for modified EIIP where the $pAtr$ data appears but no S_{cds} data does. For example, compare Fig. 2-21 and Fig. 2-22. In the region $AV_2 - AV_3 > 0.8$, there is no data from S_{cds}, while there is some data from $pAtr$. A similar situation also appears in the region $AV_2 - AV_1 > 1$ in Fig. 2-19 and Fig. 2-20; also, the region $AV_3 < 0.5$ in Fig. 2-17 and Fig. 2-18 contains not data from $Scds$ but contains data from $pAtr$.

2. In contrast, S_{cds} data is available in some regions where no $pAtr$ data appears. This can be seen in the region $AV_2 < 1$ in Fig. 2-15 and Fig. 2-16, as well as in the region of values of AV_1 around 1 in Fig. 2-13 and Fig. 2-14.

3. Also, some regions exist which contain a large proportion of data from one group, while having a relatively small proportion of data from another group. For example,

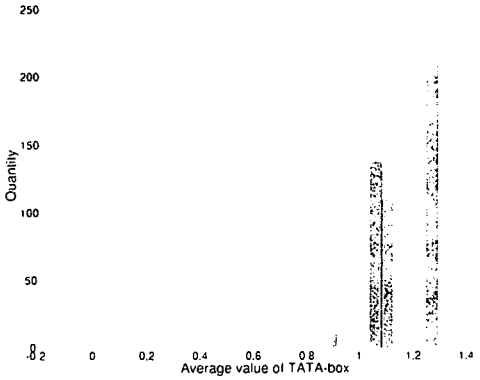

Figure 2-16: Distribution of AV_2 for sequences from S_{sds}.

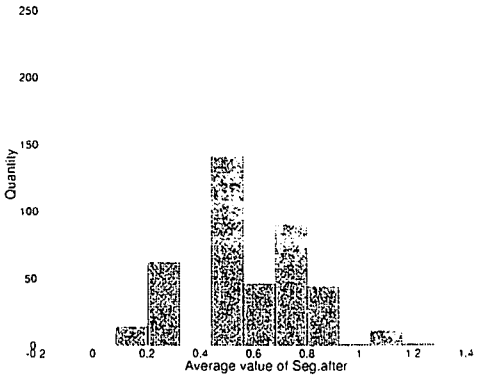

Figure 2-17: Distribution of AV_3 for sequences from $pAtr$.

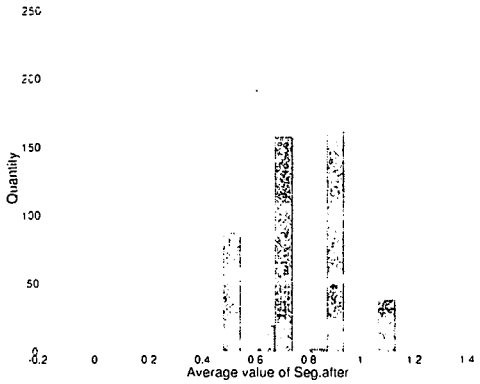

Figure 2-18: Distribution of AV_3 for sequences from S_{cds}.

Figure 2-19: Distribution of the difference $AV_2 - AV_1$ for sequences from $pAtr$.

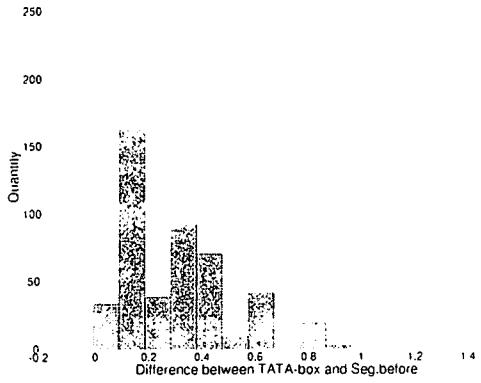

Figure 2-20: Distribution of the difference $AV_2 - AV_1$ for sequences from S_{cds}.

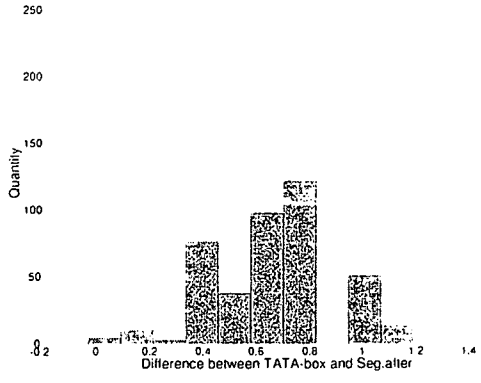

Figure 2-21: Distribution of the difference $AV_2 - AV_3$ for sequences from $pAtr$.

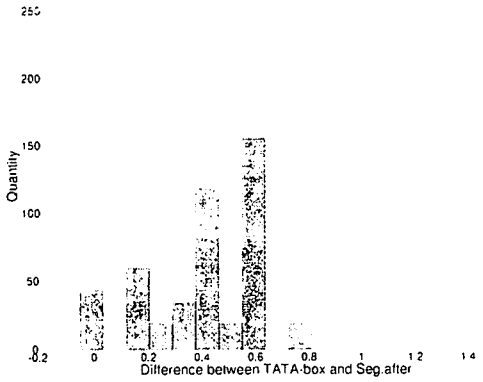

Figure 2-22: Distribution of the difference $AV_2 - AV_3$ for sequences from S_{cds}.

the region $[1.272, 1.4]$ for AV_2 (in Fig. 2-15 and Fig. 2-16), contains about 90.5% of $pAtr$ data, while only 69.3% of S_{cds} data is within the region.

The observations indicate that there are significant statistical differences in the distribution of values AV_1, AV_2 and AV_3, as well as their differences $D_1 = AV_2 - AV_1$ and $D_2 = AV_2 - AV_3$. One can in principle attempt to utilize these statistical regularities and try filtering out sequences that potentially contain a $TATA$ like motif. For illustrative purposes only, some results on filtering $pAtr$, $pAtst$, and S_{cds} data by restricting the values of AV_2, D_1 and D_2 are given in Table 2.4. In this case a window of 18 nucleotides in length slides along the 43 bp long seqence and AV_2, D_1 and D_2 are calculated for each position of the window. When conditions given in Table 2.4 are satisfied the sequence of 43 bp is considered to contain a $TATA$ motif.

We can also combine the value restriction filtering with the PWM matching score in the algorithm that is depicted in Fig. 2-23.

The recognition results using the algorithm from Fig. 2-23 are presented in Table 2.5. The threshold for the matching score of PWM is 0.45. Table 2.5 indicates that we

	Interval	*pAtr* (%)	*pAtst* (%)	S_{cds} (%)
1	$D_1 \in [0.3, 1.3], D_2 \in [0.2, 1.2]$	91.6	89.8	71.5
2	$D_1 \in [0.3, 1.3], D_2 \in [0.2, 1.2]$ $AV_2 \in [0.7, 1.097] \cup [1.272, 1.31]$	86.2	83.6	54
3	$D_1 \in [0.3, 1.3], D_2 \in [0.2, 1.2]$ $AV_2 \in [1.07, 1.097] \cup [1.272, 1.31]$	75.1	71.4	19.0
4	$D_1 \in [0.4, 1.3], D_2 \in [0.4, 1.2]$ $AV_2 \in [1.07, 1.097] \cup [1.272, 1.31]$	64.5	58.8	10.3
5	$D_1 \in [0.4, 1.3], D_2 \in [0.6, 1.2]$ $AV_2 \in [1.07, 1.097] \cup [1.272, 1.31]$	55.0	46.2	3.75
6	$D_1 \in [0.5, 1.3], D_2 \in [0.6, 1.2]$ $AV_2 \in [1.07, 1.097] \cup [1.272, 1.31]$	52.7	41.2	2.9

Table 2.4: Recognition results based on the restriction of different numrical paramters

	Data region		*pAtr* (%)	*pAtst* (%)	S_{cds} (%)	S_{int} (%)
	D_1	D_2				
1	[0.3, 1.3]	[0.39, 1.2]	58.1	45.4	2.6	4.8
2	[0.5, 1.3]	[0.43, 1.2]	43.1	33.2	1	1.5
3	[0.5, 1.3]	[0.58, 1.2]	41.4	31.5	0.51	0.55
4	[0.7, 1.3]	[0.6, 1.2]	22.4	15.1	0.3	0.19
5	[0.95, 1.3]	[−0.1, 1.2]	13.1	6.7	0.025	0
6	[0.97, 1.3]	[−0.1, 1.2]	11.1	5.8	0	0
$AV_2 \in [1.07, 1.097] \cup [1.275, 1.4]$						

Table 2.5: Recognition of promoters by means of recognition of the TATA motif

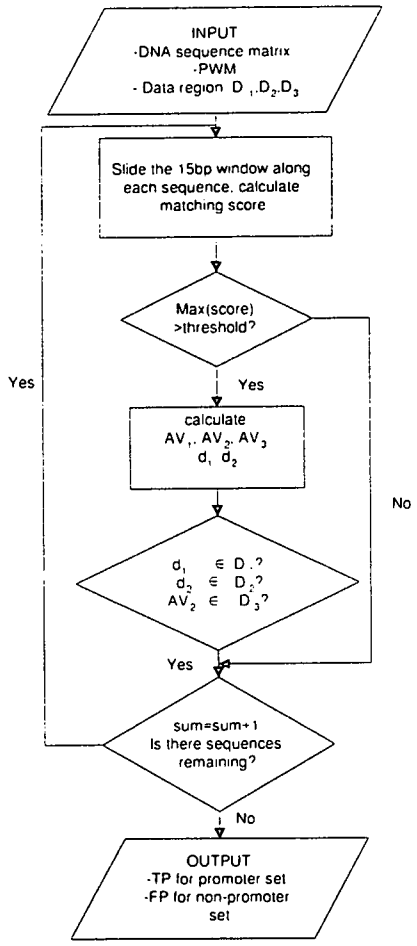

Figure 2-23: Simplified flow-chart of the algorithm for promoter recognition based on the recognition of the $TATA$ motif. The PWM threshold restriction and AV_2, D_1 and D_2 values restriction are used for sequence filtering.

position	AV_1	AV_2	AV_3
$-34bp$	0.11348	0.12712	0.11283
$-33bp$	0.11435	0.12829	0.11362
$-32bp$	0.11312	0.1283	0.11208
$-30bp$	0.11529	0.12817	0.11342
$-29bp$	0.11478	0.12776	0.11198
$-28bp$	0.11319	0.12861	0.11068
$-27bp$	0.12081	0.12831	0.10944

Table 2.6: Different average EIIP values of the 5′ nucleotide of $TATA$ motifs at several positions

obtained $FP = 4.8\%$ on S_{int} and $FP = 2.6\%$ on S_{cds}, while $TP = 58.1\%$ on $pAtr$. By using the PWM directly, we have obtained $TP = 65.26\%$ on $pAtr$, and $FP = 13\%$ on S_{int} and $FP = 7.01\%$ on S_{cds}. The results achieved show some encouraging reduction of FP, although TP is also decreased. However, this property of the $TATA$-box and the segments before and after could be used in more sophisticated systems for promoter recognition.

In addition to this, we found that in a different locations of the 5′ end of the $TATA$ motif, the average EIIP values are not the same. This suggests that if we cluster data according to different locations of the 5′ end of $TATA$ motif, the recognition accuracy may be increased. Table 2.6 shows the average EIIP values in several different positions.

2.3.3 Position Analysis

Figure 2-8 indicates the distribution of the 5′ end of $TATA$ hexamer in the promoter set $pAtr$ which is concentrated around position $-30\,bp$. For a comparison, we examine the distribution of the $TATA$ motifs falsely recognized in the non-promoter set S_{cds}. With the threshold $\tau = 0.45$ which produced 75% of hypothetical $TATA$ motifs in $pAtr$, we found 1040 sequences in S_{cds} that are falsely predicted as $TATA$-box elements. This makes $FP = 13\%$. The distribution of the position of the false predictions is shown in

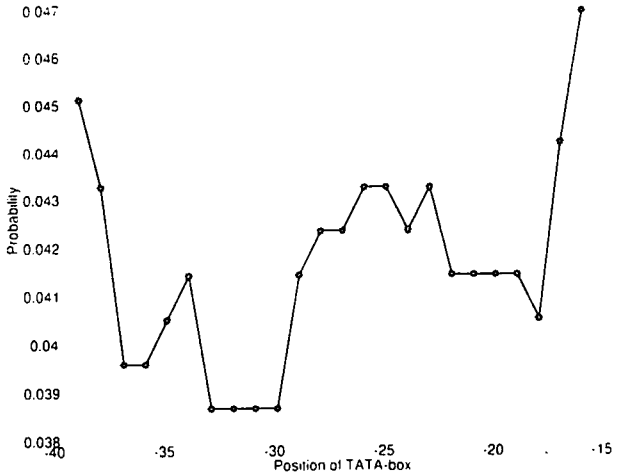

Figure 2-24: Distribution of the 5′ end of $TATA$-box motifs falsely predicted in the S_{cds} set.

Fig. 2-24. In an analogous way, we found the distribution of falsely recognized $TATA$ motifs in the S_{int} set. This distribution is depicted in Fig. 2-25. As can be noticed, these two distributions are very different from the distribution of $TATA$-box motifs from $pAtr$.

The distribution of the $TATA$ elements can be suitably modelled by a Gaussian type function, as given by

$$Y = ke^{-a(i-b)^2}, \tag{2.8}$$

where Y is the probability that the $TATA$ motif will be found at the appropriate position relative to the TSS; i is the position of the first T of the $TATA$-box hexamer; a, b, k are the coefficients. The fitted values of these coefficients are $a = 0.125$, $b = -30$, and $k = 0.192$. Both the experimentally obtained distribution and the modelled one are given in Fig. 2-26.

The distribution of the $TATA$ motifs in promoter sets can be regarded as one of the statistical and positional characteristics of these motifs. As Figs. 2-24 to 2-26 show,

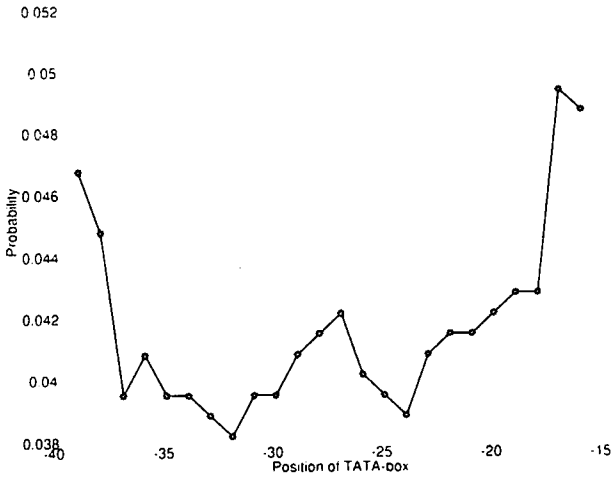

Figure 2-25: Distribution of the 5′ end of $TATA$-box motifs falsely predicted in the S_{int} set.

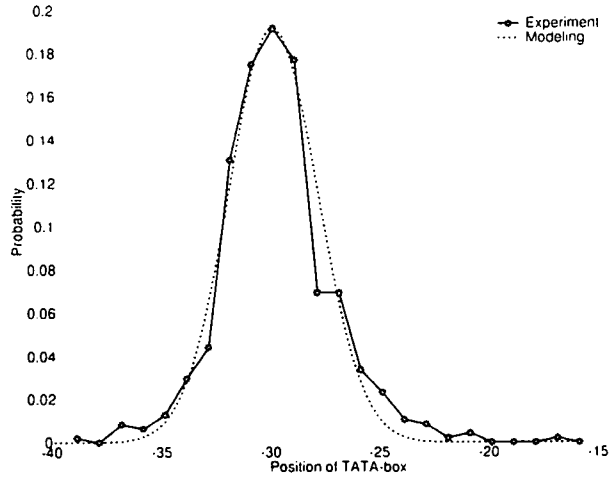

Figure 2-26: Modeling of distribution of $TATA$-box motifs in promoter set.

Position	*pAtr*	S_{cds}	S_{inf}
−39	0.4546	0.4259	0.4341
−38	0	0.4262	0.4350
−37	0.5047	0.4259	0.4355
−36	0.4544	0.4259	0.4355
−35	0.5121	0.4256	0.4352
−34	0.4758	0.4260	0.4349
−33	0.4963	0.4264	0.4358
−32	0.4916	0.4272	0.4355
−32	0.4965	0.4271	0.4359
−30	0.4983	0.4273	0.4364
−29	0.4976	0.4269	0.4363
−28	0.4998	0.4270	0.4361
−27	0.4929	0.4270	0.4360
−26	0.4998	0.4270	0.4361
−25	0.4862	0.4274	0.4359
−24	0.4720	0.4275	0.4358
−23	0.4723	0.4273	0.4365
−22	0.4889	0.4284	0.4365
−21	0.4576	0.4281	0.4360
−20	0	0.4277	0.4360
−19	0	0.4277	0.4359
−18	0	0.4279	0.4360
−17	0.4077	0.4282	0.4351
−16	0	0.4284	0.4355

Table 2.7: Average matching scores obtained for different location of motifs and for different data sets

the distribution of predicted $TATA$ motifs in promoter sets and non-promoter sets are quite different. We also assume that different positioning of $TATA$ motifs is also tied to different statistical properties of $TATA$ motif groups. For this reason, we calculated the average matching scores for each position and we found that they also vary with the position of the $TATA$ motif. Table 2.7 gives the average matching scores for both promoter sets and non-promoter sets for different positions. This also reflects specific statistical properties.

2.4 Conclusion

In this chapter we described and derived some statistical regularities of $TATA$ like motifs that can be used in the context of promoter recognition. These include positional information, PWM matching scores, and derived numerical parameters based on modified EIIP values of the $TATA$ motifs and its neighboring hexamers. These descriptions reflect the statistical and biological features of $TATA$-contained DNA sequences from different aspects. By this method, we can obtain a set of numerical parameters for each analyzed sequence. For a sequence of length $43\,bp$, we use a window with the length of 15 nucleotides to slide along the sequence, and calculate 24 possible matching scores $p_i, i = -39, -38, ..., -16$. Then, the position that corresponds to the maximum score appears to determine the most likely position of the $TATA$-box motifs. Then, the following eight numerical data are generated

$$\begin{aligned} x_1 &= \max(p_i), x_2 = (j+40)/24\ , x_3 = 0.192e^{-0.125(j+30)^2}, \\ x_4 &= AV_1,\ x_5 = AV_2,\ x_6 = AV_3,\ x_7 = AV_2 - AV_1,\ x_8 = AV_2 - AV_3 \end{aligned} \tag{2.9}$$

where j is the position where $\max(p_i)$ was found. This data is obtained by a program which is the variant of the predictor program depicted in Fig. 2-23. This data will be used in the subsequent chapters as inputs to the neural network system for promoter recognition.

Chapter 3

LVQ Neural Network for TATA-box Recognition

Artificial neural networks (ANNs) are found to be very efficient in a number of complicated tasks of pattern recognition. For this reason we will attempt to combine 'statistical filtering' based on the results of the previous chapter and capabilities of efficient classification by the ANN in order to obtain good promoter prediction.

3.0.1 Artificial Neural Network

It is well known that ANNs can efficiently solve many difficult prediction, classification, modeling, estimation, and optimization problems [12], [19], [21], [30], [54], [55], [69], [90], [98], [99], [109], [117], [112], [113]. A particularly efficient application domain of ANNs is in pattern recognition [57], [82], [107], [108], [118], [117]. ANNs can overcome some deficiencies of the conventional pattern recognition methods and thus, sometimes, achieve excellent recognition results. This suggests the utilization of ANN in promoter recognition. To design a neural network normally requires a large set of examples, but not much prior knowledge about the problem. This feature is exactly suitable for promoter recognition under the current conditions: current knowledge about eukaryotic promoter structure and its characteristics is not complete. However, there are reasonable amounts

of data available so that a computational model of eukaryotic promoters or some of their features can be successfully built.

The structure of ANN

ANNs are relatively crude mathematical models based on the neural structure of the brain. Just as humans apply knowledge gained from past experience to new situations, an artificial neural network solves new problems through a system of "artificial neurons" trained by given examples. The fundamental processing element of an ANN is an artificial neuron. In a human brain each biological neurons can connect with up to 200000 other neurons. The power of the human brain comes from the numbers of these basic components and the multiple connections between them. Naturally, artificial neurons are much simpler than the biological neurons. Figure 3-1 shows the basic schematics of an artificial neuron. Inputs to the neuron are represented by $X_n = [x_0, x_1, x_2 ... x_n]^T$. Vector $W_n = [w_0, w_1, w_2 ... w_n]$ gives the weights of input channels. When an input x_i is present it is multiplied by a weight w_i and all such products obtained from currently present inputs are simply summed to make $net = W_n X_n = \sum_{i=0}^{n} w_i x_i$, which is then fed into the neurons 'transfer elements' characterized by a function f, thus making the neuron's output $Y = f(net)$.

Biological neural systems are developed as three dimensional structures with a great freedom of neuronal connections. However, in ANNs only the simple clustering of the artificial neurons is used. As Fig. 3-2 shows, the neurons are grouped into layers, and these layers then interconnect to one another. In more complex cases, the neurons within the same layer can also communicate with each other. Usually, there are input layers, output layers and several hidden layers between them in an ANN. Figure 3-2 shows a simple structure with only one hidden layer. When the input layer in an ANN receives the input from the external environment, its nodes produce output, which becomes input to the other layers of the ANN. The process continues until an output layer produces its response.

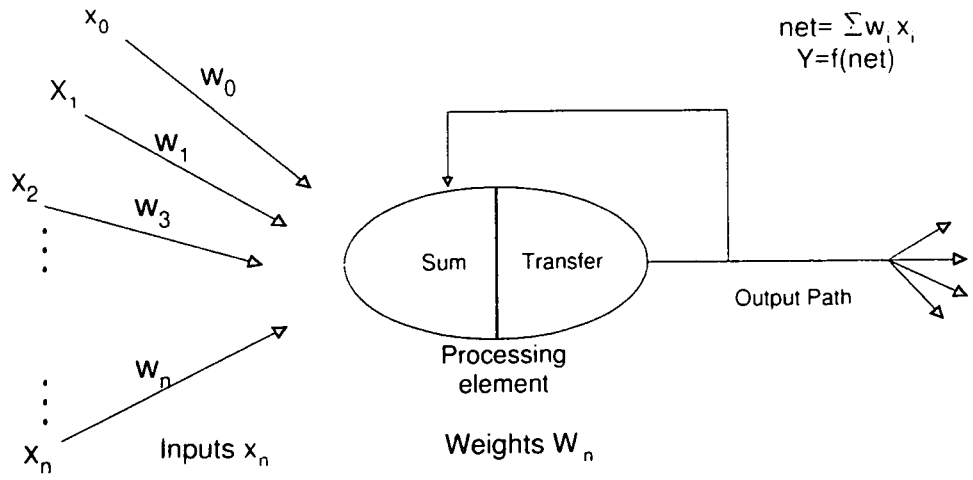

Figure 3-1: A basic structure of an artificial neuron.

Designing ANNs is a complex task which consists of a number of steps:

1. Arranging neurons in various layers.
2. Deciding the type of connections among neurons for different layers, as well as among the neurons within a layer.
3. Deciding the way a neuron receives input and produces output.
4. Determining the strength of connection within the network by allowing the network to learn the appropriate values of connection weights by using a training data set. The process may be iterative, using a training data set to determine the connection weights and the number of hidden neurons until the network performs at its best.

Learning Methods

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights can be chosen randomly. Then learning, or training begins. As the brain basically learns from experience, ANNs learn the solution to a problem by changing their connection weights during the training. Different

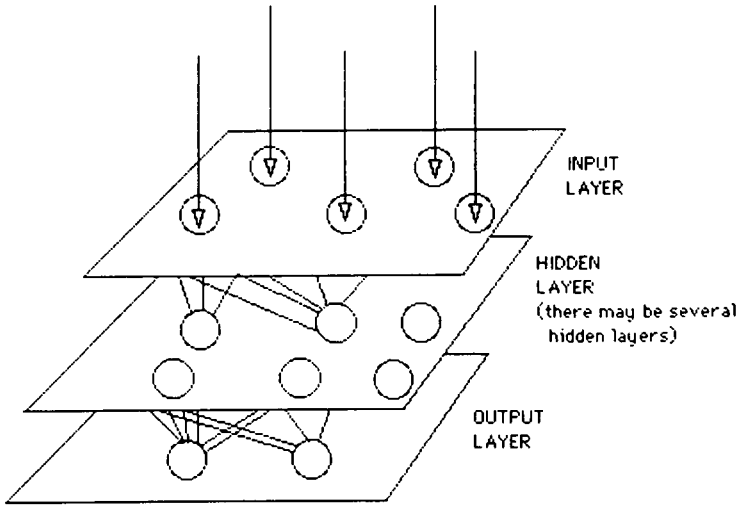

Figure 3-2: A simple structure of ANN with an input, hidden, and output layers.

learning methods and structures thus lead to various kinds of neural networks. From the viewpoint of learning approaches, there are two big categories of learning methods: an unsupervised learning scheme or a supervised one. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must decide what features it will use to organize its structure and weights to achieve good behavior. This is called self-organization or adaption. In supervised training, both the inputs and the desired outputs are provided. The network then processes the inputs and compares the resulting outputs against the desired outputs. The connection weight among the neurons are initially randomly assigned and during the training continually modified until the ANN produces the outputs sufficiently close to the desired target values.

Learning laws (rules) are mathematical algorithms used to update the connection weights. There are a variety of learning laws commonly used. Learning rules in principle make ANNs operate during the training as an optimization system. There are inputs to the ANN and (in the supervised learning system) known targets. The ANN produces the response which is the function of the current values of the ANN's weights, the error produced as a measure of deviation of ANN output and the desired targets is calculated and this information is fed back to the ANN learning rule to provide information for

determination of the next values for the weights. One can also categorize the learning methods into **off-line** or **on-line**. The so-called on-line learning is when the ANN adjusts its weights after each presented example during the learning process. Most of the real-time control applications based on ANN use this type of learning. In the off-line learning mode, all available examples are first presented to the ANN and only then the weights of the ANN are updated. The process of presenting the whole set of available examples to the ANN is called epoch. The learning process repeats for a number of epochs, until the ANN produces an overall error within acceptable limits. Most of ANNs use this type of learning.

Applications of ANNs

ANNs have found more and more applications in recent years. Basically, most of their applications fall into the following four categories ([4], [13], [57], [91], [127]):

Prediction. ANNs use input values to predict future outputs, e.g. in forecasting.

Pattern recognition or classification. ANNs use input values to recognize or classify input patterns, e.g. in letter identification, image processing, document classification.

Real-time control. ANNs use input data to realize intelligent control, e.g. in process control.

Data Filtering. ANNs alter input signals for a specific purpose, e.g. in noise reduction problems, etc.

The most successful applications of ANNs are in categorization and pattern recognition. Such ANNs classify an input pattern under investigation (e.g. an illness, an image, a chemical compound, a word, the financial profile of a customer) as one of numerous possible categories that, in return, may trigger the recommendation of an action (such

as a treatment plan or a financial plan). In many situations ANNs perform successfully where other methods do not, particularly in recognizing and matching complicated, vague, or incomplete patterns. For this reason we will apply one specific class of ANNs to the problem of $TATA$-motif recognition. It should be mentioned that ANNs found successful applications in the problems of gene and promoter recognition (see [14], [47], [48], [50], [52], [51], [53], [70], [103], [115], [124], [131], [5], [7], [8], [49], [62], [71], [72], [74], [75], [104], [105], [106]).

We will apply the Learning Vector Quantization (LVQ) neural network to enhance the problem of $TATA$ motif recognition based on statistical filtering. The fact that the basic problem of TATA recognition belongs to the area of pattern recognition suggests the use of LVQ in this area. In this chapter we use LVQ to recognize $TATA$ box motifs and in this way recognize a promoter. The use of LVQ is combined with statistical analysis as presented in the previous chapter.

3.1 Artificial Neural Network Classifiers

Different ANN models can be constructed by connecting processing elements in different ways. The relationship between the output of ANNs and input can be continuous or discrete (e.g. in a classification problem with two classes: all inputs belonging to one class correspond to output equal to 1, while those belonging to the second class correspond to output equal to 0, therefore the relationship is in the form of a discrete signal). Promoter recognition falls into a category of classification problems. A classifier system derives its name from its ability to learn to classify input messages (patterns) into general sets and is similar to a control system in many respects. As a control system uses feedback to "adapt" its output for an environment, a classifier system uses feedback to "teach" itself what is correct classification. For classification problems, many kinds of ANN can be selected, such as backpropagation (BP) ANNs, Hopfield ANNs, etc. At present Kohonen networks are the most widely used for classification. We will use here, one of Kohonen

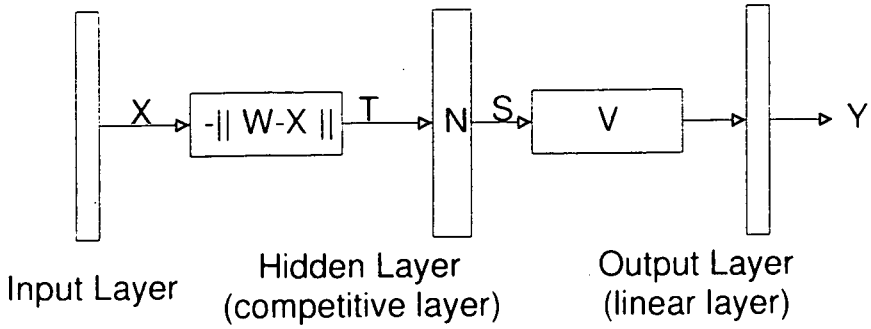

Figure 3-3: LVQ network architecture.

ANNs, the LVQ [63], [64], [65], [66]. The networks show good performance in pattern recognition ([29], [78]) and thus they are natural candidates for making the basis of our recognition system.

LVQ competitive networks are used for supervised classification. Each codebook vector (weight vector) is assigned to one of the target classes. Each class may have one or more codebook vectors associated with it. An input pattern is classified by finding the nearest codebook vector and assigning the input pattern to the class corresponding to the codebook vector. Hence LVQ performs a kind of nearest-neighbor clustering.

3.1.1 LVQ Classifier

The LVQ network architecture is shown if Fig. 3-3.

An LVQ has an input layer, a competitive layer, and a linear output layer. The competitive layer learns to classify input vectors X in subclasses in a P_1 space; then the linear layer transforms the competitive layers's classes into target classifications in a P_2 space defined by the problem. Both the competitive and linear layers have one neuron per class, thus, the competitive layer can learn up to P_1 subclasses, and these, in turn will be combined by the linear layer to form the P_2 target classes.

The LVQ is a supervised type of ANN. It basically uses the nearest-neighbor learning rule. Because the linear layer is only used to transform the competitive subclassifications

into target classifications, its weights are fixed to be 1 or zero after the ANN structure is initialized. Thus, the weights of the output layer are not subject to adjustment during the training.

Regarding the ANN given in Fig. 3-3, let us assume that the LVQ has N nodes in the input layer, M nodes in the hidden layer, and K nodes in the output layer. This means that the input pattern vectors have N components and that they should be classified into K target classes. The weight matrix $W_{M\times N}$ describes the connections from input layer to the hidden layer. It is also known as the codebook vector; $V_{K\times M}$ describes the connection from the hidden layer to the output layer. $X_{N\times 1}$ is the input signal (input pattern); $S_{M\times 1}$ is the output from the hidden layer, while $Y_{K\times 1}$ is the LVQ's output signal. The output from the hidden layer for any input signal will have only one node which outputs 1, while all others will output 0. In the output layer only one node will output 1. For the output of the LVQ the vector $[1,0,...0]^T$ presents the first class, $[0,1,0...0]^T$ presents the second class, and so on.

Because $V_{K\times M}$ is constant in LVQ, we need only to tune $W_{M\times N}$ using training data. The objective of the learning procedure is to place the codebook vectors W in the input space in such a way, so as to describe well the boundaries of the classes by taking into account data from the training set. Thus, the LVQ algorithm variants attempt the optimal placement of codebook vectors in the input space, so as to optimally describe class boundaries. This is achieved through an adaptive iterative process. Class boundaries are segments of hyperplanes placed at the mid-distance of two neighboring codebook vectors that belong to different classes. The update rules of the weight vector in the hidden layer for the input vector X at input layer is given by

$$dW_i = \eta * (X - W_i),\ dW_j = -\eta * (X - W_j) \tag{3.1}$$

where dW_i represents the required increments of the codebook vector, and η is the learning rate which usually is taken to be a small positive constant. The subscripts i and j relates

to the node whose weights are reinforced or anti-reinforced, depending on whether the node made correct classification of the input pattern or not. This learning rule belongs to the so-called lvq2 type algorithm [65], [66]. Once learning is complete, new input patterns X in the test set are assigned to the class i whose relative weight or codebook vector W_i is the nearest to X.

For a classification problem, we know that the number of input nodes and the output nodes can be easily determined. However, determination of the number of hidden nodes and the learning rate are not so transparent, especially in the absence of a prior knowledge of the class probability densities. Too few hidden nodes may not be enough for good class separation, while too many lead to prohibitive learning times and bad generalization results. Thus, a compromise has to be achieved and determined through experimentation for the best results.

3.2 Data Preprocessing

For the quality of ANN training the preparation of the training data in very important. Proper data preparation enables ANNs to extract efficiently important information from the data.

3.2.1 Data interpreted from DNA sequences

Frequently, DNA sequence data is directly used for ANN training [3], based on binary codes for nucleotides (as discussed before) which have the same Hamming distance between any two nucleotide codes. However the input vector may be of considerable dimension, the ANN structure may be complex, so it may easily happen that the resulting ANN has too many weights that require adjustment and this may not be feasible. Moreover, using binary coding of nucleotides does not allow virtually any preprocessing of data so as to make ANN data processing more efficient. For this reason we will use the modified EIIP values to represent nucleotides as commented in Chapter 2 and will apply proven

statistical techniques to prepare data for efficient ANN processing.

The data we selected for the training and test set are described in Chapter 2. Scanning the DNA sequence with a window of appropriate length as described in Chapter 2, we generate for each window position 8 numerical data that will be source information used for ANN training. Then a basic feature vector $X = [x_1, ..., x_8]$, is obtained for data window as (see Chapter 2)

$$\begin{aligned} x_1 &= \max(p_i), x_2 = (j+40)/24\ , x_3 = 0.192e^{-0.125(j+30)^2}, \\ x_4 &= AV_1, x_5 = AV_2,\ x_6 = AV_3,\ x_7 = AV_2 - AV_1,\ x_8 = AV_2 - AV_3 \end{aligned} \tag{3.2}$$

Due to the statistical feature of PWM matching scores and the distribution region of their average values, we first use these statistical regularities to make initial filtering of the data. This process significantly reduces data from the non-promoter group, while the reduction of data from the promoter group is relatively small.

At first, we set a threshold τ for the matching score p_i, and only those vectors whose first element x_1 is larger than the threshold τ will be selected and used for the further analysis. Others will be regarded as those that do not contain $TATA$ motif, and thus as non-promoters. So, we get $X_{f1} = \{X \mid x_1 >= \tau\}$. We also use the distribution of x_7 and x_8 as a bounding filtering condition based on the analysis from Chapter 2. So finally we get a data set $X_f = \{X \mid x_1 >= \tau,\ x_7 \in [0.1, 1.3],\ x_8 \in [0.1, 1.2]\}$. This we will call the first statistical filtering.

When this first statistical filtering is applied to the training data sets $pAtr$ and S_{cds}, the results of the achieved TP and FP scores for different values of the threshold τ are given in Fig. 3-4. One can observe the significant difference between TP and FP. This difference is above 50% when threshold $\tau \in [0.36, 0.48]$.

Thus, data in the training and test sets is represented by the vectors X which will be used for further analysis, and eventually as inputs to the LVQ. In other words, only those

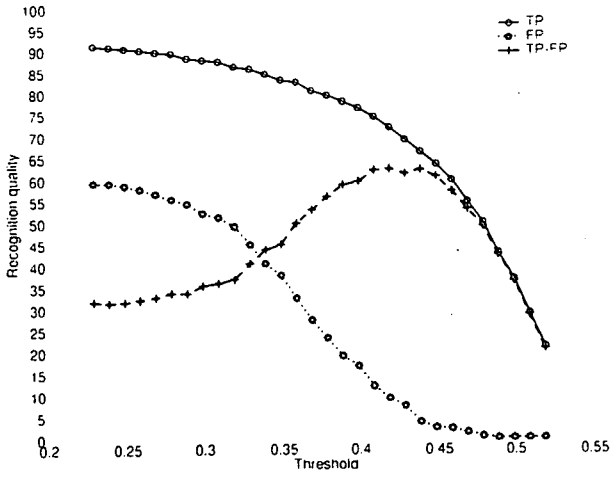

Figure 3-4: The efficiency of the first statistical filter on the training sets $pAtr$ and S_{cds}.

sequences in a training or test set predicted as potential $TATA$ -containing sequences by the first statistical filter will be characterized by the 8-element vector and used for further analysis.

3.2.2 Principal component analysis

After input data from the training sets are filtered by the first statistical filter. those that passed this filtering are to be prepared for efficient ANN processing. To do this we apply the principal component analysis (PCA) which helps in such data preparations. With the PCA we will be able to preserve as much as possible of the original information content, and possibly to reduce the dimension of the input data, thus enhancing the generalization ability of the ANN. PCA is a classical statistical method which has been widely used in data analysis and compression ([23], [46]). PCA is a linear procedure to find the direction in input space where most of the energy of the input lies. In other words, PCA performs specific feature extraction. Let the set of 8-component vectors that correspond to the filtered promoter data be denoted by X^p, while that of the non-promoter data by X^{np}. The whole PCA process is applied as follows:

Step1. Normalization. All data vectors from X^p are normalized to have a zero-mean and a standard deviation of 1. Let us assume that there are N_1 filtered promoter sequences. Then the normalization is done by making $X^p_{norm} = \left(X^p - \overline{X^p}\right)/\sigma$, where the mean $\overline{X^p}$ and standard deviation σ are determined as follows:

$$\overline{X^p} = \frac{1}{N_1}\sum X^p, \quad \sigma = \sqrt{\sum (X^p - \overline{X^p})^2} \tag{3.3}$$

Step2. Applying a PCA method to pre-processes the network input training set means transforming X^p_{norm} so that the elements of the transformed data are uncorrected. This is done by a linear transformation by means of a matrix M_{pca} composed of eigenvectors of the covariance matrix of the data, where the eigenvectors are in descending order. Such a transformation will make the variability in data clustered so that the greatest variability is in the first component of the transformed data, and so on. In addition, the size of the transformed data vectors may be reduced by retaining only those components which contribute more than a specified fraction of the total variation in the data set. In our case we selected only that data that contributes more than 1% to the total variability in the data set, and apply this to the normalized data X^p_{norm}. Consequently we get $X^p_{pca} = M_{pca} * X^p_{norm}$. By retaining only those components that make more than 1% variability in the total data set we obtained a reduction of the data vectors from 8 to 6 components. So, each vector in X^p_{pca} has only 6 components. Any other data vectors to be used in the analysis should be transformed according to $X_{new} = M_{pca} * \dfrac{X_{new} - \overline{X^p}}{\sigma}$, where $\overline{X^p}$ and σ were determined in Step 1.

So, the final input to the LVQ consists of 6-point data vectors for each filtered sequence.

3.3 Sequence Classification by LVQ

3.3.1 LVQ training and test

The data used for the LVQ training and test contains data initially filtered by the first statistical filter and then normalized and PCA transformed as described. Thus the LVQ input data is represented by vectors of dimension 6. The LVQ training algorithm used is lvq2. At the beginning of the LVQ training the weight is randomly initialized. It is customary to specify the proportion of the neurons in hidden subclass layers that are to belong to different classes. As, in our case, we have two class problems (data representing a $TATA$ motif and data not representing that motif), we need to specify only the proportion of neurons for these two classes. They are given as 0.4/0.6, i.e. 40% of the neurons in the hidden layer will relate to Class 1($TATA$ motif class) and 60% of the neurons relate to Class 2($TATA$ less class). Initially, the number of nodes in the hidden layer is set to 500, and the number of learning epochs to 1000. We will vary these parameters to examine their effects.

A snapshot of the organization for a trained LVQ for a 2-class problem is shown in Fig. 3-5. The inputs are classified into two categories in the target classes layer, and outputs are finally produced by the Logic judge. Practically, the neurons which relate to the different classes in the subclasses layer are scattered randomly when LVQ is initialized. Through the training process they become associated with the right clusters in some statistical manner.

3.3.2 Effects of Threshold Variation

We used a threshold in the statistical filtering of the first statistical filter. The results of threshold variation is summarized in Table 3.1. "$CC_{training}$" denotes the correlation coefficient of the achieved score on the training set, while "CC_{test}" represents the correlation coefficient for the test set.

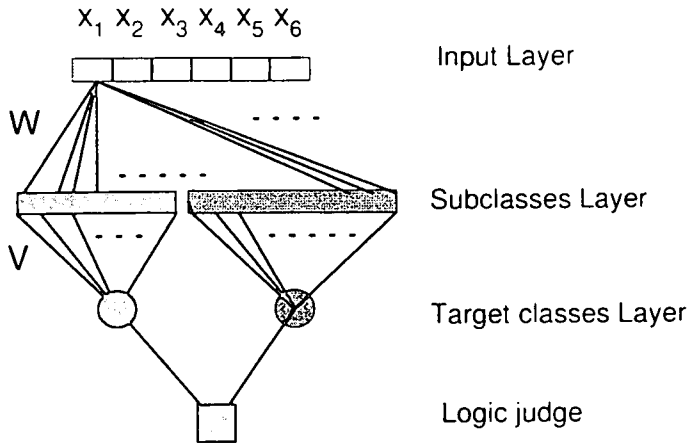

Figure 3-5: Structure of a trained LVQ for a two class problem.

It can be seen that the CC for the training set varies around 0.7 which is rather good, while CC for the test set is around 0.4. For the training set $TP = 61.5625\%$, while $FP = 0.2125\%$ which is good ($CC = 0.75$). If we adjust the threshold, the neural network recognizes 49.5798% promoters in the test set with only 1.45% false positive predictions.

When the number of nodes in the hidden layer is changed to 100 and the number of epochs to $10,000$, then the results of threshold variation are summarized in Table 3.2. In this case we can achieve a very satisfactory $FP = 0\%$ while having $TP = 50.6250\%$ on the training set. Unfortunately, the recognition performance for the test set is rather low.

Discussion

The result achieved by the LVQ system shown in Table 3.1 and Table 3.2, suffer from the common problem of recognition systems: the results on the test sets are much worse than

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.3600	59.8438	1.0375	0.6818	54.2017	2.3375	0.4523
0.3700	60.9375	1.4000	0.6665	53.7815	2.8500	0.4195
0.3800	59.8438	0.9750	0.6860	51.6807	2.0250	0.4551
0.3900	59.0625	0.9125	0.6847	50.8403	2.0750	0.4454
0.4000	61.2500	1.2125	0.6805	52.5210	2.3625	0.4388
0.4100	59.5312	0.3250	0.7319	49.5798	1.4500	0.4853
0.4200	60.3125	0.2125	0.7464	48.7395	2.1625	0.4240
0.4300	61.5625	0.2125	0.7548	52.1008	2.4625	0.4297
0.4400	56.2500	0.1000	0.7282	42.8571	1.5000	0.4278
0.4500	55.1562	0.1125	0.7195	43.2773	1.3000	0.4492
0.4600	53.9062	0.0875	0.7130	39.9160	1.4750	0.4057
0.4700	51.2500	0.0750	0.6952	38.2353	1.6875	0.3748
0.4800	46.5625	0.0750	0.6608	34.8739	2.0875	0.3201
0.4900	41.4062	0.0125	0.6276	29.4118	1.5750	0.3059
0.5000	35.9374	0.0375	0.5804	22.6891	0.6250	0.3310
0.5100	28.5937	0.0500	0.5137	17.6471	0	0.4150
0.5200	20.6250	0.0625	0.4311	11.7647	0	0.3386

Table 3.1: Results for promoter recognition: first statistical filter + LVQ.

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.3600	57.1875	0.7375	0.6836	51.2605	5.4250	0.3060
0.3700	58.5938	0.4500	0.7154	44.9580	6.2375	0.2484
0.3800	57.0312	0.6875	0.6862	44.9580	7.2500	0.2284
0.3900	62.0313	0.9125	0.7057	53.3613	4.5750	0.3445
0.4000	59.6875	0.3875	0.7279	47.0588	4.7125	0.3002
0.4100	61.8750	0.2625	0.7529	50.8403	4.9250	0.3177
0.4200	61.7187	0.2250	0.7549	51.2605	4.2500	0.3422
0.4300	63.7499	0.2375	0.7674	52.9412	4.1750	0.3557
0.4400	60.0000	0.0500	0.7581	48.3193	2.8375	0.3819
0.4500	59.2188	0.1000	0.7485	47.0588	2.7375	0.3780
0.4600	56.5625	0.0750	0.7326	43.2773	2.2750	0.3758
0.4700	50.6250	0	0.6979	39.0756	2.3000	0.3417
0.4800	49.6875	0.0375	0.6875	37.3950	2.1875	0.3348
0.4900	41.4063	0	0.6289	31.0924	1.4000	0.3348
0.5000	35.4687	0	0.5808	23.5294	0.4625	0.3657
0.5100	28.5937	0.0125	0.5185	18.4874	0	0.4248
0.5200	20.1562	0	0.4353	12.6050	0	0.3505

Table 3.2: Promoter recognition: Second set of results

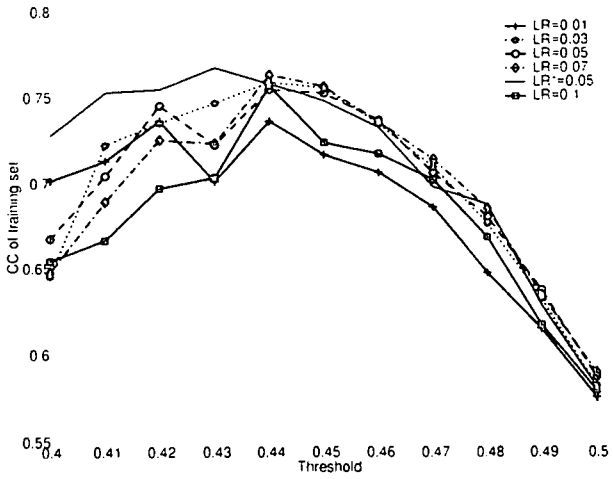

Figure 3-6: Recognition quality of the LVQ system under different learning rates.

those on the training set. To discuss this phenomena, we will vary the training set to see the effect of training data to recognition quality. Additionally, some parameters of the LVQ algorithm are varied in order to determine their effects on the overall classification performance.

As can be observed, the TP and FP reduce with the increase of the threshold. In fact, the threshold here acts as a filter, and by it we can control the amount of data that will eventually be used for training LVQ. Therefore we can use a smaller scale LVQ by proper selection of τ. However, the disadvantage of increasing τ is that it may lose information required for good LVQ training. We observe that the best results are achieved with $\tau \in [0.41, 0.47]$.

3.3.3 The Effects of Different Learning Rates

In Fig. 3-6, we show the change of the achieved CC when different learning rates (LR) are used and when different thresholds are applied. Different LRs contribute to different trajectories in the parameter space during the LVQ training and this may lead to different, good or bad, minima which impact on the final recognition quality of the

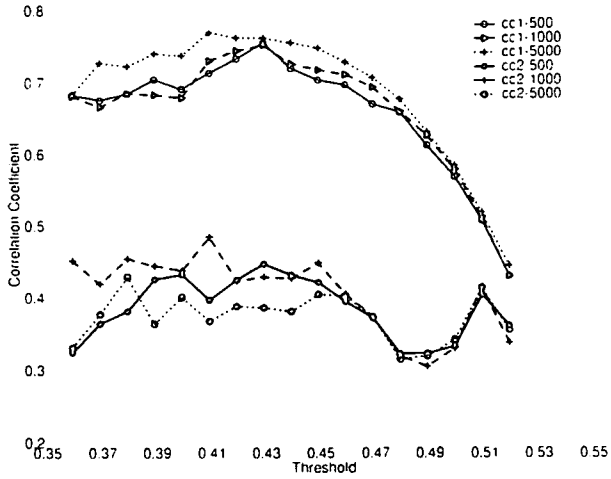

Figure 3-7: Variation of the correlation coefficient with the threshold obtained for different number of epochs.

system. The number of nodes in the hidden layer is set to 500, and the number of epochs 1000. No significant difference is observed as shown in Fig. 3-6 when learning rate varied from 0.01 to 1. We notice that $LR = 0.05$ only a bit better than others.

3.3.4 Effects of the Number of Epochs

In Fig. 3-7 we present the variation of the CC against the threshold, as the number of learning epochs change. Explanation of the curves and legend in Fig. 3-7 is as follows:

cc1-500 denotes CC for the training set when the number of epochs was set to 500

cc1-1000 denotes CC for the training set when the number of epochs was set to 1000,

cc1-5000 denotes CC for the training set when the number of epochs was set to 5000,

cc2-500 denotes CC for the test set when the number of epochs was set to 500,

cc2-1000 denotes CC for the test set when the number of epochs was set to 1000,

cc2-5000 denotes CC for the test set when the number of epochs was set to 5000.

The number of epochs given relates only to the training phase.

From the results given in Fig. 3-7, we cannot notice a significant difference between the recognition quality for different numbers of epochs. One can observe that increasing

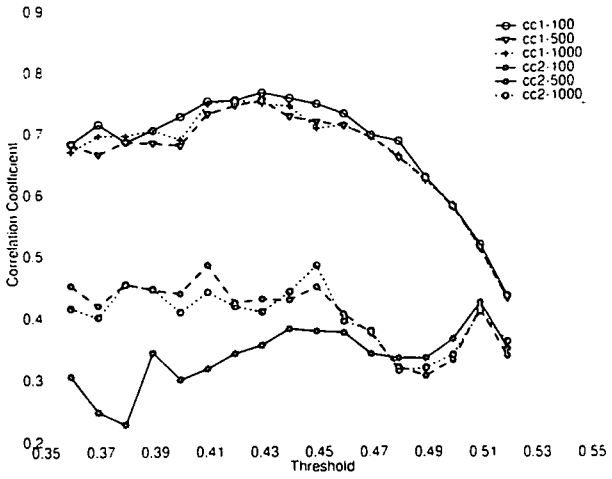

Figure 3-8: Correlation coefficient variation against the threshold obtained for different number of nodes in the hidden layer.

the number of epochs does not always result in a better recognition quality. The largest number of epochs (5000) gives the lowest CC for the test set, but it gives the best CC curve for the training set. It seems 1000 epochs brings a reasonable compromise.

3.3.5 Effects of the Number of Nodes in the Hidden Layer

A certain relationship exists between the recognition quality and the number of nodes in the hidden layer. In our case, as shown in Fig. 3-8, increasing the number of nodes in the hidden layer does not always result in better detection.

The explanation of the annotation in Fig. 3-8 is as follows:
cc1-100 denotes CC for the training set when the number of nodes is set to 100,
cc1-500 denotes CC for the training set when the number of nodes is set to 500,
cc1-1000 denotes CC for the training set when the number of nodes is set to 1000,
cc2-100 denotes CC for the test set when the number of nodes is set to 100,
cc2-500 denotes CC for the test set when the number of nodes is set to 500,
cc2-1000 denotes CC for the test set when the number of nodes is set to 1000.

As can be observed from Fig. 3-8. the best overall results are obtained when the number of nodes is set to 500. When the number of nodes is set to 1000 the CC for the test set decreases a bit. Selection of the number of nodes in the hidden layer should be related to the number of observations. Actually, the biggest number of inputs to LVQ in our case is no more than 6000 (after first statistical filtering), and too many nodes in the hidden layer may cause the case of "overgeneralisation" [128], which means if you allow uncontrolled growth of the number of hidden nodes then eventually the LVQ will learn the noise in the training set.

We also set the number of nodes to 25, 30, 40, etc. in the hidden layer. However, they produced similar, but slightly worse results than when the number of nodes was 100. For too small a number of nodes in the hidden layer the large number of input data may pull the weight vector for each node in the hidden layer away from the optimal decision regions, so that increasing the number of nodes may alleviate this problem. In fact, some observations were made that the number of nodes in the hidden layer for different classification problems should be chosen as a function of each classification variance. We however, will note that 500 nodes produces overall the best results.

3.3.6 Effects of Different Initial Weights

As with many learning systems, the initialization of the weights in the LVQ is crucial to the ultimate success of the learning process. When we randomly generated three sets of initial weights (cases A. B. and C), and trained the LVQ, it resulted in the CC change as depicted in Fig. 3-9.

In Fig. 3-9, cc1 denotes the CC for the training set, while cc2 denotes the CC for the test set. A, B, and C stand for the three different initial weight vectors. The results are obtained when the number of hidden layer nodes was 500, when the learning rate was 0.05 and when 1000 epochs were used.

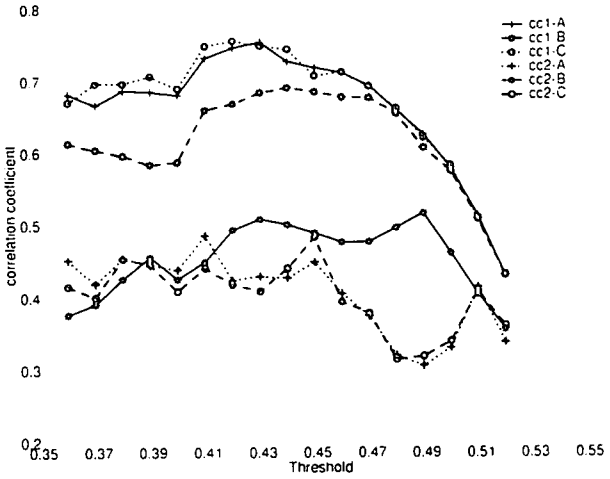

Figure 3-9: Correlation coefficient against the threshold for three difference set of initial weights.

3.4 Conclusions

The recognition accuracy of the system with a single LVQ does not appear to be very good, although the system performs reasonably well. One can observe that from all investigated parameters, the initial weights have the greatest influence on the final recognition quality of the system. Thus in the next chapter, we will perform a more sophisticated determination of the initial weights of the LVQ by means of a genetic algorithm, in an attempt to obtain improved recognition accuracy of the overall system. We will combine this with two level LVQ models and statistical filtering, which will in the end produce an effective system for recognition of $TATA$ containing sequences.

Chapter 4

Multistage LVQ for TATA-box Recognition

In the last chapter, we attempted to use LVQ to recognize promoters based on the recognition of the $TATA$ motif. The results obtained were promising, but still insufficiently good. In the examination of the parameters that influence the prediction quality and which are subject to our control, we noticed that the greatest influence is made by changing initial values of the weights in the LVQ. It is known that the LVQ usually performs efficiently if the initial weight vectors are close to their final values. That is why frequently some heuristic procedures are used for weight initialization, such as k-means for Self Organizing Map (SOM) ANNs or any other vector quantization procedure. The number of weight vectors remains fixed during training. Usually this number is set prior to training to be equal to the number of classes, or to a fixed number of weight vectors per class. Several sets of initial weight vectors are usually tested and the best solution is chosen.

In this light we will attempt to find good initial values of the weights for the LVQ. The best way to solving this problem is to optimize a LVQ network globally. This means optimizing the initial weight vectors, and the number of weight vectors as well. In this chapter we will discuss how to use genetics algorithms (GAs) to tune the weights, instead

of tuning them from randomly selected initial values by the lvq2 algorithm. Based on these results we will develop our final system for promoter recognition.

4.0.1 Genetic Algorithms

The Genetic Algorithm is an artificial intelligence procedure which was first described by John Holland in the 1975 ([56], [40]). *GA* is a stochastic search algorithm based on the mechanics of natural selection. It is a global search method and thus it may offer significant benefits over more typical local optimization approaches.

Traditional search and optimization methods can be classified into two distinct groups: direct and gradient-based methods ([28]). Direct search methods are usually slow, requiring many function evaluations for convergence. On the other hand, gradient-based methods quickly converge, but they are not efficient in non-differentiable or discontinuous problems and usually stack to local minimum. Commonly, the two methods may converge to an optimal solution depending on the chosen initial solution and most algorithms tend to be attracted to locally optimal solutions from where they cannot easily escape. Contrary to this, *GAs* are proven to provide a robust search in complex spaces which can solve various search and optimization problems and are not prone to such problems ([10]).

GA performs its search balancing the need to retain population (the solution for a problem) diversity, so that potentially important information is not lost, with the need to focus on fit portions of the population. Simply, *GA* is a method for moving "chromosomes" from one generation to an new generation, using genetic operators of reproduction, crossover, and mutation ([35], [39], [59], [60], [76], [77], [79], [80]).

In the *GA* processing, individuals are represented by a linear string of letters of an alphabet and they are allowed to mutate, crossover and reproduce. In each generation, a subset of parents and offsprings enters the next reproduction cycle. The process iterates until the population is in compliance with the termination criterion. The basic outline of a genetic algorithm is shown is Figure 4-1.

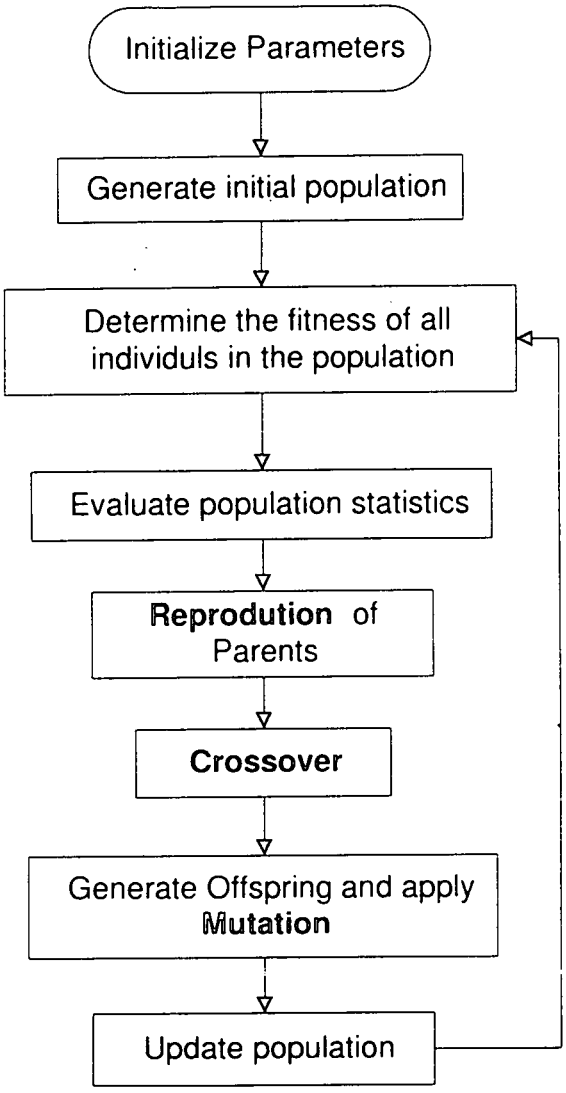

Figure 4-1: Simple flowchart of GA.

In Fig. 4-1, activities in the block entitled "Generate initial population" can be done either randomly or with specific background knowledge to start the search. Then during iterations, the fitness of all individuals in the population is determined. Therefore in Fig. 4-1 each loop represents one epoch. A generation is generated initially, and after each epoch. In each epoch, the activities are as follows:

- **Reproduction,** determines the population which will be reproduced, and that will become potential parents. There are many possible reproduction algorithms. In some reproduction strategies, just a certain percentage of the strongest individuals are selected. However, most research has shown that stochastic selection biased by fitness is more productive. For stochastic reproduction, the reproduction probability is proportional to the fitness of an individual. During reproduction, highly fit individuals have a greater probability of producing offspring for the next generation than less fit individuals. This approach guarantees that good individuals are not lost during a run. With total generation replacement it can happen that good individuals die out because they produce only inferior offspring. A fitness proportional reproduction is a simple rule where the probability of reproduction during a given generation is proportional to the fitness of the individual. For example, for a individual i, its probability to be selected is given simply by:

$$P_i = \frac{S_i}{\sum_{k=1}^{n} S_k} \tag{4.1}$$

  where, P_i is the probability of reproduction for individual i. S_i is the fitness of the individual i, and n is the total number of individuals. This gives every member of the population a finite probability of becoming a parent, with stronger individuals having a better chance.

- **Crossover,** which takes a portion of each parent and combines the two portions to create offspring. After reproduction, the potential parents are prepared to generate offspring. The mate for each parent is randomly chosen. So, the two parents

are aligned and one location on their strings is randomly chosen as the so-called crossover site. Now the parts from the beginning of the individuals to the crossover site are exchanged between them. The resulting hybrid individuals are taken as the new offspring individuals. Although the mechanisms of the reproduction and crossover operators are simple, they make up much of the power of *GAs*.

- **Mutation,** which is the random alteration in a string position, plays a secondary role in the reproduction process. Mutation is needed to ensure variations and to guard against premature convergence, as well as to guarantee that any location in the search space may be reached. The frequency of mutation is about one mutation per ten thousand position transfers biologically. In *GA*, however, it varies according to applications.

- **Replacement** introduces new individuals into a population. For the classic *GAs* which have constant population size, for each new individual created, another individual must be eliminated.

Although a *GA* is a simple algorithm, it is robust and it has been found that it is capable of solving various search and optimization problems. An advantage of *GAs* is their simplicity, global operation mode, and inherent parallel processing. A strong disadvantage of *GAs* is the required computational time. *GAs* are slower than most of the other search and optimization methods. For numerous fields of application of *GAs* see [41], [68], [61], [73], [86], [111].

4.1 Working principles of genetic algorithms

GAs are a part of the evolutionary computing field. A *GA* is a search and optimization method developed by mimicking the evolutionary principles and chromosomal processing in natural genetics. From a numerical point of view it represents an iterative optimization procedure which operates with a number of solutions (populations) in each iteration.

Normally a GA begins its search with a randomly assigned initial population in the absence of knowledge of the problem domain. Then three different genetic operators: reproduction, crossover, and mutation, are applied to update the population and in this way complete one iteration. The process repeats until a sufficiently good solution is found.

The first thing we should know regarding the GA is how to create a chromosome or an individual. For example, in an ANNs initial weights initialization problem we can use the whole set of weights as an individual. The next question is how to select parents for the crossover operation. This can be done in many ways, but the main idea is to select the better parents (in a hope that the better parents will produce better offsprings). But in order not to lose the best chromosome in each epoch during the search, at least the best solution is copied without changes to a new population throughout any 'genetic' procedure, so that the best solution can survive to the end of the process.

We will discuss the detail of creating an individual and GA operators in the following subsection, which is the basis for applying GA in initializing weights of LVQ.

4.1.1 Encoding

When we are starting to solve problems with GA, we first should know how to encode individuals (chromosomes). The encoding can be different depending on the problem. The individual should in some way contain information about a solution which it represents. Here we comment on some encoding schemes which have been widely used.

1. **Binary Encoding** Binary encoding is the most common encoding mechanism, mainly for historical reasons, as the first works on GA used this type of encoding. In binary encoding every individual is represented by a string of bits, expressed by 0 or 1, such as

   Individual A =101100101100101011100101

   Individual B =111111100000110000011111

Each individual is represented only by one binary string, and each bit in the string represents a particular characteristic of the solution. Binary encoding produces many possible individuals even with a small number of bits. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

2. **Permutation Encoding** Permutation encoding can be used in ordering problems, such as in a task ordering problem. In permutation encoding every individual is a string of numbers, which represent a position in a sequence, such as,

   IndividualA, 178964325

   IndividualB, 951723648

3. **Value Encoding** Direct value encoding can be used in various problems. Sometimes the use of binary encoding for a specific type of problems could be very difficult. In value encoding, every individual is a string of some values. Values can be anything related to the problem, real numbers, or letters, or categories, such as,

   Individual A : 1.2345, 2.6678, 0.9789, 1.5288

   Individual B : $AVTGCRUABCEKMSNUVNQP$

   Individual C : $(black), (red), (green), (yellow)$

   Value encoding is very good for some special problems. However for this encoding it is often necessary to develop some new crossover and mutation operators specific for the problem. For a neural network with a given architecture, real values in individuals can represent corresponding weights for inputs. In the next section, we will use value encoding in a GA for the search of LVQ. weights.

4. **Tree Encoding** Tree encoding is used mainly for evolving programs or expressions, and for genetic programming structures.

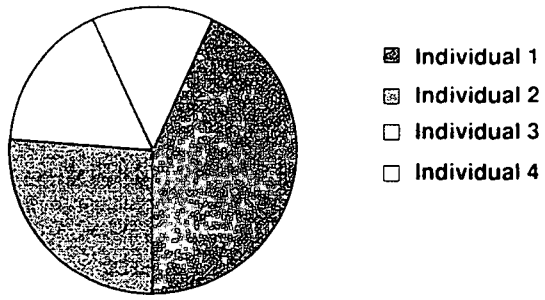

Figure 4-2: Roulette wheel.

4.1.2 Reproduction

A reproduction is usually the first operator applied on a population. By the reproduction process, individuals from the population are selected to be copied into a mating pool for crossover. The problem is how to select these individuals. From Darwin's evolution theory, the essential idea is that the fitter individual from the current population survives and duplicates of them are inserted in the mating pool. There are many methods for the selection of the best individuals, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady-state selection, and a number of others. We will comment on some of these.

A. Roulette Wheel Selection

The commonly used reproduction operator is the fitness proportional reproduction operator, which is described by (4.2), where parents are selected according to their fitness. The better the individuals are, the more chances they have to be selected. One way to achieve this proportional reproduction is to use a roulette wheel as shown in Fig. 4-2, where all individuals in the population are allocated and angle proportional to their fitness. The roulette wheel is spun n times, where n is the population size. During each round an individual is chosen by the roulette wheel pointer. Obviously an individual with larger fitness will be selected more times. This method is inherently slow and somewhat noisy.

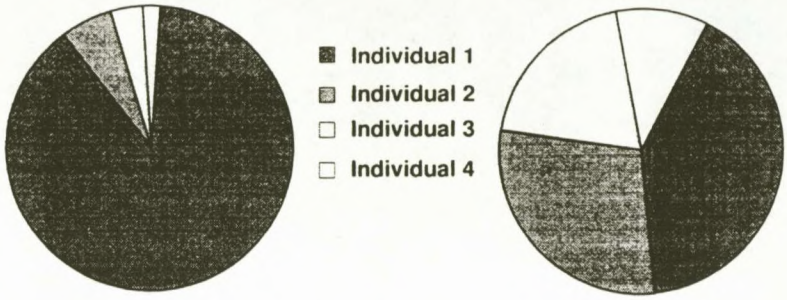

Figure 4-3: (a - left) graph of fitness; (b - right) graph of order numbers; (a) represents situation before ranking, while (b) represents situation after ranking.

B. Rank Selection

The roulette wheel selection will have problems when the fitness of different individuals differ very much. For example, if the best individual fitness is 90%, then the other individuals will have very little chance to be selected (see Fig. 4-3(a), the individuals #2, #3 and #4 have a very small allocated angle according to their fitness). By using rank selection, all individuals in the population are ranked according to their fitness, then the numbers are assigned to them from 1 to n, where the worst case will have fitness 1, second worst 2, etc., and the best will have the fitness n (number of individuals in the population).

Figure 4-3 shows how the situation changes after changing fitness into order number. In Fig. 4-3(b), the individuals #1, #2, #3, and #4, have their ranks as 1, 2, 3, and 4, respectively. Therefore they are assigned numbers 4, 3, 2 and 1 according to their fitness, and consequently allocated angles proportional to their rank. Thus, the individuals #2, #3 and #4 have better chance to be selected than if fitness is as in Fig. 4-3(a).

C. Steady-State Selection

The main idea of this selection is that a large part of the individuals should survive to the next generation. In every generation, a few fitter individuals are selected for creating new offsprings. Then some individuals with lower fitness are replaced by the new offsprings, and the rest of the population survives to the new generation.

4.1.3 Crossover and Mutation

Crossover and mutation operators are applied after the reproduction. Crossover is mainly responsible for the search aspect of *GAs*. The mutation serves for this purpose too, but in addition it is also needed to maintain diversity in the population. The type and implementation of genetic operators depends on encoding and also on the problem.

There are many ways to make crossover and mutation, but in almost all crossover operators, two strings are selected from the mating pool at random, and some portion of the strings are exchanged between the strings. In this section we only comment on some simple examples for several encoding types.

A. Binary Encoding

1. Crossover A single-point crossover is such an operation where two individual strings are cut at a random position and the binary string from the beginning of an individual to the crossover point is copied from one parent, while the rest is copied from the second parent, such as depicted in what follows:

$$1100|1010 + 1101|1111 = 11001111.$$

The first four bits from the first parent are copied to a new individual, as well as the last four bits of the second parent.

In a two-point crossover case, two random sites on an individual are chosen and the contents cut by these sites are exchanged between the two parents individuals. This idea can extended to a multi-points crossover operator. The extreme case of this is the so-called uniform crossover, where bits are randomly copied from the first or from the second parent. We illustrate the two-point crossover operation by the following:

$$110|010|11 + 110|111|01 = 11011111$$

Other methods can also be used such as arithmetic crossover, where some arithmetic operation is performed to make a new offspring. In the following example the local AND operation is used on the whole strings of both parents:

$$11001011 + 11011101 = 11001001(AND)$$

2. Mutation operator is normally applied after crossover operation and it changes a 1 to a 0 with a small mutation probability, such as illustrated in what follows

after crossover 11001001

mutation

after mutation 10001001

B. Permutation Encoding

1.Crossover In a single-point crossover one crossover point is selected. Up to this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added. There are more ways how to produce the remaining part after crossover point:

$$(12789|6453) + (894536721) = (127894536)$$

2. Mutation makes order changing: two numbers are selected and exchanged such as in

$$(12\check{7}894\check{5}3\check{6}) => (126894537)$$

C. **Value Encoding**

1. **Crossover** All crossover variants from binary encoding can be used.

2. **Mutation** consists in adding a small number (for real value encoding) to selected values. This operation also can be used in a single-point or in multi-points.

$$(2.66, 3.55, \overline{2.35}, 1.89) => (2.66, 3.55, 2.38, 1.89)$$

The main purpose of the crossover operator is to enable a search in the parameter space, as well as to perform (in a way) preservation of information stored in the parent individuals selected as the fitter ones by the reproduction operator. So, not all individuals in the population participate in the crossover operation, and there exists a crossover probability for each individual to be part of the process.

4.2 GA for optimizing the initial weights of LVQ

As we know, the initial weights play an important role in ANN training. Bad initial weights may make the learning process converge slowly or get stuck in a locally optimal solution, or even not to converge. GA is a global optimization method, and so we will attempt to use GA to search for relatively good initial weights for LVQ.

4.2.1 Method

Our implementation of GA is a bit different to the traditional GA. The population in GA in our case consists of a family and members. There are $number\_Pops$ families, each family has $number\_Max\_pop$ individuals, and each individual represents a weight vector (or codebook-vector) of LVQ. In order to use an LVQ to recognize $TATA$ motifs, one has first to determine the structure of LVQ. We focus on using the GA to search for initial weight vectors, while we fix the number of nodes in the hidden layer. As shown in

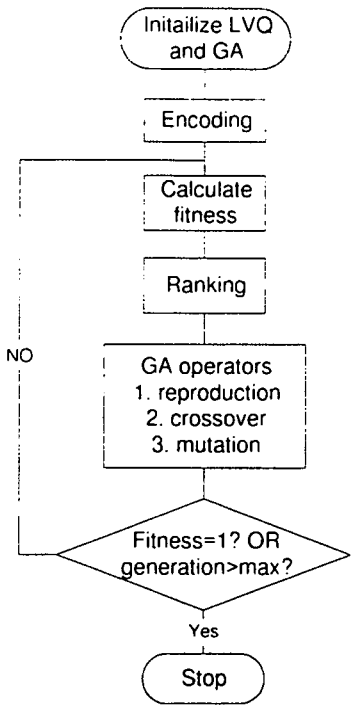

Figure 4-4: Flowchart for GA.

Fig. 4-4, we will start with encoding as well as with GA operators.

1. Encoding. We apply real value encoding. In an LVQ, only the weights from the input layer to the hidden layer need to be trained. If N denotes the size of the input layer and M the number of nodes in the hidden layer, then the number of these weights is $N \times M$. Therefore the length of an individual in our GA is $N \times M$. In our case, we set 100 nodes in the hidden layer in an LVQ, i.e. $M = 100$, and as we described in Chapter 2 we have $N = 6$. So, each individual is a long string with the length $length\_W = 6 \times 100$; each string unit denotes a weight from the input layer to the hidden layer. For the convenience of description, let W^p represent the p-th individual, and let $SW^p(k)$, $k = 1, 2, ..., n$, denote the k-th unit value in W^p.

2. Population. Population is grouped into families for the purpose of assigning some common attribute for the members in the same family. The attributes are different from one family to another. As in biology the members of the same family may have

some characteristics the same, such as the color of their skin, for our problem we let all members in a family have the same value range. Then different families haves different ranges of weight values.

In order to make the problem simple, let us have 10 families, and let each of the families have 45 members. The population size is thus 450. A number of studies suggest that a population size must be of the order of the individual string length. In our case, for the individual of length 600 ($length\_W = 6 \times 100$), the population size set of 450 seems to be appropriate [28]. Crossover and mutation occurs between the people in the same family, and sometimes occur between different families in order to create new individuals.

3. Fitness. Each individual in each family represents a weight vector, but it also represents a classifier, since if we use it under the assumption that it is the final weight, then we get an LVQ with specific classification properties. So we use each individual to alter the LVQ, and then test the whole training data set to determine the fitness of the selected weight vector. The fitness can be calculated as

$$Fitness = \frac{the\ number\ of\ correct\ classifications}{the\ size\ of\ the\ whole\ training\ set} \tag{4.2}$$

Because in the whole search process, the size of the training data set does not change, we can also use "the number of correct classifications" as the "Fitness".

4. Production of the next generation. This is a complex process that includes a copy of the best individual, random selection of better individuals, and production of new offsprings by applying crossover and mutation between the members in the same family, and sometimes applying crossover between members of different families.

At first, for each family, the members are ranked according to their fitness. The best one will be put on the top, the worst on the bottom. Then they are divided, in our case, into 5 groups in the proper order. As shown in Fig. 4-5, in the i-th family and the j-th family, we divide the members into A, B, C, D, E (five) groups, so that the fitness of members in group A are better than that in group B; the fitness of members in group

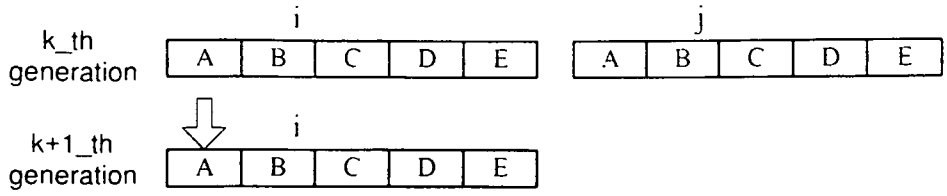

Figure 4-5: Production of the next generation.

B are better than that in group C, and so on. We apply the same process for the all families at each generation (i.e. search epoch).

In order to explain the process of production of the new generation, let us assume that N_A, N_B, N_C, N_D, N_E represent the number of members in these 5 groups. Then the process is as follows:

Step1. Group A of the $(k+1)$-th generation is directly copied from group A of the k-th generation (see Fig. 4-5);

Step2. Members in group B in the $(k+1)$-th generation are randomly selected from $A \cup B$ of the k-th generation. Here the routlette wheel selection or other types of selection can be used.

Step3. N_C new individuals are randomly generated to be placed into group C. This process can replenish individuals in the search space according to

$$SW^i(k) \in [-0.5, 0.5],\ k = 1, ..., length\_W,\ i = 1, ..., N_C$$

Step4. Randomly select N_D members from A, B, C, D in the k-th generation to generate new D group,

$$W^i \in \{A, B, C, D\},\ i = 1, ..., N_D$$

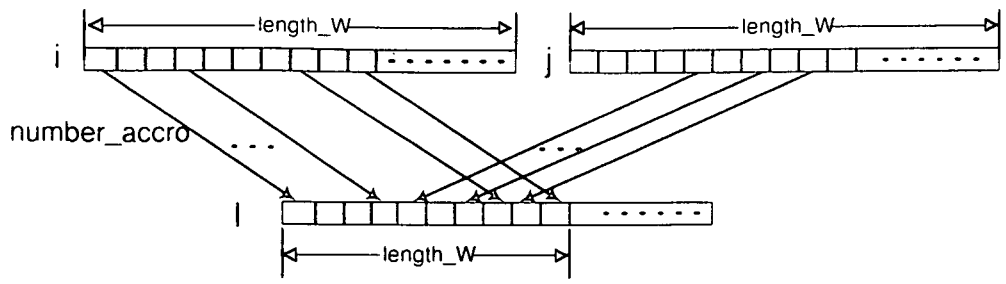

Figure 4-6: Crossover for group E.

and then make small mutations. There are three parameters of mutation: the number of set-points to be mutated, their position and the mutation strength. Of course mutation can occur in any unit of an individual, and the number of occurring mutation points can be fixed as a constant, such as $5, 10$, etc. But in practice, we do not know where the good point for mutation in an individual is, so the units are selected randomly. The number of units is denoted by $number\_k \in [1, 0.5 * length\_W]$; the mutation strength is also a random value $r \in [-0.2, 0.2]$ which is different for every mutation point;

$$\begin{aligned} SW^i(k) &= (1+r) * SW^i(k), \\ i &= 1...N_D, \\ k &\in [1,\ length\_W], \\ number\_k &\in [1, 0.5 * length\_W] \end{aligned}$$

Step5. Randomly select two members from groups A, B, C, D. Then the crossover points in the two strings are randomly decided, as well as the number of crossover points represented by $number\_accro \in [1, 0.2 * length\_W]$ (see Fig. 4-6). The member with high fitness has more opportunity to be selected. This process should be repeated N_E times.

This can be symbolically presented by

$$SW^l(k) = SW^i(p) \circ SW^j(q), l = 1...N_E,$$
$$SW^i, SW^j \in [A, B, C, D, E],\ k, p, q \in [1,\ length\_W]$$

where $\circ$ is the crossover operator. After every 5 generations, we select one parent from $[A, B, C, D, E]$ in the i-th family. Its mate is chosen from $[A, B, C, D, E]$ in the j-th family. The same crossover method in Fig. 4-6 is applied to generate N_E new offsprings to form group E for the $(k+1)$-th generation.

4.2.2 *GLVQ* for Classification

Searching

As we described in Chapter 2, each sequence that potentially contains a $TATA$ motif is described as a vector X after initial data processing. After that, the first statistical filter is applied with a threshold from the PWM matching score and restricted to the data regions. After this primary selection of sequences, the potential $TATA$ -containing sequences are normalized and transformedby PCA, so that at the end they are represented by feature vectors of length 6. This data is then ready for presentation as the LVQ inputs. Note that it is not necessary to filter data by the first statistical filter but when it is, it reduces considerably the negative data set.

In the section below, an LVQ is constructed with a fixed number of nodes in the input layer and the hidden layer. After that, a GA is used to search for the good initial LVQ weights.The training sets are $pAtr$ and S_{cds}. The procedure of GA searching is presented in Fig. 4-4. After a good initial weight vector is obtained, it is applied to the LVQ which is at that point considered as trained, and then the classification performance of both training sets and test sets were tested. We will discuss two cases with the threshold $\tau = 0.4$ and $\tau = 0.2$.

Case1: The threshold is set at $\tau = 0.4$, and the data region at $D_1 = [0.1, 1.3]$,

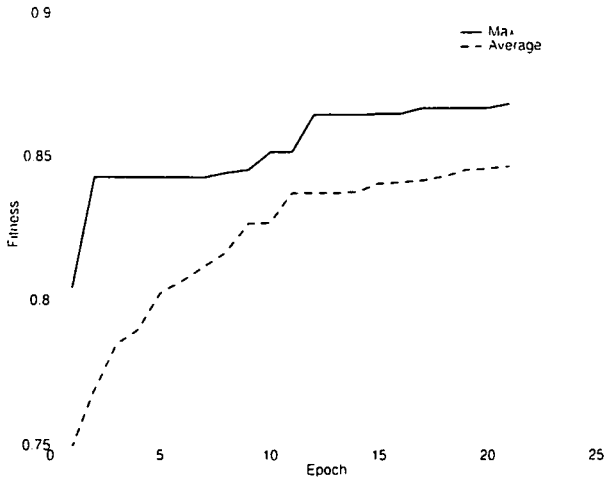

Figure 4-7: Fitness evolution during GA training when $\tau = 0.4$.

$D_2 = [0.1, 1.2]$. After this, the first statistical filter is applied to the training data. After filtering 493 feature vectors corresponding to sequences from $pAtr$ and 1359 corresponding to sequences from S_{cds} are obtained. The LVQ is constructed with 100 nodes in the hidden layer, where 40 are related to $TATA$ containing motifs and 60 related to $TATA$-less motifs. We apply GA to search for good weights for 21 epochs. The fitness curve is shown in Fig. 4-7, where the **Max** means the best solution at each epoch, and **Average** is the average fitness of the best solution in every family.

Although the fitness is still rising after 21 epochs, it reaches about 87%. This is relatively good, and since it took a considerable time to obtain such a fitness, we stopped the run The best obtained weights were applied to the LVQ and then the recognition performance of the LVQ tested. The results are shown in Table 4.1. The LVQ is fine tuned by the lvq2 algorithm. $LR = 0.01$ and 5500 epochs were used. The results are much better on the training set, and a bit better on the test set as indicated in Table 4.2. For comparison only, we present recognition results obtained in Chapter 3 with an LVQ having 500 nodes in the hidden layer, where the same threshold is applied and where the initial weights were obtained by random selection. The results based on GA are obviously

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.4	67.5	2.4	0.6586	58.8235	2.875	0.4525

Table 4.1: Case 1. Results without fine tuning of weights

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.4	66.8750	1.2125	0.7198	58.8235	2.55	0.4712

Table 4.2: Case 1: Results after fine tuning with lvq2 algorithm

better. As one can observe, by finding good initial weights by means of GA, we obtained relatively good solution with a simpler LVQ. Note that in these and other tables unless specified differently, $TP\%$ shows the percentage of promoters recognized based on the recognition of the recognized $TATA$ motifs.

Case 2: This time, the threshold is set to $\tau = 0.2$, while the other conditions are kept the same as in Case1. After statistical filtering 585 feature vectors from $pAtr$ and 4772 feature vectors from S_{cds} were obtained and were used as the training set for the LVQ and GA search. The evolution of fitness during the GA training is depicted in Fig. 4-8. After 51 epochs the GA training is stopped due to excessive time. A fitness of 81% was reached at that time, which is relatively good. The best weight vector is chosen and applied to the LVQ. The recognition ability of such an LVQ is then evaluated and the results are shown in Table 4.4. After that, the LVQ is fine tuned by means of the lvq2 algorithm, with the learning rate $LR = 0.01$. The recognition results with the tuned LVQ are given in Table 4.5. One can notice that the CC for the training sets has improved from 0.4712% to 0.6316%, and CC for the test sets from 0.2225% to 0.3437%. However, these are still not sufficiently good results.

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.4	61.2500	1.2125	0.6805	52.5210	2.3625	0.4388

Table 4.3: Results with the LVQ from Chapter 3

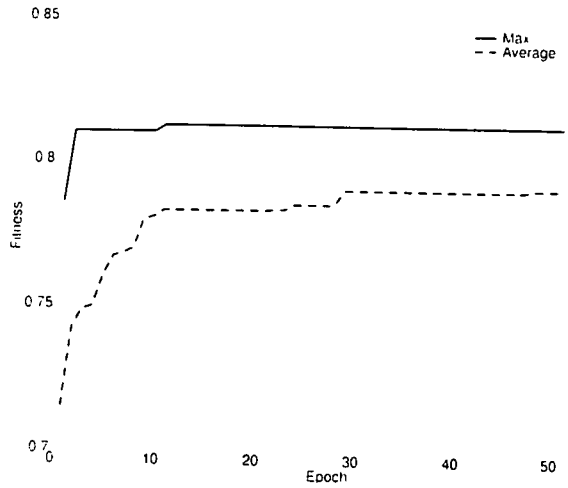

Figure 4-8: Fitness evolution during GA training when $\tau = 0.2$.

$Threshold$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
0.2	71.5625	9.3	0.4712	60.084	13.2125	0.2225

Table 4.4: Case2: Recognition results without fine tuning of weights in the LVQ

$Epoch$	$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	CC_{train}	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
500	73.7500	4.0375	0.6316	65.9664	7.2875	0.3437

Table 4.5: Case2: Recognition with fine tuned LVQ.

Discussion

1. In general, for any classifier the performance on the test set is usually not as good as the performance on the training set. As we know, in the competitive layer of an LVQ, for our problem the nodes are divided into two subclasses. One relates to $TATA$ motifs and will be represented by the set Γ, while the other relates to $TATA$-less motifs and will be represented by the set Ψ. Then the input signals to the hidden layer nodes for these two groups of nodes are

$$\begin{aligned} T_\Gamma &= \{-\|X - W^i\|,\ i \in \Gamma\}, \\ T_\Psi &= \{-\|X - W^i\|,\ i \in \Psi\}. \end{aligned}$$

If $Max(T_\Gamma) > Max(T_\Psi)$, then the LVQ input X can be regarded as containing a $TATA$ motif, otherwise it is regarded as $TATA$-less motif. For the hidden layer in an LVQ, the compete rule is that the winner has the opportunity to output the signal. Let us assume that one node in the $TATA$ motif subclass wins, but maybe it wins with a very small margin. Therefore, in order to reduce the level of FP, we may consider it as a $TATA$-less motif rather than a $TATA$ motif. Thus, we can specify an additional threshold r that modifies the classical compete rule so that X is considered as containing a $TATA$ motif only when

$$Max(T_\Gamma) - Max(T_\Psi) > r, \quad r > 0. \tag{4.3}$$

Note that this condition is applied only to the trained LVQ, during the recall (i.e. testing). This additional condition will reduce FP, but it will also reduce TP. Thus, it is important to select r carefully to balance TP and FP to get a relatively good result. Table 4.6 shows the effects of changing r. These results are applied for Case2.

r	$pAtr\,(TP\%)$	$S_{rds}\,(FP\%)$	CC_{train}	$pAtst\,(TP\%)$	$I_{rst}\,(FP\%)$	CC_{test}
0	71.5625	9.3000	0.4712	60.0840	13.2125	0.2225
0.2	65.3125	7.4375	0.4711	56.3025	11.0125	0.2308
0.4	**61.2500**	**6.1250**	**0.4768**	**53.7815**	**8.6875**	**0.2519**
0.5	58.2812	5.6000	0.4705	49.5798	7.5875	0.2483
1	48.5938	3.5625	0.4654	40.3361	0.4654	0.2804
2	23.5938	0.5500	0.4063	13.4454	0.5750	0.2226

Table 4.6: Case2: The effect of r.

As r increases, TP and FP fall as expected. With $r = 0.4$, CC becomes improved for training sets and test sets. But as r continues to increase further, CC worsens. So, r can not be regulated infinitely.

2.Deciding the number of nodes in the hidden layer and arranging their distribution according to subclasses are important problems in LVQ design. For Case1 and Case2, the only difference was the threshold used. However, it leads to a different distribution of the input data fed to LVQ. The lower threshold in Case2 makes data distribute in a wider and a more complex space, which can be suspected to be from the worse CC values than in Case1. It may be that more neurons in the hidden layer are required in Case2. Also, the larger number of filtered examples used in the GA search, makes it more time consuming. In Case1 it required only 21 epochs to reach 87% fitness, while in Case2 only 81% fitness is achieved with 51 epochs. This indicates that the size and quality of input data, as well as the complexity of the LVQ are very important in the proper classifier design.

4.3 Multistage $GLVQ$ for TATA-box recognition

4.3.1 Structure of the multistage $GLVQ$ system

In the last section, we used GA to search for the initial weights of LVQ. While GAs

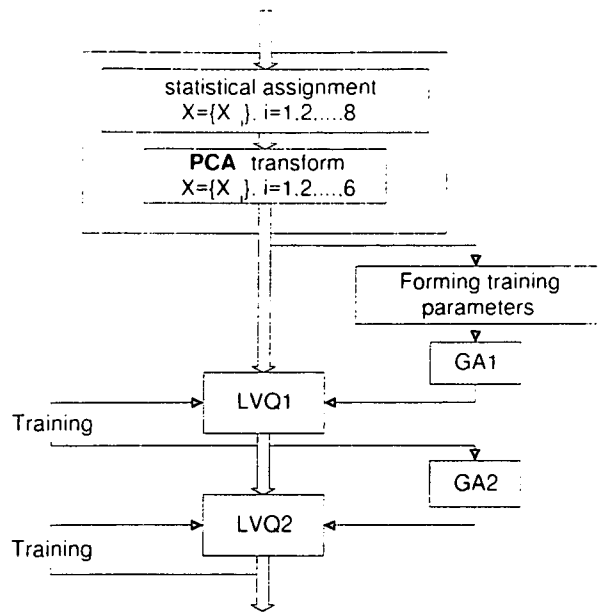

Figure 4-9: Structure of the multistage $GLVQ$ for TATA-box recognition.

can reach the global optimal solution, they are usually slow in the optimization. On the other hand, learning algorithms for ANNs are usually fast as most of them are gradient based. If the initial values of optimization parameters are close to the solution, then gradient based algorithms would find the solution fast. With this in mind we combined both types of algorithms in the search for good LVQ weights.

As we discussed in the last section, the distribution of data greatly affects the recognition quality of the system. We also observed that the overall recognition quality of the LVQ system with the statistical pre-filtering was not sufficiently good. Thus, we will attempt to make a 2-stage LVQ system (which we will name multistage $GLVQ$) with the idea to use the first stage LVQ as a rough filter, and the second stage LVQ as a fine filter. As shown in Fig. 4-9, the multistage $GLVQ$ consists of 2 stages denoted by $LVQ1$ and $LVQ2$. The raw data is fed to $LVQ1$, and the outputs of $LVQ1$ become the inputs of $LVQ2$. In each stage, an associated LVQ has the same number of nodes in the hidden layer. However the distribution of the nodes is a bit different which relays on the class distributions. In the training procedure, genetic algorithm $GA1$ is used to search

for the solution for $LVQ1$, and the initial weights for $LVQ2$ are found by $GA2$. Each of the $LVQs$ is further fine tuned by the lvq2 algorithm.

The training set for multistage $GLVQ$ is obtained from $pAtr \cup S_{rds}$, while the test set is obtained from $pAtst \cup S_{tst}$. The initial feature vectors of dimension 8 are filtered with a second statistical filter (the threshold $\tau = 0.4$, and bounds for data regions for D_1 and D_2 selected as $D_1 = [0.1, 1.3]$, $D_2 = [0.1, 1.2]$). This second statistical filter (as opposed to the first one, see Chapter 2) will let more sequences pass through than if the first filter is applied. However, we want to train the multistage $GLVQ$ on larger sets of sequences, so that it captures more of the characteristics that may be influential in recognition problem. The feature vectors that passed this filter were subject to normalization and PCA transformation as described in chapter 2. Consequently, each sequence after second statistical filtering is represented by a vector X which has dimension 6, as explained in Chapter 2.

4.3.2 Multistage $GLVQ$ training and test

The training procedure for the multistage $GLVQ$ network actually consists of two steps for training each of the two $LVQs$. For each LVQ, GA is primarily used to find an initial weight vector; thereafter the lvq2 algorithm is used for fine tuning the LVQ. The training procedure and testing is described in what follows.

1) The first stage LVQ1

There are 100 nodes in the hidden layer in the first stage $LVQ1$, and the proportion of neurons in the hidden subclasses layer selected as $[0.4, 0.6]$, i.e. 40% of the neurons relate to Class 1($TATA$ motif class) and 60% of the neurons relate to Class 2 ($TATA$-less motif class). First we apply the GA to search for the initial weight vector of $LVQ1$. The method and parameters of the GA are described in the previous section. In order to speed up the search by GA, 640 feature vectors from sequences from $pAtr$ are used as well as 640 feature vectors from sequences from S_{rds}. Data is not filtered by either threshold for the matching score nor by the value region restrictions. After 51 generations, the fitness reaches 85.03%. The fitness evolution is shown in Fig. 4-10. As can be seen, the fitness

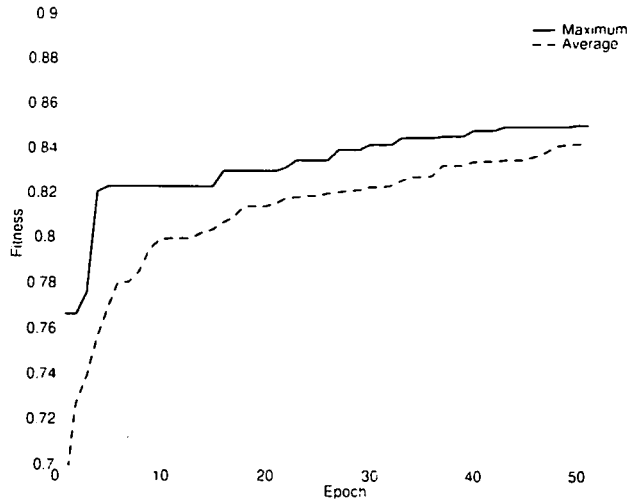

Figure 4-10: Fitness evolution for $LVQ1$.

$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
79.3750	11.9500	0.4707	74.3697	14.4125	0.27

Table 4.7: Results with LVQ1 without fine tuning.

has still not reached saturation, but we already have a rather good initial weight vector. The performance of $LVQ1$ with such obtained weights is summarized in Table 4.7. The results with the fine tuned $LVQ1$, where the lvq2 algorithm is used with the learning rate $LR = 0.01$ and with 500 epochs, are shown in Table 4.8. In the tests, the feature vectors correspond to the complete sets $pAtr$, $pAtst$, S_{cds}, and S_{int}. Thus the TP and FP relate to the whole training and test sets.

2) The second stage LVQ2

We applied feature vectors corresponding to each sequences from $pAtr$ (640 of them) and all feature vectors that correspond to every sequence from S_{cds} (8000 of them) as

$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
76.4063	5.75	0.5918	69.3277	8.1125	0.3436

Table 4.8: Recognition results with fine tuned LVQ1.

$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
56.25	0.8000	0.6722	48.3193	1.4375	0.4767

Table 4.9: Results with LVQ2 which is not fine tuned.

$pAtr\,(TP\%)$	$S_{cds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{int}\,(FP\%)$	CC_{test}
58.1250	0.7625	0.6885	51.6807	0.6885	0.5377

Table 4.10: Results with fine tuned LVQ2.

inputs to the fine tuned $LVQ1$. Only 489 feature vectors related to $pAtr$ passed through ($TP = 76.4063\%$), and 460 feature vectors related to S_{cds} ($FP = 5.75\%$) (see Table 4.8). This data is used for training of the $LVQ2$ network. Since there is a very similar number of positive and negative examples (489 and 460), we set up the distribution of neurons in the hidden layer of $LVQ2$ regulated as $[0.5, 0.5]$, i.e. 50% of all neurons will be associated with Class1 and the other 50% of neurons with Class2. The number of neurons in the hidden layer is 100. Then, $GA2$ is used to search for the initial weights of $LVQ2$. After 101 epochs the obtained fitness was 79.66%, while the average fitness of the last generation was 77.24%. The fitness evolution is depicted in Fig. 4-11. The results of the $LVQ2$ recognition with the best weight vector applied is summarized in Table 4.9. Compared with $LVQ1$, FP for S_{cds} reduces 7 times, FP for S_{int} reduces 6 times, while TP for the training set and the test set fall about 30%. The improvement obtained is thus significant. The $LVQ2$ is fine tuned with the lvq2 algorithm in 7500 epochs and with the learning rate $LR = 0.01$. gives recognition results given in Table 4.10. Compared with the results obtained in Chapter 3, one can observe that the results obtained with the fine tuned $LVQ2$ are far better. It should be indicated that the results for $LVQ2$ are obtained by passing feature vectors first through the second statistical filter, then making the normalization and the PCA transformation of the filtered data, then passing them through the $LVQ1$, and finally, passing this filtered data through $LVQ2$.

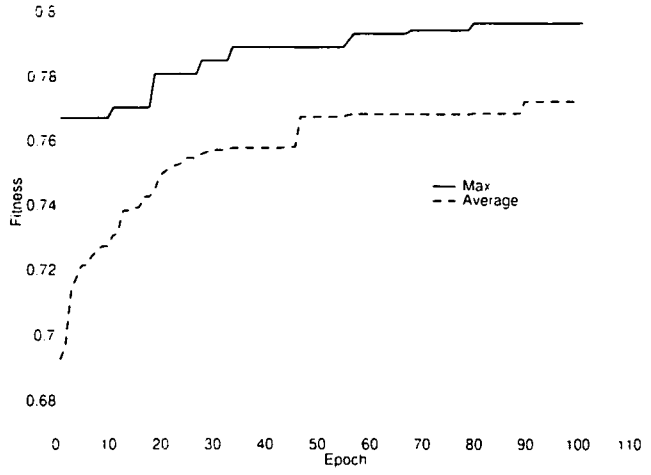

Figure 4-11: Fitness evolution for $LVQ2$.

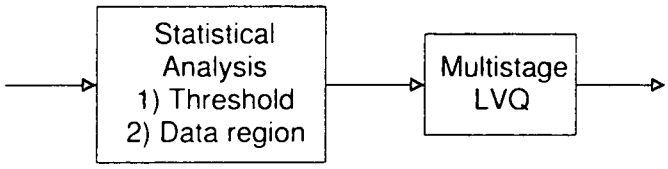

Figure 4-12: Combined multistage $GLVQ$.

4.3.3 The final solution: Combined multistage $GLVQ$ and statistical analysis

We will use the tuning made in the previous section and combine this with the statistical analysis made in Chapter 2, in order to obtain the most efficient $TATA$ motif recognition system. The structure of our final classifier is depicted in Fig. 4-12. Initially, the data is filtered by the first statistical filter (as defined in Chapter 2). Then, the filtered data is normalized and PCA transformed as described in Chapter 3, producing the feature vectors of size 6. Then, these feature vectors are fed to the input of the multistage $GLVQ$ trained in the previous section. We will examine only the influence of the threshold in the first statistical filter. The results are summarized in Table 4.11. Quite good recognition quality has been achieved. The best results for the test set are

$Threshold$	$pAtr\,(TP\%)$	$S_{tds}\,(FP\%)$	$CC_{training}$	$pAtst\,(TP\%)$	$S_{tst}\,(FP\%)$	CC_{test}
0.3600	57.9688	0.7625	0.6874	50.8403	1.0875	0.5312
0.3700	57.9688	0.7625	0.6874	50.8403	1.0875	0.5312
0.3800	57.9688	0.7625	0.6874	50.8403	1.0875	0.5312
0.3900	57.9688	0.7625	0.6874	50.8403	1.0875	0.5312
0.4000	57.9688	0.7125	0.6911	50.8403	1.0000	0.5411
0.4100	**57.6563**	**0.6625**	**0.6925**	50.8403	1.0000	0.5411
0.4200	57.5000	0.6625	0.6914	50.8403	0.9250	0.5500
0.4300	57.0313	0.6125	0.6918	**50.8403**	**0.9125**	**0.5515**
0.4400	56.2500	0.6000	0.6871	49.1597	0.9125	0.5384
0.4500	54.6875	0.5875	0.6767	47.4790	0.7625	0.5442
0.4600	52.6563	0.5875	0.6617	43.6975	0.6750	0.5254
0.4700	49.6875	0.5875	0.6393	39.4958	0.6625	0.4913
0.4800	46.2500	0.5000	0.6197	35.7143	0.4750	0.4867
0.4900	41.2500	0.4625	0.5825	30.6723	0.3000	0.4716
0.5000	36.2500	0.4625	0.5396	23.9496	0.1375	0.4408

Table 4.11: Results for the complete system. The effects of threshold.

obtained with $\tau = 0.43$ ($C_{test} = 0.5515$ with $C_{train} = 0.6918$). The best results for the training set were obtained with $\tau = 0.41$ ($C_{train} = 0.6925$ with $C_{test} = 0.5411$). We notice that there is not much difference between the two best cases, which indicates that this structure of system is trained rather well and that it generalizes reasonably well. When the single-stage LVQ is applied (results from Chapter 3), although we achieved higher CC for the training sets at the level around 0.75, we unfortunately obtained too low a CC for the test set (0.4297). Thus the advantage of the multistage GLVQ is obvious.

4.3.4 The final test

Let us summarize our results up to now. We have developed a new model of $TATA$ motifs. We attempt to recognize promoter sequences based on the recognition of the $TATA$ motif. However, all tests made so far are only on the short segments of DNA. We want to evaluate our recognition model in a more realistic situation. For this reason we use the independent test set of sequences selected by Fickett and Hatzigeorgiou [32]

that comprises 18 mammalian sequences of total length of 33120 bp which contained 24 promoters.

The experiment is made as follows. First, windows of length 43 nucleotides slide along the sequence from its 5′ toward its 3′ end. The content of the window is analyzed as explained in Chapter 2, and the feature vector of length 8 (see Chapter 2) is formed for each of the window positions. Feature vectors are filtered by the first statistical filter (see Chapter 2) using the threshold $\tau = 0.43$ (and for the other set of experiments $\tau = 0.41$). The data that passes through is normalized and PCA- transformed making new feature vectors of length 6 that enter the multistage $GLVQ$. The output of the system then indicates if the input data corresponds to the hypothetical $TATA$ motif or not. Since the $TATA$ motifs are very regularly distanced from the TSS, we add the mean distance (from the TSS) of the $TATA$ motif (30 bp) to position where the hypothetical $TATA$ motif is detected by the system, and this position will represent the predicted TSS. Such analysis is also made for the reverse complement strand of the examined sequence. According to [32], the predicted TSS was counted as correct if it was within $[-200, 100]$ of any experimentally mapped TSS. The recognition results with our system are given in Table 4.12. The '+' sign denotes positions that indicate the correct prediction of the TSS.

The results are as follows. with the threshold $\tau = 0.41$ (which corresponds to the best results on the training set) the system identified 8 of the 24 known TSSs ($TP = 33\%$) and made 47 false positives (one false prediction for 705 bp denoted as 1/705 bp). When the threshold was $\tau = 0.43$ (corresponding to the best results on the test set) then 7 of the 24 TSSs were recognized making 41 false predictions ($1/808\,bp$). In Table 4.13 we present the results achieved by several other programs on the same test set.

Since it is difficult to compare the overall performance of different programs since they produce different levels of TP and FP, we will use the average score measure

	Seq. name	TSS location		
No.	(*Length in bp*)	(*# of TSS*)	$\tau = 0.41$	$\tau = 0.43$
1	*L*47615(3321)	[2078, 2108] (1)	2510, *r*866, *r*2018	2510, *r*866, *r*2018
2	*U*54701(5663)	[935], [2002] (2)	119, 228, 3118, 5369, *r*45, *r*1392	119, 228, 3118, 5369, *r*45
3	*Chu*(2003)	[1483, 1554] [1756, 1783] (2)	246, 1487(+)	246, 1487(+)
4	*U*10577(1649)	[1169, 1171] [*r*1040, *r*1045] (2)	354	354
5	*U*30245 (1093)	[850, 961] (1)	194, 446, 690(+), *r*276, *r*385	446, 690(+), *r*276, *r*385
6	*U*69634 (1515)	[1450] (1)	299, 934, *r*871	299, 934, *r*871
7	*U*29912 (565)	[143, 166] (1)	None	None
8	*U*29927 (2562)	[738, 803] [1553, 1717] (2)	332, 1532(+), 2114, *r*905	332, 1532(+), 2114
9	*Y*10100 (1066)	[1018, 1033] (1)	159, 781, *r*401, *r*718, *r*886	159, 781, *r*718, *r*886
10	*Nomoto* (2191)	[1793, 1812] (1)	486, 867, 1191, 1375, 1643(+)	486, 867, 1191, 1375, 1643(+)
11	*U*75286 (1984)	[1416, 1480] (1)	*r*1122	None
12	*U*52432 (1604)	[1512, 1523] (1)	217, 1380(+), *r*154, *r*1467	217, *r*154, *r*1467
13	*U*80601 (632)	[317, 400] (1)	94	94
14	*X*94563 (2693)	[1163, 1200] (1)	489, 2527, *r*517, *r*2466	489, 2527, *r*2466
15	*Z*49978 (1352)	[855], [1020] [1150] (3)	1011(+), 1024(+), *r*193, *r*231, *r*948	1011(+), *r*193, *r*231, *r*948
16	*U*49855 (682)	[28, 51] (1)	*r*521	*r*521
17	*X*75410 (918)	[815, 835, 836] (1)	473, *r*398, 685(+) 732(+), 786(+)	473, *r*398, 685(+) 732(+), 786(+)
18	*U*24240 (1728)	[1480] (1)	*r*200, *r*658	*r*200, *r*658

Table 4.12: Results of recognition on the Fickett and Hatzigeorgiou test set

No.	Programs	TP		FP	
1	Audic [2]	5	24%	33	$1/1004bp$
2	Autogene [67]	7	29%	51	$1/649bp$
3	Promoter2.0	10	42%	43	$1/770bp$
4	NNPP	13	54%	72	$1/460bp$
5	PromoterFind [58]	7	29%	29	$1/1142bp$
6	PromoterScan	3	13%	6	$1/5520bp$
7	TATA [15]	6	25%	47	$1/705bp$
8	TSSG [116]	7	29%	25	$1/1325bp$
9	TSSW [116]	10	42%	42	$1/789bp$
10	SPANN1 [?]	12	50%	44	$1/752\ bp$
11	SPANN2 [8]	8	33%	16	$1/2070bp$
12	our system ($\tau = 0.41$)	8	33%	47	$1/705bp$
13	our system ($\tau = 0.43$)	7	29%	41	$1/808bp$

Table 4.13: Results of different programs obtained on the test set of Fickett and Hatzigeorgiou.

(ASM) introduced recently in [6] to make the final ranking of program performances, and in this way to see how our prediction system compares to others. The results are given in Table 4.14. The best system was assigned the position 1, second best 2, etc. We can observe that our system (in both versions with $\tau = 0.41$ and $\tau = 0.43$) performs better than three other programs: Audic, Autogene, and TATA. Since the performance of the program denoted as 'TATA' is the recognition based only on the PWM matching score, we see that we have achieved the goal of developing a more efficient model for recognition of the $TATA$ motifs.

#	Program name	Ranking pos.
1	Audic [2]	9
2	Autogene [67]	10
3	Promoter2.0	4
4	NNPP	6
5	PromoterFind [58]	5
6	PromoterScan	2
7	PWM of TATA-box [15]	11
8	TSSG [116]	1
9	TSSW [116]	3
10	MGLVQ ($\tau = 0.41$)	8
11	MGLVQ ($\tau = 0.43$)	7

Table 4.14: Ranking different programs based on the results of predictions achieved on the Ficket and Hatizigoergiou data set

Chapter 5

Conclusion

In this research we focused on modeling $TATA$ box motifs from eukaryotic promoters. Our new model was based on a combined statistical analysis and neural network modelling. The model that we obtained appears to have very good sensitivity and good specificity. It outperforms three other programs in recognition of promoters (although it is designed to recognize the $TATA$ motifs and not promoters themselves). This demonstrates that the methodology applied and ideas used were good.

The essence of our model was discrimination of the core $TATA$ hexamer and its immediate neighboring hexamers before the core motif and after the core motif. These discrimination characteristics were derived on the basis of modified EIIP values, as described in Chapter 2. Moreover, we incorporated into the model the positional characteristics of $TATA$ motifs. This was combined with the filtering of feature vectors by threshold restrictions relative to the PWM matching score of the $TATA$ motifs, and subsequent normalization and PCA transformation of feature vectors, that resulted in dimension reduction of the new feature vectors. Such processed data is further analyzed by a multistage LVQ system that contained two LVQ networks. These networks have been initially trained by means of genetic algorithms in the attempt to find good initial weights for them. After these are found the $LVQs$ were further fine tuned by means of the lvq2 learning algorithm.

As an efficient Neural network classifier, LVQ is successfully used on promoter recognition in this study. Combined with statistical analysis, including matching scores of PWM, average values of segments around $TATA$-box and position model, the recognition quality is apparently improved. Based on studying LVQ parameters in detail, it is shown that the values of the learning rate, the number of the nodes in the hidden layer, the number of iterations and the initial weights should be carefully selected. The training data for LVQ is a crucial factor, especially on promoter recognition, how to construct the input data to be fed to LVQ for the training and test set is a basic and influential problem for Neural networks on promoter recognition.

GA has been applied widely in searching global optimal solutions. Recognition quality is improved with optimized initial weights found by GA. Based on genetic algorithms, a multistage $GLVQ$ is proposed which sufficiently utilizes the information contained in the feature vectors. This system achieves a rather good recognition quality, better than some other reported results. The techniques and the method proposed may serve as a basis for development of recognition programs based on other short motifs contained in DNA sequences.

It should be mentioned that the basic ideas and results obtained in this study will appear in H. Wang, X. Li, V. B. Bajić, Neural-Statistical Model of TATA-box motifs in Eukaryotes, Chapter 2 in *Bioinformatics and Biocomputing*, to be published by Physica-Verlag, New York, 2001.

Bibliography

[1] Andrade M.A., Sander C. (1997), Bioinformatics: from genome data to biological knowledge, *Biotechnology*, **8**:675-683

[2] Audic S., Claverie J. -M. (1997) Detection of eukaryotic promoters using Markov transition matrices, *Computer & Chemistry*, **21**(4):223-227

[3] Baldi , Brunak (1998)

[4] Balakrishnan P.V., Cooper M.C., Jacob V.S., Lewis, P.A. (1994) A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering. *Psychometrika*, **59**: 509-525.

[5] Bajić I. V. (1998) Digital Signal Processing Techniques in the Analysis of DNA/RNA and Protein Sequences, BScEng Thesis, University of Natal, Durban, South Africa

[6] Bajić V. B. (2000) Comparing the success of different prediction software in sequence analysis: A review, *Briefings in Bioinformatics*, **1**(3)

[7] Bajić V. B., Bajić I. V. (1999) ANN in DNA regulatory region recognitions: The case of promoters, Tutorial, CD edition, International Joint Conference on Neural Networks, Washington, DC, USA, July 10-16, 146 pages

[8] Bajić V. B., Bajić I. V. (2000) Neural network system for promoter recognition, Chapter in *Future Directions for Intelligent Systems and Information Science* (Nik Kasabov, Ed.), Physica-Verlag, New York

[9] Bajić V. B., Bajić I. V. (2001) How Neural Networks Find Promoters Using the Recognition of Micro-Structural Promoter Components, Chapter in *Bioinformatics and Biocomputing* (V Bajić, Ed.), Physica-Verlag, New York

[10] Beasley D., Bull D.R., Martin R.R.(1993). An overview of genetic algorithms: Part 1, fundamentals, *University Computing* **15**(2), ftp://ralph.cs.cf.ac.uk/pub.papers/GAs/ga overview1.ps

[11] Benoist C., O'Hare K., Breathnach R., Chambon P. (1980) The ovalbumin gene - sequence of putative control regions, *Nucl. Acids Res.*, **8**:127-142

[12] Bose N. K., Liang P. (1996) *Neural Network Fundamentals with Graphs, Algorithms, and Applications*, McGraw-Hill, Inc., New York

[13] Broomhead D., Lowe D. (1988) Multivariable function interpolation and adaptive networks, *Complex Systems*, **2**:321-355

[14] Brunak S. J., Engelbrecht J., Knudsen S. (1991) Prediction of human mRNA donor and acceptor sites from the DNA sequence, *J. Mol. Biol.*, **220**:49-65

[15] Bucher P. (1990) Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences, *J. Mol. Biol.*, **212**:563-578

[16] Bucher P., Trifonov E. N. (1986). Compilation and analysis of eukaryotic Pol II promoter sequences. *Nucl. Acids Res.* **14**:10009-10026

[17] Burge C., Karlin S. (1997) Prediction of complete gene structures in human genomic DNA, *J. Mol. Biol.*, **268**:78-94

[18] Burset M., Guigo R. (1996) Evaluation of gene structure prediction programs. *Genomics*, **34**:353-367

[19] Caudill M. (1989) *Neural Networks Primer*, Miller Freeman Publications, San Francisco, CA

[20] Chen Q., Hertz G. Z., Stormo G. D. (1995) MATRIX SEARCH 1.0: a computer program that scans DNA sequences for transcriptional elements using a database of weight matrices. *Computer Applic. Biosci.* **13**:29-35

[21] Chen C. H., Ed. (1996) *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill, Inc., New York

[22] Chen Q., Hertz G. N., Stormo G. D. (1997) PromFD 1.0: a computer program that predicts eukaryotic pol II promoters using strings and IMD matrices. *Computer Applic. Biosci.*, **13**: 29-35

[23] Cherkassky V., Mulier F. (1998) *Learning from data*, John Wiley & Sons, Inc.

[24] Claverie J. (1997) Computational methods for the identification of genes in vertebrate genomic sequences, *Human Molecular Genetics*, **6**(10) review issue:1735-1744

[25] Claverie J. M., Audic S. (1996) The statistical significance of nucleotide position-weight matrix matches. *Comput Applic. Biosci.* **12**:431-439

[26] Claverie J., Sauvaget I. (1985) Assessing the biological significance of primary structure consensus patterns using sequence databanks. I. Heat-shock and glucocorticoid control elements in eukaryotic promoters. *Computer Applic. Biosci.*, **1**:95-104

[27] Corden J., Wasylyk B., Buchwalder A., Sassone-Corsi P., Kedinger C., Chambon P. (1980) Promoter sequence of eukaryotic protein-coding genes, *Science*, **209**:1406-1414

[28] Deb, K.(1995) Optimization for engineering desigh: Algorithms and examples. Delhi: Prentice_Hall.

[29] Devroye L., Gyorfi L., Lugosi G. (1996), *A Probabilistic Theory of Pattern Recognition*, New York, Springer

[30] Fausett L. (1994) *Fundamentals of Neural Networks*, Prentice-Hall, Inc., Englwood Cliffs, NJ

[31] Fickett J. W. (1996) Finding genes by computer: The state of art, *Trends Genet.*, **12**:316-320

[32] Fickett J. W., Hatzigeorgiou A. G. (1997) Eukaryotic promoter recognition, *Genome Research*, **7**(9): 861-878

[33] Fickett J.W., Wasserman W.W. (2000) Discovery and modeling of transcriptional regulatory regions, *Biotechnology*, **11**:19-24

[34] Fogel D. B. (2000) *Evolutionary computation* (Second edition), IEEE Press, New York

[35] Fogarty T.C.(1989) Varying the probability of mutation in the genetic algorithm. In J.D. Schaffer, editor, *Proceedings of the Third*

[36] Frech K., Werner T. (1997) Specific modelling of regulatory units in DNA sequences. *Proceedings of the 1997 Pacific Symposium on Biocomputing*, World Scientific Publishing Co. Pty. Ltd., Singapore, 151-162

[37] Frech K., Herrmann G., Werner T. (1993) Computer-assisted prediction, classification, and delimitation of protein binding sites in nucleic acids. *Nucl. Acids Res.*, **21**:1655-1664

[38] Frech K., Dietze P., Werner T. (1997c) ConsInspector 3.0: new library and enhanced functionality. *Computer Applic. Biosci.*, **13**:109-110

[39] Goertzel B. (1995) A convergence theorem for the simple GA with population size tending to infinity, *Proc. 2nd IEEE Int'nl Conf. Evolutionary Comput.*, Vol. 1, Piscataway (NJ), IEEE Press.

[40] Goldberg, D.E. (1989). Genetic algorithms in search, optimization, and machine learning. New York: Addison-Wesley.

Goertzel95

[41] Gruau F. (1994). Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm.

[42] Guigo R. (1997) Computational gene identification: an open problem, *Computer & Chemistry*, **21**(4):215-222

[43] Guigo R., Knudsen S., Drake N., Schmith T. (1992) Prediction of gene structure. *J. Mol. Biol.* **226**:141-157

[44] Hahn S., Buratowski S., Sharp P. A., Guarente L.(1989) Yeast TATA-binding proteinTFIIDbinds to TATA elements with both consensus and nonconsensus sequences, *Proc. Natl. Acad. Sci. USA*, **86**:5718-5722

[45] Harr R., Haggstrom M., Gustafsson P. (1983) Search algorithm for pattern match analysis of nucleic acid sequences, *Nucl. Acids Res.*, **11**:2943-2957

[46] Hastie T., Stuetzle W. (1989) Principal curves. *Journal of the American Statistical Association,* **84**:502-516

[47] Hatzigeorgiou A. (2000) Mathematical models for feature recognition in nucleotide sequences, Dr. rer. nat. Dissertation, University of Jena, Germany

[48] Hatzigeorgiou A., Mache N., Wieland J., Reczko M., Zell A. (1994) Recognition of promoters and coding regions on eukaryotic sequences with neural networks, in *Bioinformatics* **94**:70-74

[49] Hatzigeorgiou A., Mache N., Reczko M. (1996) Functional site prediction of the DNA sequence by artificial neural networks, *Proc. IEEE Int. Joint Symposia on Intelligence and Systems*, 12-17

[50] Hatzigeorgiou A., Reczko M. (1995) Recognition of protein coding regions and reading frames in DNA using neural networks, *Proc. World Congress on Neural Networks*, INNS Press **3**:136-138

[51] Hatzigeorgiou A., Reczko M. (1996) Gene identification with neural networks, *Proc. Symp. Control, Optimization and Supervision* (CESA'96 IMACS Multiconference), **1** :140-143

[52] Hatzigeorgiou A., Harrer T., Mache N., Reczko M. (1995) The gene sequence analysis system diana, in *Bioinformatics: from Nucleic Acids and Proteins to Cell Metabolism* (D. Schomburg and U. Lessel, eds.), GBF Braunschweig, VCH

[53] Hayes W., Borodovsky M. (1998) How to interpret anonymous genome? Machine learning approach to gene identification, *Genome Research*, **8**:1154-1171

[54] Hecht-Nielsen R. (1990) *Neurocomputing*, Addison-Wesley, Reading, MA

[55] Hertz J., Krogh A., Palmer R. G. (1991) *Introduction to the Theory of Neural Computation*, Adison-Wesley, Redwood City, CA

[56] Holland, J.H.(1975) Adaptation in Natural and artificial system. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[57] Hussain B., Kabuka M. R. (1994) A novel feature recognition neural network and its application to character recognition, *IEEE Trans. Pattern Anal. Machine Intell.*, **16**:98-106

[58] Hutchinson G. B. (1996) The prediction of vertebrate promoter regions using differential hexamer frequency analysis. *Computer Applic. Biosci.*, **12**:391-398

[59] Janikow C. Z. (1993) A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, **13** (2-3):189-228

[60] Kawabe T., Tagami T., Katayama T. (1996). A Genetic alogorithm based minimax optimal design of robust I_PD controller, Proc. IEE Int. Conf. Control. London, U.K., 436-441

[61] Kelly J. D. Jr., Davis L. (1991). A hybrid genetic algorithm for classification. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence , 645-650.

[62] Knudsen S. (1999) Promoter2.0: for the recognition of Pol II promoter sequences, *Bioinformatics*, **15**(5):356-361

[63] Kohonen T. (1986) Learning Vector Quantization for Pattern Recognition, Technical Report TKK-F-A601, Helsinki University of Technology

[64] Kohonen T. (1988) Learning Vector Quantization. *Neural Networks*, **1**(Suppl 1):303

[65] Kohonen T. (1990) Improved Versions of Learning Vector Quantizations, *Proc. Int. Joint Conf. on Neural Networks*, I, 545-550

[66] Kohonen T. (1997), *Self-Organizing Maps*, Second ed., Berlin: Springer-Verlag

[67] Kondrakhin Y. V., Kel A. E., Kolchanov N. A., Romashchenko A. G., Milanesi L. (1995) Eukaryotic promoter recognition by binding sites for transcription factors. *Computer Applic. Biosci.*, **11**:477-488

[68] Koza J.R. (1991) A non-linear genetic algorithms for solving problems, Australian patent 611,350. Issued Sep. 21 1991)y([10]) ga:Koza91h

[69] Kung S. Y. (1993) *Digital Neural Networks*, PTR Prentice Hall, Engelwood Cliffs, NJ

[70] Lapedes A. S., Barnes C., Burks C., Farber R. M., Sirotkin K. M. (1990) Application of neural networks and other machine learning algorithms to DNA sequence analysis, in *Computers and DNA, SFI Studies in the Science of Complexity*, **VII**

[71] Mache N., Levi P. (1996) Detection of eukaryotic POL II promoters with multistate time-delay neural network, *Proc. of the German conference on Bioinformatics GCB'96*, IMISE Report No. 1, Inst. fuer Medizinische Informatik, Statistik und Epidemilogie, Leipzig, ISB 3-000000872-1

[72] Mache N., Reczko M., Hatzigeorgiou A., Multistate time-delay neural networks for the recognition of POL II promoter sequences, `http://www.informatik.uni-stuttgart.de/ipvr/bv/personen/mache`.

[73] Manderick B., de Weger M., Spiessens P. (1991) The genetic algorithm and the structure of the fitness landscape. Proceedings of the Fourth International Conference on Genetic Algorithms, 222-229. La Jolla, CA: Morgan Kaufmann.

[74] Matis S., Xu Y., Shah M., Guan X., Einstein J. R., Mural R., Uberbacher E. (1996) Detection of RNA polymerase II promoters and polyadenylation sites in human DNA sequence, *Computers Chem.*, **20**:135-140

[75] Milanesi L., Muselli M., Arrigo P. (1996) Hamming-Clustering method for signal prediction in 5′ and 3′ regions of eukaryotic genes, *Comput. Applic. Biosci.*, **12**:399-404

[76] Mitchell, M., & Holland, J. H. (1993). When will a genetic algorithm outperform a hill climbing? Proceedings of the Fifth International Conference on Genetic Algorithms. Forrest, S. (ed.), Morgan Kaufmann, 647

[77] Mitchell M., Forrest S., Holland J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and GA performance. In Proceedings of the First European Conference on Artificial Life. Cambridge, MA: MIT Press/Bradford Books.

[78] Miyahara K., Yoda F. (1996) Printed Japanese character recognition based on multiple modified LVQ Neural Network, *IEEE Press*, 0-7803-2566-4/96.

[79] Muhlenbein H., Voosen D.S. (1993) Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, **1**(1).

[80] Muhlenbein H., Schomish M., Born J. (1991) The Parallel Genetic Algorithm as Function Optimizer, *Parallel Computing* , **17**:619–632

[81] Murakami K., Takagi T. (1998) Gene recognition by combination of several gene-finding programs, *Bioinformatics*, **14**(8):665-675

[82] Murino V. (1998) Structured Neural Networks for Pattern Recognition, *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, **28**(4):553-561

[83] Nussinov R., Owens J., Maizel J. V. (1986) Sequence signals in eukaryotic upstream regions, *Biochim. Biophys. Acta*, **866**:109-119

[84] Ohler U., Reese M.G. (1998) Detection of eukaryotic promoter regions using stochastic language models. *Molekulare Bioinformatik*, 89-100.

[85] Ohler U., Harbeck S., Niemann H., Noth E., Reese M. G. (1999) Interpolated Markov chains for eukaryotic promoter recognition, *Bioinformatics*, **15**(5):362-369

[86] Patton A., Punch W., Goodman E. (1995). A standard ga approach to native protein conformation prediction. In Eshelman, L., editor, Proc. Sixth Int. Conf. Gen. Algo., **574**. Morgan Kaufmann.

[87] Pedersen A. G., Baldi P., Chauvin Y., Brunak S. (1999) The biology of eukaryotic promoter prediction - a review, *Computers & Chemistry*, **23**:191-207

[88] Penotii F. (1990) Human DNA TATA boxes and transcription initiation sites, *J. Mol. Biol.* **213**:37-52

[89] Perier R. C., Junier T., Bonnard C., Bucher P. (1999) The Eukaryotic Promoter Database (EPD): recent developments, *Nucl. Acids Res.*, **27**(1):307-309

[90] Pham and Liu (1995). Neural networks for identification, prediction and control. Springer-Verlag London Limited.

[91] Poggio T., Girosi F. (1990) Regularization algorithms for learning that are equivalent to multilayer networks, *Science*, **247**:978-982

[92] Prestridge D. S. (1991) SIGNAL SCAN: a computer program that scans DNA sequences for eukaryotic transcriptional elements. *Computer Applic. Biosci.* **7**:203-206

[93] Prestridge D. S. (1995) Predicting Pol II promoter sequences using transcription factor binding sites, *J. Mol. Biol.*, **249**:923-932

[94] Prestridge D. S. (1996) SIGNAL SCAN 4.0: Additional databases and sequence formats, *Computer Applic. Biosci.*, **12**:157-160

[95] Prestridge D. S. (1999) Computer software for eukaryotic promoter analysis, (published on internet). `http://biosci.umn.edu/class/bioc/8140/Promoter.html`

[96] Prestridge D. S., Burks C. (1993) The Density of transcriptional elements in promoter and non-promoter sequences, *Hum. Mol. Genet.*, **2**:1449-1453

[97] Prestridge D. S., Stormo G. (1993) SIGNAL SCAN 3.0: New database and program features, *Computer Applic. Biosci.*, **9**:113-115

[98] *Proceedings of the IEEE,* Special Issue on Neural Networks, I: Theory and Modeling, **78**(9)

[99] *Proceedings of the IEEE,* Special Issue on Neural Networks, II: Analysis, Techniques and Applications,**78**(10)

[100] Quandt K., Frech K., Karas H., Wingender E., Werner T. (1995) MatInd and MatInspector: new fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucl. Acids Res.* **23**:4878-488

[101] Quandt K., Grote K., Werner T. (1996) GenomeInspector: a new approach to detect correlation patterns of elements on genomic sequences. *Computer Applic. Biosci.*, **12**:405-413

[102] Quandt K., Grote K., Werner T. (1996) GenomeInspector: basic software tools for analysis of spatial correlations between genomic structures within megabase sequences. *Genomics*, **33**:301-304

[103] Rampone S. (1998) Recognition of splice junctions on DNA sequences by BRAIN learning algorithm, *Bioinformatics*, **14**(8):676-684

[104] Reese M. G. (1994) Erkennung von Promotoren in pro- und eukaryontischen DNA-Sequenzen durch Künstliche Neuronale Netze, Diploma work, University of Heidelberg, Germany

[105] Reese M. G., Eeckman F. H. (1999) Time-delay neural networks for eukaryotic promoter prediction, unpublished

[106] Reese M., NNPP program internet address `http://www-hgc.lbl.gov/projects/promoter.html`

[107] Richard M. D., Lippmann R. P. (1991) Neural networks classifiers estimate Bayesian *a posteriori* probabilities, *Neural Computation*, **3**:461-483

[108] Ripley B.D. (1996), Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press.

[109] Rumelhart D. E., McClelland J .L., and the PDP Research Group, Eds., (1986) *Parallel Distributed Processing*, **1**(2) Cambridge, MA: The M.I.T. Press

[110] Scherf M., Klingenhoff A., Werner T. (2000) Highly specific localization of promoter regions inlarge genomic sequences by PromoterInspector: A novel context analysis approach, *J. Mol. Biol.*, **297**:599-606

[111] Schoenauer M., Xanthakis S. (1993). Constrained GA optimization. In Proceedings of 4 th ICGA, S. Forrest (ed.), San Mateo, CA: Morgan Kauffmann. 573 580.

[112] Sietsma J., Dow R. J. F. (1988) Neural net pruning - why and how. *Proc. IEEE Int. Conf. Neural Networks,* **I,** San Diego, 325-333

[113] Simon H. (1994) Neural Networks. Macmillan College Publishing Company Inc. ISBN 0-02-352761-7

[114] Singer V. L., Wobbe C. R., Struhl K.(1990) A wide variety of DNA sequences can functionally replace a yeast TATA element for transcriptional activation, *Gene Dev*, **4**:636-645

[115] Snyder E. E., Stormo G. D. (1993) Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucl. Acids Res.*, **21**: 607-613

[116] Solovyev V., Salamov A. (1997) The Gene-Finder computer tools for analysis of human and model organisms genome sequences, in *Proc. of the Fifth Int. Conf. on Intelligent Systems for Molecular Biology* (T. Gaaserland, P. Karp, K. Karplus, C. Ouzounis, K. Sander and A. Valencia, Eds.), ISMB97, 294-302, AAAI Press, Menlo Park, CA

[117] Specht D.F. (1996) Probabilistic Neural Networks and General Regression Neural Networks, in *Fuzzy Logic and Neural Network*

[118] Specht D. F. (1988) Probabilistic Neural Networks for Classification, Mapping or Associative Memory, *Proc. IEEE Int. Conf. on Neural Networks,* **1**:525-532

[119] Staden R. (1984) Computer methods to locate signals in nucleic acid sequences. *Nucl. Acids Res.*, **12**:505-519

[120] Staden R. (1988) Methods to define and locate patterns of motifs in sequences. *Computer Applic. Biosci.*, **4**:53-60

[121] Stormo G. D., Schneider T. D., Gold L., Ehrenfeucht A. (1982) Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*, *Nucl. Acids Res.*, **10**:2997-3011

[122] Stormo G. D.(1988) Computer methods for analyzing sequence recognition of nucleic acids. *Ann. Rev. Biophys. Biophys. Chem.* **17**:241-263

[123] Stormo G. D. (2000) DNA binding sites: representation and discovery, *Bioinformatics*, **16**(1):16-23

[124] Uberbacher E. C., Mural R. J. (1991) Locating protein-coding regions in human DNA sequences by a multiple sensor-neural approach. *Proc. Natl. Acad. Sci. U.S.A.* **88**:11261-11265

[125] Veljković V., Slavić I. (1972) Simple General-Model Pseudopotential, *Phys. Rev. Lett.*, **29** (5):105-107

[126] Veljković V., Ćosić I., Dimitrijević B., Lalović D. (1985) Is It Possible to Analyze DNA and Protein Sequences by the Methods of Digital Signal Processing?, *IEEE Trans. Biomed. Eng.*, **32**(5):337-341

[127] Wang L. (1999) Multi-associative Neural Networks and Their Applications to Learning and Retrieving Complex Spatio-Temporal Sequences, *IEEE Trans. on Systems, Man, and Cybernetics - part B: Cybernetics*, **29**(1):73-82

[128] Warwick, K., Irwin, G. W., Hunt, K. J. (1992) Neural networks for control and systems, Peter Peregrinus Ltd.

[129] Wobbe C. R., Struhl K. (1990) Yeast and human TATA-binding proteins have nearly identical DNA sequence requirement for transcription in vitro, *Mol. Cell Biol.*, **10**:3859-3867

[130] Workman C. T., Stormo G. D. (2000) ANN-Spec: A method for discovering transcription factor binding sites with improved specificity, *Pacific Symposium on Biocomputing*, **5**:112-123

[131] Xu Y., Uberbacher E. C. (1998) Computational gene prediction using neural networks and similarity search, in *Computational Methods in Molecular Biology*, eds. S. Salzberg, D. Searls, and S. Kasif (Elsevier Science B.V)

[132] Zhang M. Q. (1998) Identification of Human Gene Core Promoters in Silico, *Genome Research*,**8**:319-326