PATTERN RECOGNITION IN THE CASE OF STRONG BACKGROUND NOISE

by

Xingmei Wang

PATTERN RECOGNITION IN THE CASE OF STRONG BACKGROUND NOISE

by

Xingmei Wang

Dissertation submitted in compliance with the requirements for Maters Degree in Technology in the Department of Mechanical Engineering at Technikon Natal.

This Dissertation represents my own work

APPROVED FOR FINAL SUBMISSION

28/02/2001

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

Date

Durban 28th of February, 2001

ABSTRACT

This study presents a development of a method for recognition of a class of patterns in signals contaminated by strong noise. The class of signals considered is described by a finite alphabet. The target class of patterns is assumed to have specific statistical properties that can be conveniently captured by the position weight matrices (PWM) description. It is also assumed that the signals contain numerous patterns similar to the patterns of the target class, but which belong to different classes. These other patterns represent the noise in the signals.

The method for improved recognition of the target class of patterns is based on clustering of the target motifs with regard to distance form the reference point (event) in the signal. This positional clustering enables more precise description of the target class of patterns by means of the PWMs. However, it requires the use of as many PWMs as there are clusters of the target class. The method developed is of general nature, applicable to the situations described. It is however, applied to the recognition of the specific short motifs in DNA sequences.

The short motif considered is the TATA-box, one of the most important docking sites for proteins in Eukaryotic polymerase II promoter regions. The reference point in the singals obtained form DNA sequences is the transcription start site (TSS). Thus the positional clustering of the TATA-box motif resulted in 20 different PWMs, instead of only one that describes the whole TATA motif class. This however, resulted in more discriminative PWMs and the recognition accuracy has increased by about a factor of two when compared to the recognition of the TATA motif based on the original PWM.

ACKNOWLEDGEMENTS

I express my sincere gratitude to my promoter, Professor Vladimir Bajic, who suggested the problem to me and gave ideas for the initial solution. This served as a basis for full development of the method presented in the study. He was a permanent source of support and encouragement during my research work. I am especially grateful for his patience with me -- a Chinese national, a foreigner.

I also wish to thank my husband -- Dr. Daohang Sha who is always supporting me from all aspects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This study is aimed at developing a method for successful extraction/recognition of patterns that belong to a particular class from a signal that contains numerous very similar patterns, but from different classes. Thus, one can consider these other patterns as noise in the signal, and the basic class of patterns searched for as the useful information (clean signal). We will develop a technique that is applicable to signals which take values form a discrete alphabet, and where the pattern is described as the 0-order Markov chain. This amounts to be equivalent to the so-called position weight matrix description. One can argue that 0-order Markov chains are not chains due to the independence of positional symbols.. However this fits into the description of k-order Markov chains and PWM description. We will apply a specific technique of positional clustering to a pattern named TATA-box motif in DNA sequences found in promoters, which plays a very important role in the transcription of DNA. Recognition accuracy of this motif by computational methods is not very good. Our positional clustering will result in considerably improved recognition accuracy of this pattern than if no positional clustering is applied. The approach used is not bound in any way to the TATA motif or to DNA sequences, and has a more general character. Although we will develop the method for the TATA motif recognition due to a convenience, the method can in a straightforward manner be extended to any motif found in the finite alphabet type signal such as, for example, in digital coding,

quantized speech signals, etc.

In what follows we will present briefly some of the most important techniques that are in use in pattern recognition. Later in this section, we will also briefly present some of the basics of biological sequences that are utilized in the subsequent chapters.

1.1 Signal processing and pattern recognition

1.1.1 Signal and noise

Our environment contains signals that we sense, such as sound, temperature, light, etc. [47]. We adjust our behavior based on our perception and assessment of signals form the environment.

In general, a signal contains information and transmits it. The information conveyed in a signal may be used by organisms or machines for communication, forecasting, decision making, control, etc. When signals are received (observed) they are often distorted, incomplete and noisy, so, we consider signals to contain both useful information and other part that we cannot use (or do not know how to use) and which we denote as noise.

What is noise? The noise, as used here, denotes a non-useful portion of a signal. The noise may itself contain information, even useful information, but if we are not able to use if effectively, that information is lost. Noise may have different causes, and generally it represents deviations of the measured (observed) signal values from their normally expected values [39]. Generally, there are two types of sources that may cause noise. The first comprises the sources external to the system; the second comprises the sources internal to the system. Such noise represents our main interest.

A noisy signal can be modelled in many cases as

$$y(m) = b(m)x(m) + n(m),$$

where m is a time/event index, $x(m)$ is the value of the signal x at m and $x(m)$ has to be detected, $n(m)$ is the noise and $b(m)$ is a binary-valued state sequence; when $b(m) = 1$, this indicates the presence of the signal $x(m)$, while if $b(m) = 0$, this means that the signal is absent.

1.1.2 Signal processing and noise reduction

In general, there is a translation or mapping between the information and signal that carries the information. The process of embedding information into signals, handling signals, and extracting information from them, roughly represents signal processing. In principle, there are two types of signals. One represents the analogous signals, those that change their values without interruption and the opposite group are those that are termed discrete. Usually the discrete signals are conveniently processed by digital computers and thus such signals are sometimes termed digital. The respective types of signal processing techniques are called analogous signal processing and digital signal processing [57], [47], [26].

The predominant mode of technical signal processing is digital, mainly caused by the enormous progress of computer technology. There are two main areas of applications of DSP (digital signal processing) techniques, although they overlap in many respects: (a) telecommunication, where signals need to be transmitted, received, stored, and represented efficiently and reliably, and (b) information extraction that comprises pattern recognition, forecasting, decision making, signal enhancement, control, etc., where information has to be extracted from the noisy signals. So, the information extraction relates to identification, modelling and utilization of patterns and structures in DSP. It plays a critical roll in information technology and digital telecommunication. DSP is broadly applied in domains such as adaptive noise reduction, channel equalization, pattern recognition, audio signal coding, signal detection, and so on. Our interest will focus on pattern recognition [57].

Noise reduction is an important part of a signal processing system. Since the noise

denotes an undesirable portion of the signal which contaminates the useful information, we have an interest to apply such signal processing techniques that can reduce the level of the noise. In fact, we are interested in making the so-called signal-to-noise ratio as large as possible, thus emphasizing the useful content of the signal relative to the noise part [26]. There are many techniques that can be used for this purpose, but this also depends on the type of signal and the type of noise contained in it.

1.1.3 Signal processing methods

DSP algorithms can be clustered into four broad categories according to the methodological approaches on which they rely. These are: non-parametric signal processing, model-based signal processing, Bayesian statistical signal processing and neural network based signal processing. Some of these overlap to some extent. But in practice, a combination of these methods performs better. In what follows we give a brief description of these methods.

Model-based signal processing

A model-based signal processing approach utilizes a parametric model which describes the signal generation process. This parametric model describes the structures and patterns in the signal processing which are predictable or observable. The parametric model can be used to forecast the future values of a signal from its past trajectory. The disadvantage of the model-based signal processing method is that it can be sensitive when the input deviates from the class of signals which is characterized by the model. Model-based signal processing has been widely used in many applications. The most widely used parametrial model is the linear prediction model which has found effective applications in diverse areas of data forecasting, speech recognition, low bit rate coding, model-based spectral analysis, interpolation, signal restoration, etc.

Most signals can be viewed as the combination of partially predictable and partially random (unpredictable) components. These signals can be modelled as the output of a

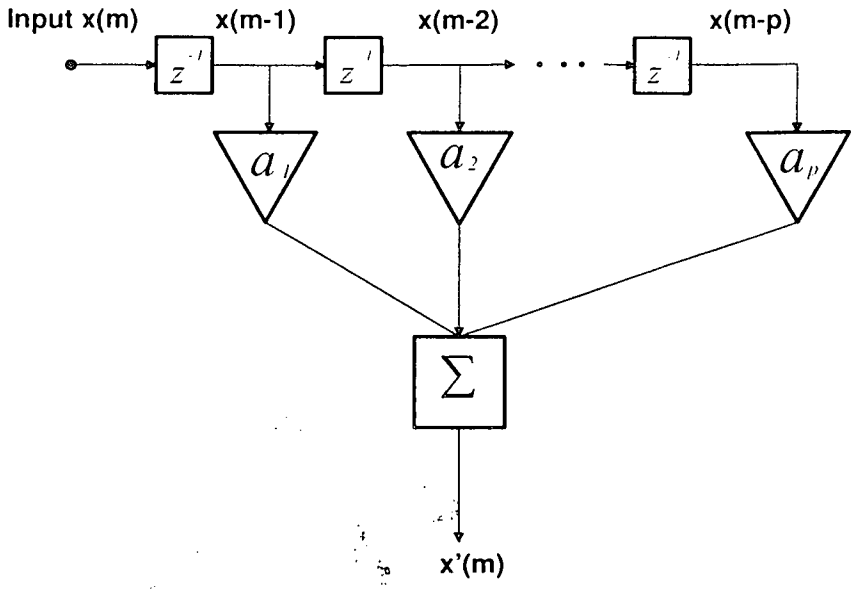

Figure 1-1: Structure of a linear prediction system.

filter excited by an uncorrelated input. The random input models the unpredictable part of the signal, while the filter models the predictable structure of the signal. The aim of linear prediction is to model the mechanism that introduces the correlation into a signal.

Let $x(m)$ represent a value of the signal x at the discrete time (moment) m. Note that m is an integer. A linear prediction model forecasts the amplitude of signal $x(m)$ at the moment m, using a linearly weighted combination of p past samples $[x(m-1), x(m-2), ...x(m-p)]$, as given by the following equation

$$x'(m) = \sum_{k=1}^{p} a_k x(m-k) \tag{1.1}$$

where $x'(m)$ is the prediction of $x(m)$, and a_k are the predictor coefficients. A block diagram implementation of the equation above is illustrated in Figure $1-1$.

The difference $e(m)$ between the actual sample $x(m)$ and its predicted value $x'(m)$, is given by

$$
\begin{aligned}
e(m) &= x(m) - x'(m) \\
&= x(m) - \sum_{k=1}^{p} a_k x(m-k),
\end{aligned}
\tag{1.2}
$$

so, the signal can be described by the following recursive equation (1.3)

$$
x(m) = \sum_{k=1}^{p} a_k x(m-k) + e(m). \tag{1.3}
$$

Non-parametric signal processing

A non-parametric signal processing approach is the opposite of the model-based signal processing. It does not use any parametric model of the signal generation, or a model of the statistical distribution of the signal. The signal is processed as a waveform or a sequence of digits. It is a broadly applicable method that can be applied to any signal regardless of the characteristics or the source of the signal, and it is not restricted to any particular class of signals. The disadvantage of this method is that it does not use the distinct characteristics of the signal process which may improve the performance. That is why it is always outperformed by the model-based signal processing if the model-based approach is applied to a specific signal class. A non-parametric signal processing method is utilized in power spectrum estimation because it is relatively robust and computationally less expensive than the model based method.

Power spectrum reveals the existence, or the absence, of repetitive patterns and correlation structures in a signal. These structural patterns are important in a wide range of applications such as telecommunication systems, data forecasting, signal coding, signal detection, radar, pattern recognition, and decision making systems. The most common method of spectral estimation is based on the fast Fourier transform (FFT). The power spectrum is the Fourier transform of the correlation function. It reveals information on

the correlation structure of the signal.

By using Furier transform, a signal can be expressed by a set of relatively simple basic signals. In practice, signal is a sampled discrete time variable, so the Fourier transform of a sampled signal $x(m)$ is

$$X(f) = \sum_{m=-\infty}^{\infty} x(m)e^{-j2\pi fm},$$

and its power spectrum is defined as

$$\begin{aligned} P_{XX}(f) &= \frac{1}{N}\left|\sum_{m=0}^{N-1} x(m)e^{-j2\pi fm}\right|^2 \\ &= \frac{1}{N}|X(f)|^2, \end{aligned}$$

where N is the number of samples.

In general, the more correlated or predictable a signal, the more concentrated is its power spectrum. Figure $1-2$ illustrates a signal and its power spectrum. From the power spectrum one can observe that the components at frequencies $50Hz$ and $120Hz$ are most represented and that they constitute the mayor part of the signal.

Bayesian statistical signal processing

As mentioned before, the predictable part of the signal can sometimes be modeled by the linear prediction model. However, for the unpredictable (random) signal parts, only a very limited class of signals can rely on description as used for the predictable part. In fact, linear predictor model methods could be viewed as a special case among statistical methods, termed as autoregressive (AR) processes. It requires much more complex models to describe the fluctuations or distributions of random signals with a reasonable accuracy. We have to use statistical methods that rely on conditional probabilistic

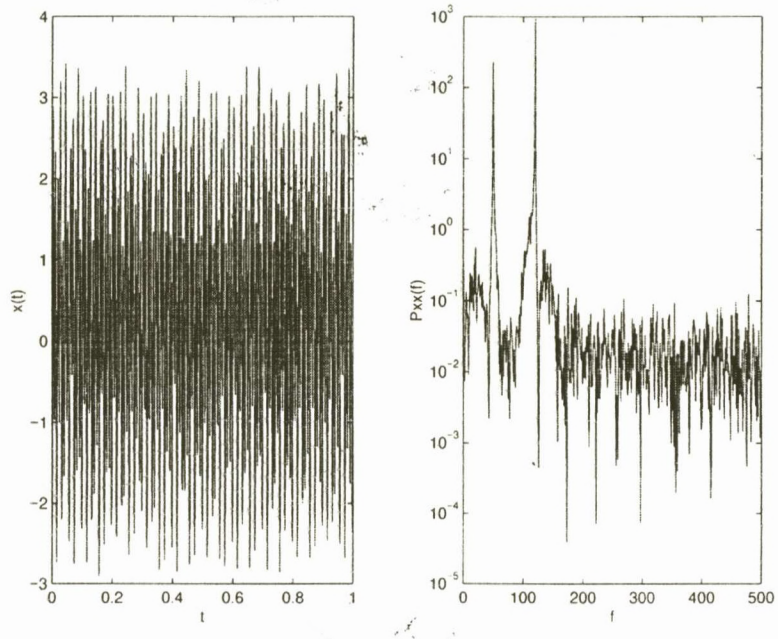

Figure 1-2: A predictable signal and its power spectrum

models.

Bayesian theory is a general estimation/classification framework in which we can include specific prior knowledge of the observation signal and the unknown parameters for a given problem. Estimation is concerned with the determination of the best estimate of an unknown parameter vector, or the recovery of a number of distorted samples. The classification is the labelling of an observed signal with one of N classes of signals. Classifiers are used in applications such as decoding of discrete-valued symbols in digital receivers, speech recognition, character recognition, impulsive noise detection, signal/noise classification etc.

Classification, assignment, identification and dissection

Classification is defined by A. D. Gordon [38] as "a portmanteau term to describe the miscellaneous collection of techniques" which are used to investigate the structure within a data set, then to determine if the objects fall into any certain smaller number of groups (or classes, or clusters). The data set comprises a set of objects, and each object is described by several variables. In classification, the number of groups and composition of any group are not known at the start of the investigation.

Gordon also pointed out the difference between classification and another three similar concepts: assignment, identification, and dissection. When classification is used to refer to the procedure of deciding to which of a known number of existing classes a new object should belong, then this is the assignment. For example, in various forms of discriminant analysis, an object is assumed to have been sampled from one of a known number of populations of objects about which one has relevant information, and the aim is to assign the object to the correct population. Then this is called *assignment*. In pattern recognition, information about patterns can be obtained from a training set of observations, and one wishes to identify a new observation with one of the existing patterns. Then this is the so-called *identification*. In these two cases, there are groups of objects. In signal processing, the aim of signal classification is to design a system

for labelling a signal with one of a number of likely categories [57]. This means that the signal classification belongs to identification or to assignment. But in *dissection*, the data set comprises only a single group of objects and the aim is to dissect this group into several sectors which have certain specified properties. Classification procedures are sometimes used to obtain a dissection of a data set in which the objects belong to a single group. However, the aim of classification is to uncover the structure in the data, rather than to impose some inappropriate structure on them. So, in a classification study, it is usually required that the groups must be different from one another, i.e. that objects within a group shall be dissimilar from objects in other groups. On the other hand, just as H. T. Clifford [23] has pointed out, "identification and dissection are only two aspects of classification".

The general purpose of a classification is either for data simplification or for prediction. According to the different methods utilized, most classification procedures fall into one of the following two types: One is a geometrical method in which each object is represented by a point in a low-dimensional space. Objects which are similar to one another are represented by points which are close together. Another is the so-called clustering or cluster analysis method which can be divided into three different methods according to different techniques utilized: (1) a partitioning method which aims at partitioning the set of n objects into g disjoint groups so that each object belongs to one and only one group. The value of g has to be specified by the investigator; (2) a hierarchical method which investigates the structure in the data at several different levels and how the groups in a partition are related to one another; (3) a clumping method in which the groups are allowed to overlap. Such overlapping groups are called clumps, and a division of the set of objects into clumps, so that each object belongs to at least one clump, is called a covering of the data set. The illustration of the latter three types of classification is as Figure $1-3$.

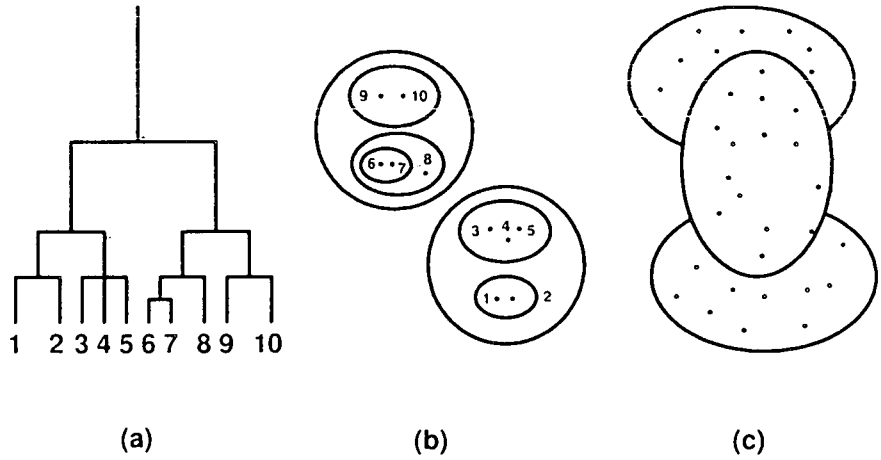

Figure 1-3: Hierarchical classification: (a) by a tree diagram, (b) by a nested set of partitions; (c) An overlapping classification displaying three clumps.

Cluster analysis

Cluster analysis essentially deals with decision making problems in the nonparametric and unsupervised learning mode. The result of a cluster analysis can contribute directly to development of classification schemes. In cluster analysis the number of categories or clusters may not even be specified. The task is to discover a reasonable structure within complex bodies of data. In a typical example, one has a sample of data units, each described by scores on selected variables. The goal is to group either the data units or the variables into clusters so that the elements within a cluster have a high degree of natural association while the clusters are relatively distinct from one another. The approach to solve the problem and the results achieved depend principally on how to give operational meaning to the terms: "natural association" and "relatively distinct".

Cluster analysis is the main technique utilized in classification procedure. It has been broadly used in many fields of study such as life sciences, behavioral and social sciences, earth sciences, medicine, engineering sciences, information and policy sciences. But how to carry out the cluster analysis entails a host of problems. The foremost difficulty is that cluster analysis is a loose collection of heuristic procedures and diverse elements of applied

statistics rather than a single integrated technique with well defined rules of utilization. The actual search for clusters in real data involves a series of intuitive decisions. M. R. Anderberg [2] has concluded nine major steps in clustering a set of data: (1) Choice of data units. There are two different situations in this step. The first one is when the sample is the complete object of the analysis, then the principle consideration here is to make sure no important data unit is omitted; the second which occurs more frequently and is more troublesome is that the sample is a portion of a much larger population which is the true object of interest. To deal with this case, statistic principles of random and independent selection is essential. (2) Choice of variables. The data units must be described by some properties termed as variables; the choice of variables may have the greatest influence on the ultimate results of a cluster analysis. (3) Decide what to cluster. In the former two steps, we choose (cluster) either data units or variables separately.. However, it is possible to consider a alternative approach in which the clustering is alternated between variables and data units until "... mutually harmonious classifications are achieved for both''. (4) Homogenizing variables. In real data, variables are not homogenized. In measuring association among variables, different types of scales present difficult problems, so, it is necessary to amalgamate all the variables into a single index of similarity. (5) Similarity measures. Most cluster analysis methods require a measure of similarity for every pair of combination of entities to be clustered. (6) Clustering criteria. The choice of a clustering criterion is tantamount to defining a cluster. In some problems there is a natural choice, while in others almost any criterion might be a candidate. Sometimes, clustering the data set several times with different criteria is a good way to reveal various facets of the structure of the data set. (7) Algorithms and computer implementation. After those 6 steps above, a proper algorithm needs to be selected according to the capabilities of an available computer program. (8) Number of clusters. This is a substantial practical problem in performing a cluster analysis. With different numbers of clusters, we may get optimal results. (9) Interpretation of results. The results are interpreted at three levels, at the least sophisticated level, the clusters are summary descriptive statistics,

at the next level; the set of choices described are well defined, so that the clusters have the desired properties; at the highest level, the results of a cluster analysis are an aid to explain the data.

Bayesian estimation

Bayesian theory is based on the minimization of a so-called Bayes risk function which includes a posterior model of the unknown parameters given the observation and a cost of error function. A simplified description about Bayesian theory will be given later while we introduce the methods of pattern recognition.

An estimator takes a set of noisy or incomplete observations as input, uses a predictive or a statistical model of the process, and estimates the unknown parameters. The estimation accuracy depends on the available information and on the efficiency of the estimator. Assume the signal can be modelled by

$$y = X\theta + e + n,$$

where y is the observation, X is the signal matrix, θ is the parameter, e is the random input to the model, and n is the noise.

The estimate of parameter vector θ is a function of the observation vector y, the length of the observation N, and the process model M

$$\theta' = f(y, N, M).$$

Bayes risk function is an average cost of error function denoted by $R(\theta')$, while cost function is denoted by $C(\theta', \theta)$. Bayesian estimation includes the classical estimators such as the maximum a posterior (MAP), maximum likelihood (ML), minimum mean squared error (MMSE), and minimum mean absolute value of error (MAVE).

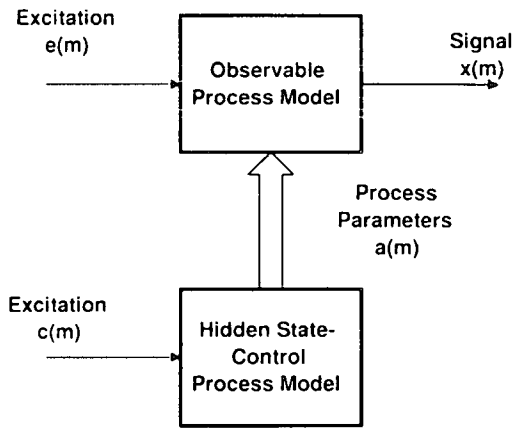

Figure 1-4: A two-layered model of a nonstationary process

Hidden Markov Model

The hidden Markov model (HMM)[57], [43], which is widely used in statistical signal processing, is also an example of a Bayesian model. It is used for nonstationary stochastic processes such as speech and time-varying noise. A nonstationary process can be modelled as a double-layered stochastic process, with a hidden process that controls the time-variations of the statistics of an observable process, as illustrated in Figure $1-4$. A HMM is a double-layered finite state stochastic process, with a hidden Markovian process that controls the selection of the states of an observable process.

A nonstationary first order autoregressive (AR) process can be modelled as:

$$\begin{aligned} x(m) &= a(m)x(m-1) + e(m) \\ a(m) &= \beta a(m-1) + c(m) \end{aligned}$$

where $x(m)$ is the observable signal, $e(m)$ is the signal excitation, $a(m)$ is the time-varying coefficient of the observable AR process, β is the coefficient of the hidden state-control process, and $c(m)$ is the parameter excitation. In a Markovian process, the next state of

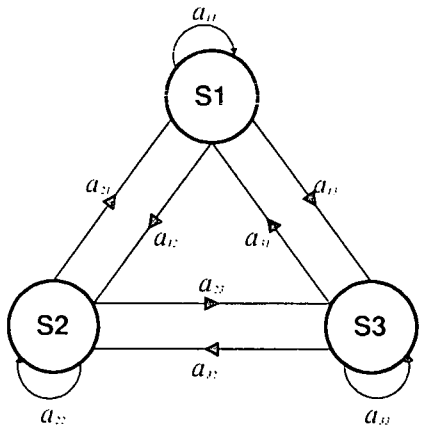

Figure 1-5: A three-state ergodic HMM structure

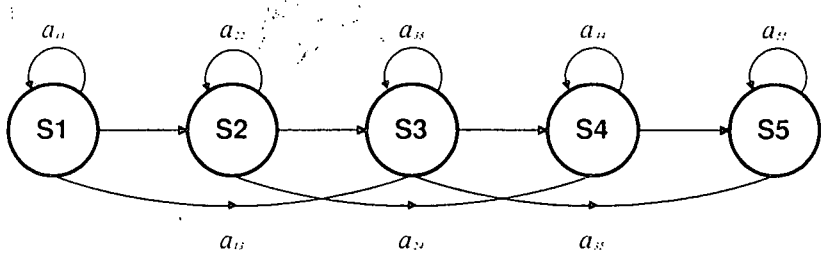

Figure 1-6: A 5-state left-to-right HMM model

the process depends on the current state.

There are two kinds of nonstationary processes: continuously variable state processes in which underlying statistics vary in a continuous manner with time, and finite-state processes in which statistical characteristics can switch between a finite number of stationary states. Figure $1-5$ represents a general form of a three-state HMM known as an ergodic HMM which means there is no structural constraints for connecting any state to any other state exist. Figure $1-6$ represents a constrained form of HMM the left-to-right model.

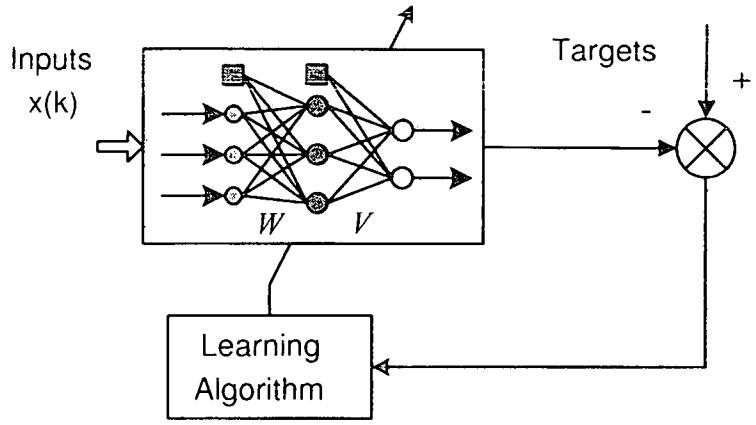

Figure 1-7: A general structure of Neural Network

Neural networks

Artificial neural networks are composed of many simple elements operating simultaneously. They are inspired by biological nervous systems. They are widely applied in various fields such as pattern recognition, identification, vision and control systems, etc. The main advantages of neural networks are that they have the ability to learn complex nonlinear relationships of input and output. They use sequential training procedures and adapt their parameters and structure to the data. Neural networks are particularly useful in nonlinear partitioning of a signal space, in feature extraction and pattern recognition, and in decision making systems.

The general structure of a neural network is illustrated in Figure $1-7$.

The architecture of a network consists of a description of how many layers a network has (except for input layer and output layer, other layers are called hidden layers), the number of neurons in each layer, each layer's transfer function, and how the layers are connected to each other and network inputs. The network is used to map the input values to output values, so the number of inputs and the number of the outputs, as well as the number of the neurons in these two layers, are determined by the particular mapping problem. The number of neurons in hidden layers is up to the designer. Except

for purely linear networks, the more neurons in a hidden layer, the more powerful the network.

Usually, use of a nonlinear transfer function makes a network capable of solving nonlinear problems, and multiple layer networks are more powerful in these tasks. Neural networks have been applied in many fields such as aerospace, automotive, banking, defense, electronics, entertainment, financial, insurance, manufacturing, medical, robotics, speech, telecommunication, transportation etc.

1.2 Patterns and pattern recognition

Pattern is the opposite to chaos, it could have some features that could be defined. Patterns can be named [42]. A pattern could be a fingerprint image, a handwritten cursive word, a human face, a speech signal, etc.

Although the best pattern recognizers in most instances are humans, in many cases computers may do a better job, especially when the database and the number of patterns is too big for humans to analyze. The rapidly growing and available computing power has facilitated the use of diverse methods for data analysis and classification while enabling faster processing of huge data sets. At the same time, the availability of large databases and stringent performance requirements (speed, accuracy and cost) are demanding automatic pattern recognition more than ever. So, automatic (machine) recognition, description, classification, and grouping of patterns have become important problems in a variety of disciplines such as biology[1], psychology, medicine, marketing, computer vision, artificial intelligence, remote sensing, and so on.

What is pattern recognition? To put it simply, pattern recognition is identification, one kind of classification. It is the study of how machines can observe the environment, learn to distinguish interesting patterns from their background, and determine which class (or group, or cluster) the patterns belong to. After almost 50 years of research, pattern recognition has been regarded as the best possible way of utilizing available sensors,

processors, and domain knowledge to make decisions automatically, although it is still an elusive goal to design a general purpose machine pattern recognizer. Pattern recognition technology has been broadly utilized in many areas, from 2D objects to 3D objects, for example, sequence analysis in bioinformatics, searching for meaningful patterns in data mining, reading machine for the blind in document image analysis domain, printed circuit board inspection in the industrial automation domain, personal identification in the biometric recognition domain, telephone directory enquiry without operator assistance in the speech recognition domain, etc.

Today, inexpensive sensors and powerful desk-top computers are available due to the various technological developments. Thus, a number of commercial pattern recognition systems can be found for individual use, e.g., machine printed character recognition and isolated spoken word recognition.

Generally, pattern recognition falls in two categories: a/ supervised classification in which the pattern classes have been defined by the system designer, and b/ unsupervised classification in which the pattern is a unknown class, needs to be learned, extracted from the background. In most cases, available features of a pattern have to be extracted from training sets and optimized by data-driven procedures, so the unsupervised classification is more popular.

1.2.1 Approaches for pattern recognition

There are many approaches for pattern recognition. The best known four are template matching, statistical classification, syntactic or structural matching, and neural pattern recognition. It should be pointed out that in many of the emerging applications, no single approach is optimal. So these approaches need not necessarily be treated independently. A combination of multiple methods and approaches has been a commonly used practice in pattern recognition[42].

Template matching

Template matching is the simplest, earliest, and most intuitive approach. Matching means determining the similarity between two entities such as points, curves or shapes of the same type. In template matching, firstly a template (typically a 2D shape), or a prototype of the pattern to be recognized, is extracted out, unless it is predefined. Then, match the pattern against the stored template while some translation, rotation, and scale changes are allowable. Often, template matching demands huge computation, but now the availability of faster computers has made it more feasible. Although it is effective in some applications, the template matching approach has a number of disadvantages. For instance, it would fail if the patterns are too distorted relative to the similarity measures used.

Syntactic approach

In syntactic approach[29], a pattern is viewed as being composed of simple subpatterns which are built from even simpler subpatterns. The simplest subpatterns are called *primitives*. Pattern forms the interrelationship between these primitives. We can also view a pattern as a sentence of a language, and, then, primitives are viewed as the alphabet of the language. The sentence is generated according to a grammar. The grammar for each pattern is inferred from the available training samples. So, a large collection of complex patterns can be described by a small number of primitives and a set of grammatical rules.

This method provides a description of how the given pattern is constructed from the primitives and has been used when the patterns have a definite structure which can be captured by a set of rules, such as, for example, the EKG waveform, textured images, shape analysis of contours, etc.

But there are many difficulties in the implementation of syntactic approach. Main problems are with the segmentation of noisy patterns, in detection of the primitives, and the inference of the grammar. Also, the syntactic approach may require a large amount

of training data, which in turn requires very large computational efforts.

Neural networks

The most commonly used type of neural networks for pattern recognition is the feed-forward network which includes multilayer perceptrons and radial-basis function networks. Both of them are organized into layers and have unique directional connections between the layers. The difference between them is that the former often uses the log-sigmoid transfer function or tan-sigmoid transfer function, while the later uses radial bases transfer functions. Although neural networks do have several advantages compared with the statistical approach, most of the well-known neural network models are implicitly equivalent or similar to the classical statistical pattern recognition methods. in spite of the different underlying principles. Just as Anderson et al. [3] points out that – neural networks are statistics for amateurs.

In many real world problems, patterns are scattered in high-dimensional nonlinear spaces, so nonlinear procedures to reduce the feature space to lower dimensions have become popular. Neural networks offer powerful tools for these dimension reduction purposes.

Statistical approach

The statistical approach has been most intensively studied and used in practice and has been successfully used to design a number of commercial recognition systems [42],[4]. In the statistical approach, two datasets, i.e. a training (learning) set and a classification (testing) set are required. The objective is to establish decision boundaries which separate patterns belonging to different classes from the training set, to train the classifier with these decision boundaries to partition different classes, and to classify different patterns from the testing set.

First of all, the desired pattern needs to be defined. We need to extract the pattern of interest from the background (training set), remove the noise, normalize the pattern,

and eventually define a compact representation of the pattern. Assume that the desired pattern is described by a vector of d features $x = (x_1, x_2, ...x_d)$. This means the pattern is represented by d characteristic numbers and could be viewed as a d-dimensional vector or a point in a d-dimensional space. It should be pointed out that the choice of the representation strongly influences the classification results. It is better to keep the dimensionality of the pattern representation as small as possible, because a limited but salient feature set simplifies both the pattern representation and the classifiers that are built on the selected representation. Consequently, the classifier will be faster and will use less memory. But on the other hand, a reduction in the number of features may lead to a loss in the discrimination power and thereby lower the accuracy of the resulting recognition system. So, there are two phases during the definition of the pattern, the first phase is called feature extraction which uses linear or nonlinear transforms, such as principal component analysis and discriminant analysis, etc., to extract all the features of the pattern. The second phase is called feature selection which uses methods such as branch-and-bound search method, best individual features method, etc., to discard those features with low discrimination ability.

Then statistical theories are utilized to establish decision boundaries between pattern classes. There are several decision rules in statistical theory of which the Bayes decision rule is the best-known. It is simply summarized in what follows.

Assume that there are c categories of patterns presented by ω_1, ω_2, ω_3, ..., ω_c, and that each pattern is described by a vector. We use the conditional probability function $p(\omega_i|x)$, which is the probability that the pattern x belongs to the class ω_i. Bayes decision rule implies that the pattern $x \in \omega_i$ if $p(\omega_i|x) > p(\omega_j|x)$ for all $j \neq i$, $j = 1, 2, ..., c$. Most of the approaches in statistical pattern recognition attempt to implement the Bayes decision rule.

According to the information available about the class-conditional density, various strategies could be used to design a classifier. If all of the class-conditional densities are completely specified, then the Bayes decision rule can be used directly. However, the

class-conditional densities are usually not known in practice. If we know the form of the class-conditional densities, and only some of the parameters of the densities are unknown, we may replace the unknown parameters in the density functions by their estimated values. The classifiers with this strategy are the so-called Bayes plug-in classifiers. If the form of the class-conditional densities is not known, then we must either estimate the density function or directly construct the decision boundary based on the training data. Then we may need to use the so-called cluster analysis.

1.3 Biological background

In this section we give some simple background necessary to define the problem we will research.

Today it is possible to research the organisms at molecular level simply by analyzing the information contained in their DNA. DNA is a very complex structure[8],[9],[16] and it requires computer analysis for discovering important information contained in it. This field falls under the auspices of bioinformatics, which involves not only the solution of complex biological problems using computational tools and systems, but also the collection, organization, storage and retrieval of biological information and databases [10], [17], [19]. Bioinformatics makes a tremendous impact on our lives already, and is expected to do so much more in future, through the design of new drugs, etc[5].

1.3.1 Protein, DNA, and gene

Organisms are made up of cells which are mainly composed of proteins [59]. *Protein* is a functional unit, a large, organic molecule, polymer, that is composed of different amino-acids. Amino-acids are joined into a long chain. One or more such chains twist together to form a 3-D protein structure. The functional properties of proteins are determined by the linear ordering of amino-acids (protein sequence) and their 3-D shapes. The life activities of a cell are determined by the interactions of different proteins that the cell

synthesizes.

Organisms are divided into two big groups: *Prokaryotes* and *eukaryotes*. *Eukaryotes* are defined by the division of each cell into a nucleus that contains the genetic material, bounded by a nuclear membrane, surrounded by a cytoplasm which is bounded by the plasma membrane that marks the periphery of the cell. The cytoplasm contains other discrete compartments, also bounded by membranes.

Prokaryotes are distinguished from eukaryotes on the basis of nuclear organization, specifically their lack of a nuclear membrane. Prokaryotes also do not have any of the discrete compartments which are characteristic of eukaryotic cells. A prokaryote is the simplest of cells.

Every organism has its unique complete set of genetic information which is called a genome. This genetic information is contained in the molecular structures made up of deoxyribonucleic acid (DNA). Genetic information contained in DNA instructs it to synthesize different kinds of necessary proteins to regulate all biochemical activities of the organism. This process of the synthesis of proteins depends on biochemical processes caused by different parts of the DNA.

The DNA structure is shown in Figure $1-8$ [28]. The DNA molecule has two strands. Each DNA strand consists of a sequence of nucleotides. Each nucleotide consists of a phosphate group, a pentose sugar and one of four bases: adenine (A), thymine (T), cytosine (C) or guanine (G). In the DNA strand, nucleotides appear in different orders. This ordering is assumed to be the most responsible for biological functions of specific parts of the DNA. Two DNA strands are connected together by weak bonds between the bases on each strand. They twist to form a helical 3-D structure. Specific pairing of nucleotides in the strands is arranged in such a way that only A-T and C-G pairing is possible. Such paired nucleotides form the so-called base pairs (bp). The complete genetic information contained in one of the DNA strands is also contained in the complementary DNA strand. One strand is called the sense strand, and the other one is called the template strand.

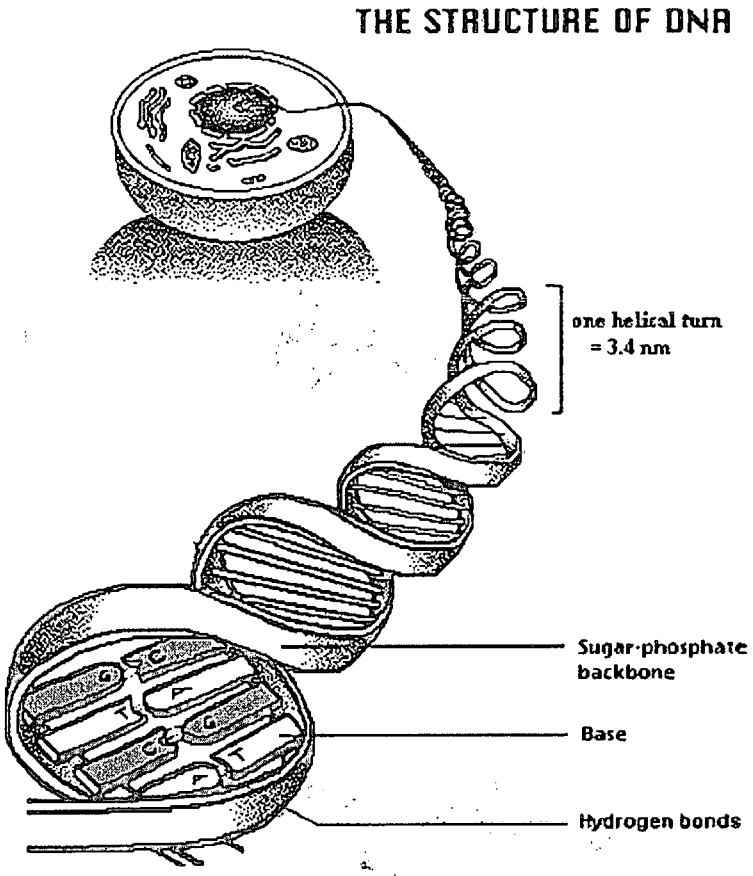

Figure 1-8: The structure of DNA (copied from website: http://www.accessexcellence .org/AB/GG/structure.html)

A *gene* is a 'relatively short' segment of DNA. It carries the information which corresponds to a particular protein. Some diseases can be caused by a change in a single gene. The number of genes differs from organism to organism. It is estimated that the human genome contains approximately $50,000 - 100,000$ genes[15].

The nucleotide sequence in a gene specifies the molecular structure of a protein. Generally, in eukaryotes, the sections in genes responsible for protein-encoding are not in continuous segments, but comprise protein-encoding parts called **exons** and parts that do not encode for proteins called **introns**. Only about 2%~5% of the genome provides codes for proteins [15], [40], but the remaining parts of the genome contain control regions and intergenic regions.

1.3.2 RNA and Protein synthesis

Another nucleic acid, called ribonucleic acid (RNA), plays an important role in protein synthesis. During the process of protein synthesis, information encoded in DNA can not be translated directly to a protein; instead, the initial phase of this process called **transcription,** which generates a single-stranded RNA identical in sequence with one of the strands of the duplex DNA, starts the process. In RNA uracil (U) replaces thymine (T), while the other three bases are the same as that in DNA. Several different types of RNA are generated by transcription. The three principal classes involved in the synthesis of proteins are: **messenger RNA (mRNA), transfer RNA (tRNA)** and **ribosomal RNA (rRNA)**. The mRNA always includes nucleotide sequences which are called **exons** that correspond exactly with the protein product according to the rules of the genetic code, but misses additional sequences which are termed as **introns** that lie within the exons in DNA [45].

The process of synthesis of a protein in eukaryotes, based on information encoded in DNA, can be presented in three steps a/ the transcription phase, which synthesizes RNA under the control of the DNA template; this RNA is called pre-mRNA; b/ the RNA processing whose main purpose is to eliminate sections of the pre-mRNA molecules that

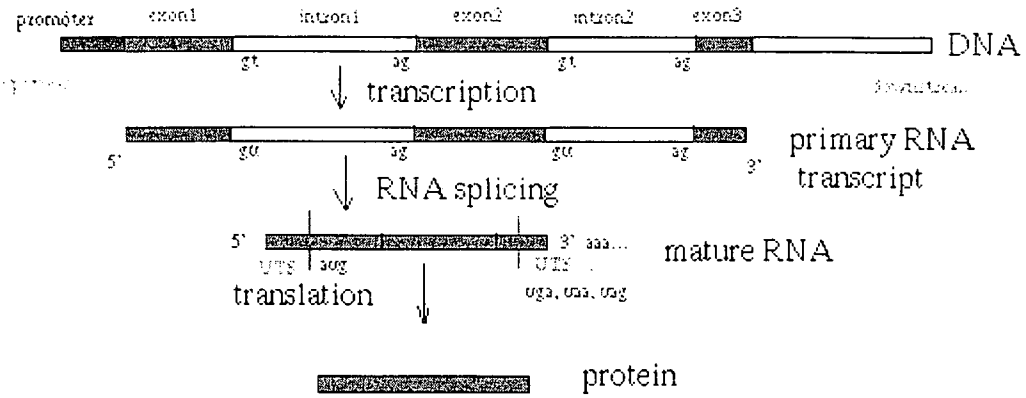

Figure 1-9: The process of synthesis of protein (copied from website: http://linkage .rockefeller.edu/wli/gene/)

correspond to the intron regions in the DNA template, splicing the sections that correspond to the consecutive exon regions in the DNA template, and consequently producing mRNA; c/ the translation, which is the process of the actual synthesis of a sequence of amino-acids based on information in mRNA. This is simplistically represented by a conversion:

DNA template→ pre-RNA (only for eukaryotic organism)→mRNA → protein (Figure $1-9$).

1.3.3 Promoters and their characterizations

Promoters are parts of DNA responsible for the initiation of the transcription process[45], [24]. The promoter surrounds the first base pair that is transcribed into RNA, the startpoint (the so-called **transcription start site (TSS)**). Nucleotides prior to the startpoint are described as upstream of it; those after the startpoint (within the transcribed

sequence) are downstream of it. Positions of nucleotides upstream of the TSS are denoted with the '−' sign, while those downstream of the TSS are denoted by the '+' sign. This will be important in the following chapters.

1.4 Problem definition

We will present here the description of the problem we intend to study. As we are focused to the recognition of patterns in signals having strong background noise we will first describe the types of signals we will work with, and then the type of noise contained in the signals.

Let $S_\alpha = [a_1, a_2, ..., a_k]$, represent an alphabet (a set of discrete values that can be either numerical, string type, or combination of these) sequence. The signal σ, as considered here is a sequence of symbols s_1, s_2, ..., s_N, where each symbol can take any of the values from the set S_α. The length of the signal σ is N. Such signals appear in many fields such as digital representation of analogous signal, coding, data compression, etc.

We assume that the useful information in a collection S of signals σ_i, is represented by relatively short segments of M ($M \ll N$) consecutive symbols that we will call motifs or patterns. If the most represented symbols for each of the possible M positions in the motif are given by $[p_1, p_2, ..., p_M]$, then this pattern is called a consensus pattern. It is allowed that there may be some alterations of this consensus pattern in the whole set of target patterns that relate to one particular motif. Anything else in the sequences σ_j are considered a noise.

The goal is to devise a system that can recognise a particular pattern (target motif) with a great accuracy. However, the problem that is evident is the possibility that patterns very similar to the patterns of the target motif be recognized as correct ones, although they do not belong to the target class. This will cause the so-called false recognition and the good accuracy of the recognition system means that the level of false recog-

nition will be very low, while the level of correct recognition will be relatively high. This is equivalent to having very strong background noise in the signals analyzed. Although the motifs are not mixed with the noise in the overlapping sense, there is no mechanism to accurately separate what is the correct motif from the false one. Equivalently, one has to find out a method to reduce the level of false recognition and consequently reduce the effect of noise in the signal, relative to the useful signal information.

The main reason for the high level of falsely recognized patterns is the inadequacy of the model describing the target pattern. Here we will attempt to enhance one class of such models, based on the initial idea of V. B. Bajić. This idea is based a/ on the assumption that in a set of sequences S_α there is a reference point, a part which contains very specific information, b/ that the target patterns are at different distances from such reference point, and c/ that the target patterns at a particular position have characteristics different form the target patterns at another location. V. B. Bajić suggested that the use of the PWMs to describe the statistical properties of target patterns clustered according to their distances form reference points in signals, will result in more accurate description of the whole set of target patterns.

In this study, we will use this idea to develop a set of techniques that can be applied to any motif recognition, subject to the constraints listed above. Signals that contain such patterns and which generally satisfy these assumptions can be found, for example, in signal processing, in digital coding and in biology. We will develop our results by analyzing one specific group of signals that have their origin in biology. These are signals obtained from nucleotide databases (DNA sequences). Our target pattern will be the TATA-box motif, one of the most important functional parts in promoter sections of DNA. However, it should be indicated that although our development is focused to these biological sequences, the methods presented here can be directly extended to other classes of sequences and other classes of target patterns.

Finally, we present at the end of this study all computer programs developed and used for the analysis, including the Graphical User Interface.

Chapter 2

Promoter recognition

2.1 Detailed description of TATA motif recognition

Currently the primary interest in the process of annotation of genomes is the identification of genes. One group of methods for gene identification is based on homology search[54]. But in many cases, there is still no close homolog and no guarantee that when there is a homolog that the functionality has been preserved.

Another group of methods for gene identification uses identification of sites of intron/exon splicing and translational control, etc., aiming at suggesting overall gene structure. Some of these methods rely upon recognizing the beginnings of genes by recognizing the promoter region or the TSS, the others by recognizing the ends of genes [33], [55].

There are several important reasons to motivate for promoter recognition:

1. A promoter is defined as the starting of a gene. Thus, as long as we find the promoter, we can locate the gene while we partition a genome into genes;

2. After partitioning the genome into genes, the greatest difficulty in eukaryotes is correctly determining the splicing structure. Locating the correct initiation codon is also a difficult and important step in this case. If the TSS is known, it is then much easier to find the start of translation.

3. Gene expression is regulated in a number of ways and at many levels. The most

important part of that regulation comes form the role of promoter, since they are responsible for the start of the protein synthesis process.. By controlling transcription, the main function of the associated gene is controlled.

Thus our interest is in finding promoters in the uncharacterized DNA. One of the ways to locate them can be in searching for some 'common' characteristics of a larger group of eukaryotic promoters.

2.2 Methods used to deal with promoter recognition

We will concentrate on eukaryotic polymerase II promoter recognition. Eukaryotic polymerase II promoter regions [11]are characterized by the presence of short sequence called transcriptional elements (TE), such as TATA-box [49], [48], CAAT-box, Inr, GC-box, together with the other different transcription factors (TF) binding sites. They are located over a region upstream of the TSS of genes [45]. The computational identification of promoter regions is quite difficult by means of recognizing these elements [41], because these TEs vary greatly, they may appear in different combinations, their relative locations are different for different promoters, and not all of them need to exist in a particular promoter. So, it is not easy to precisely define a promoter in eukaryotic organisms, which directly impacts on the efficiency of computer tools for their recognition [21], [22], [5], [31], [32], [35], [56].

Algorithms on promoter recognition rely upon the recognition of TEs through consensus, matrix methods or artificial intelligence techniques [41], [20], [25], [27]. Normally, Regions of DNA which contain a higher density of TEs relative to non-promoter sequence are more likely to be true promoter regions [50], [51]. It has been generally recognized that consensus sequences are usually inadequate to describe the characteristics of TEs, and now matrix method, i.e., Position Weight Matrix (PWM) is one frequently utilized [13], [37].

Some useful programs for promoter recognition are available via the internet [52].

Different techniques have been used in these programs to deal with the recognition of eukaryotic promoters of different classes. For example, **Markov models** of vertebrate promoter sequences (based on EPD) and non-promoter sequences (based on regions adjacent to the promoters used) have been constructed by Audic and Claverie(1997); **Autogene** utilizes a set of consensus sequences for TF binding sites; Time delay neural networks architecture has been used rather successfully in **NNPP** [53](by M. Reese, *http: //www-hgc.lbl.gov/inf/nnpp-abstract.html)* which combines recognition of the TATA-box and the Inr. **Promfind,** which is developed by Hutchinson in 1996 is based on the differences in nucleotide hexamer frequencies between promoters, protein coding regions, and non-coding regions downstream of the first coding exon; **Promoterscan,** which is developed by Prestridge in 1995 is based on both the TATA PWM, and the density of specific TF binding sites; and so on [33]. Some new results are also reported in [7], based on clustering techniques and artificial neural networks.

From the prediction results of many of these programs, it is obvious that the rate of TP (true positive) predictions is directly correlated to the rate of FP (false positive) predictions. The best program in correct predictions **NNPP** produced $TP = 54\%$ and FP at the level of $1/460\,bp$ (bp, Base Pair), on the other hand, **Promoterscan,** produced $TP = 13\%$, but achieved the lowest FP at the level of $1/5520bp$. So, a trade-off between TP and FP is very present in the currently available programs. A good review of classification of prediction results is given in [6].

2.3 Our method

We develop a method which relies upon the initial recognition of TATA-box through the Position Weight Matrix (PWM).

The TATA-box [12], [14], [18], [44], is a short region rich with tymine (T) and adenine (A) and located about $-25\,bp$ upstreams of the TSS (generally in the region of $-11bp$ to $-40bp$ upstream, see Figure $2-1$). About 70%~80% of eukaryotic promoters contain

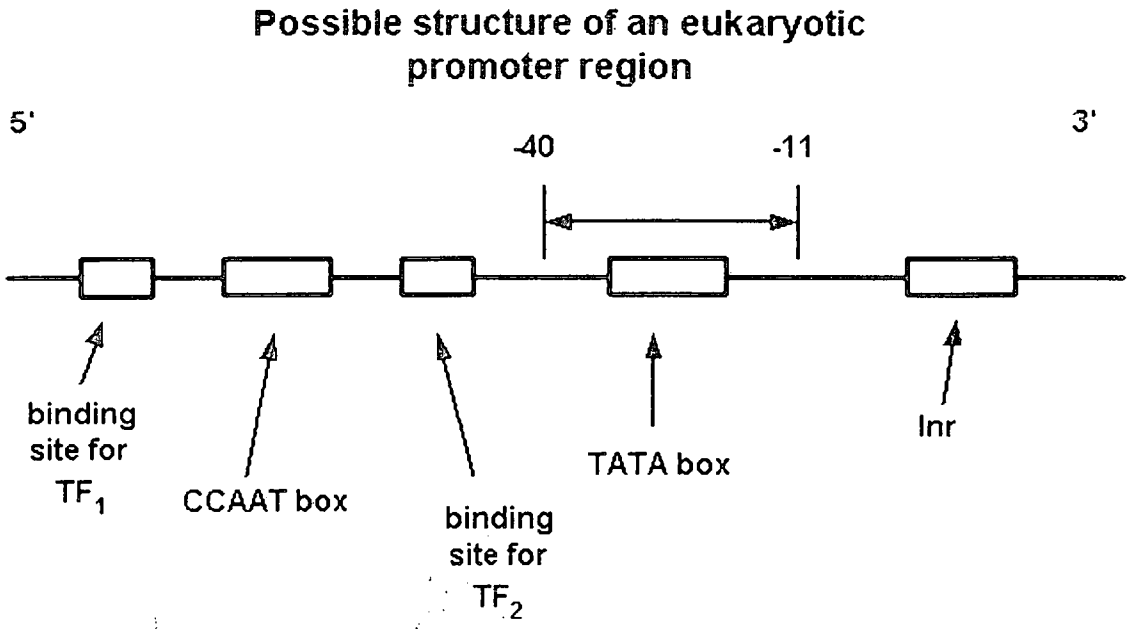

Figure 2-1: One possible eukaryotic promoter structure

TATA or TATA-like subregions. A great number of attempts have been made in promoter recognition tasks by means of utilizing information of TATA or TATA-like component. Position Weight Matrices (PWMs) is one of the methods used to describe TATA-like components. $PWMs$ are very useful and widely used in computational genomics, providing a cumulative representation of a group of data which share some similarities and are represented by the same alphabet [58].

2.3.1 Original PWM for the TATA motif

Our transposed original PWM, derived by Bucher [12], is shown as Table 2.1 :

PWM (non-transposed) has rows that correspond to the four letters of the DNA alphabet, while the columns characterize consecutive positions in the motif. Each letter has some probability to be on a particular position. From this matrix we could observe that, from position 2 to position 7, letter T, A, T, A, A, A, are most likely to appear on the respective positions. Thus, we regard $TATAAA$ as the TATA-box consensus motif.

Position	%A	%C	%G	%T
1	15.7	37.3	39.1	8
2	4.1	11.8	4.6	79.4
3	90.5	0	0.5	9
4	0.8	2.6	0.5	96.1
5	91	0	1.3	7.7
6	68.9	0	0	31.1
7	92.5	0.8	5.1	1.5
8	57.1	0.5	11.3	31.1
9	39.8	11.3	40.4	8.5
10	14.4	34.7	38.6	12.3
11	21.3	37.8	32.9	8
12	21.1	32.6	32.9	13.4
13	21.1	30.3	32.9	15.7
14	17.5	27.5	25.7	19.3
15	19.8	26	36	18.3

Table 2.1: The transposed original PWM

From experimental data we get to know that a TATA-box may be presented from $-11bp$ to $-40bp$ upstream of DNA relative to TSS. This means that the first nucleotide may be present from $-40bp$ to $-16bp$, therefore, when we define PWM and use a data window to slide it along DNA to search for the TATA motif, the possible shifting range for the beginning of the PWM is from $-41bp$ to $-17bp$ (see Figure $2-2$).

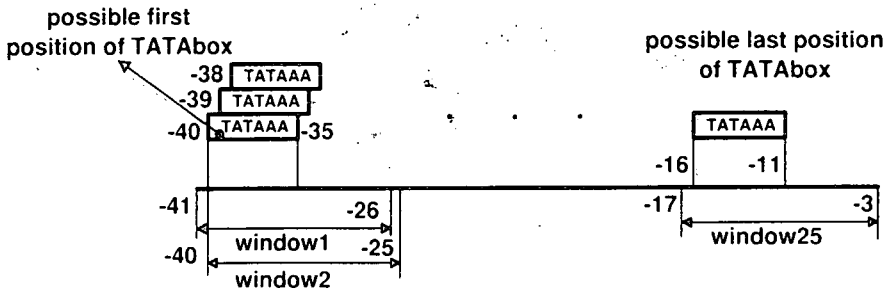

Figure 2-2: Possible position of TATA-box on DNA

2.3.2 Details about database

Our data is obtained from the internet (*http://www.fruitfly.org/sequence/human-datasets.html*)

The promoter data set is selected from the experimentally verified Eukaryotic Promoter Database (EPD) [30] published on the internet. There are 878 vertebrate promoter sequences among which about 70%~80% are believed to contain the TATA-box.

The non-promoter sequences contain 8000 cds (Coding DNA Sections) sequences and 8000 intr (non-coding (intron)) sequences. None of them contains TATA-box.

All sequences are of length of 51 — this corresponds to 51 nucleotides. Promoter sequences are from position -41 to position $+10$ relative to the TSS which is between the nucleotides with position -1 and $+1$.

Chapter 3

Possibility of motif prediction using cumulative PWM

In this chapter we will use the PWM obtained for the whole group of motifs, in an attempt to recognise the presence of one motif. Our specific example will be the TATA-box of eukaryotic promoters.

3.1 Algorithm

The original PWM for the TATA motif was obtained in [12] from the analysis of 502 unrelated promoters. It has 15 columns. To use this PWM in the search for the TATA motif, we will use a window of the length of 15 nucleotides to correspond to the 15 positions in the PWM. When we shift the window along the DNA sequence, say, from the upstream position of $-40\,bp$ relative to the TSS, we can match the 15 nucleotides in the current window, with the 15 weights (which correspond to probabilities for each specific nucleotide) from the PWM. For each possible position of the window, we calculate two scores represented by variables $score1$ and $score2$. These are obtained by applying addition, or multiplication, of all the respective 15 weights. As we assume that the core motif for the TATA-box (TATA hexamer) (see Figure $2-2$) is located in the range of

$-40\,bp$ to $-11bp$ relative to the TSS, and if we take into account that the weights for the core motive in the PWM are located from columns 2 to 7, then we will have in the promoter sequences of 51 nucleotides in length which span $-41\,bp$ to $+10\,bp$, in total 24 possible positions for the data window. The locations of the first nucleotide in the window will thus range from $-41\,bp$ to $-17\,bp$ relative to the TSS. This means that we can get 24 sets of scores for each sequence.

Then, we need to define a threshold by analyzing the obtained scores. The sequence for which the obtained scores are above the threshold will be considered as containing the TATA motif. We will also determine maximally one motif for each of the sequences.

3.2 Details about the algorithm

3.2.1 Calculation of the scores

Firstly, let us analyse one sequence. We slide 15 nucleotides long window from the first nucleotide of the sequence. For each window's position, we can observe 15 nucleotides contained in the window. Each nucleotide corresponds to one of the four letters: A, G, C, T. Then we match each letter and its position with the corresponding element on the PWM. For example, assume the first window of the sequence contains 15 nucleotides in the following order: A T T C G C T A C T G A A T C. Then, we get the corresponding weights for each of the letters according to Table 2.1. The respective weights are summarized in Table 3.1. We use p_i to denote weights of the letter at position i in the window.

Then, we apply addition and multiplication to these weights

$$\begin{aligned} score1(k,j) &= \sum_{i=1}^{15} p_i, \\ score2(k,j) &= \prod_{i=1}^{15} p_i, \end{aligned} \tag{3.1}$$

Position(i)	letter	weigth (probability) (p_i)
1	A	15.7
2	T	79.4
3	T	9
4	C	2.6
5	G	1.3
6	C	0
7	T	1.5
8	A	57.1
9	C	11.3
10	T	12.3
11	G	32.9
12	A	21.1
13	A	21.1
14	T	19.3
15	C	26

Table 3.1: Weights p_i corresponding to position i

where $score1(k, j)$ represents $score1$ for the position j of the data window for the k-th sequence in the dataset, $k = 1, 2, 3...$corresponds to the number of sequences in dataset used, while $j = 1, 2, ..., 24$ corresponds to the 24 possible positions of the window; p_i is the weight of the letter which is found at position i in the window, and $i = 1, 2, 3...15$.

3.2.2 The first assumptions

From Table 2.1 we can observe that there are some weights which are 0. This means that if such a weight corresponds to the nucleotide in the window, then $score2 = 0$ for the current window. This can be used to quickly eliminate the current position of the window as one that does not contain the hypothetical TATA motif. Thus:

- if $score2(k, j) = 0$, we can say immediately that there is no TATA motif on position j;

- if $score2(k,j) \neq 0$ then

  let

$$LEV_1(k) = max(score1(k,j)),\ LEV_2(k) = max(score2(k,j)) \tag{3.2}$$

and assume that, when

$$\begin{aligned} score1(k,j) &\geq k_1 * LEV_1(k), score2(k,j) \geq k_2 * LEV_2(k) \\ 0 &< k_1 < 1,\ 0 < k_2 < 1 \end{aligned} \tag{3.3}$$

there is a TATA motif match (15 nucleotides long) located at position j from the start of the sequence.

This assumption is implemented in the programs (see Appendix 8.1.1) to extract the TATA motifs from promoter sequences. Then, we can calculate true positive (TP) predictions for promoter data and false positive (FP) predictions for cds and intr data according to (3.4) and (3.5).

$$TP = \frac{M}{N} \times 100\% \tag{3.4}$$

where TP is the percentage of true positive predictions of promoter data, N is the number of sequences of promoter data set, and M is the number of TATA motifs containing sequences predicted from promoter data set, and

$$FP = \frac{M'}{N'} \times 100\% \tag{3.5}$$

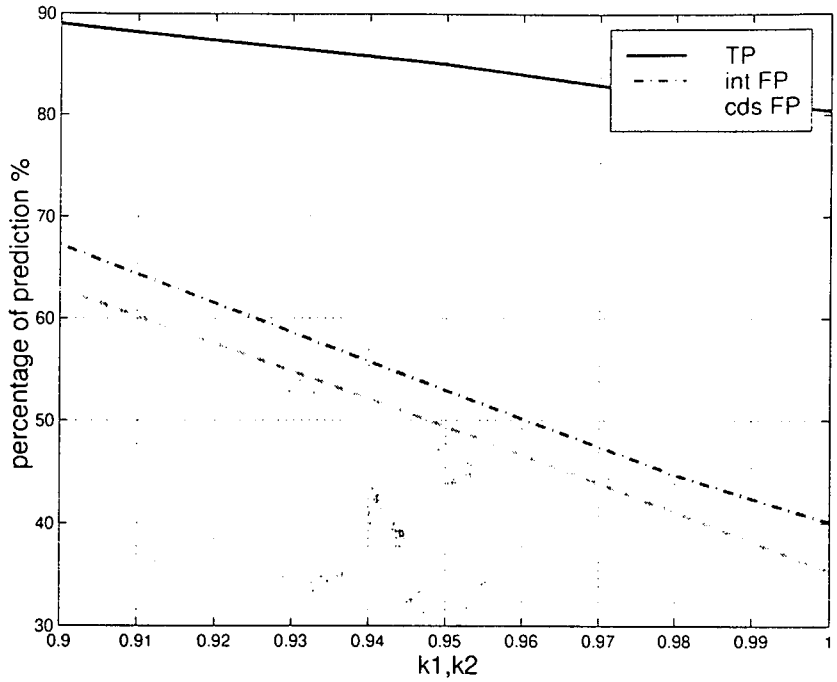

Figure 3-1: Predictions of TATA motif using only LEV1 and LEV2 and $k_1 = k_2$

where FP is the percentage of false positive predictions on non-promoter data, N' is the number of sequences of non-promoter data set, ans M' is the number of the TATA-box containing sequences predicted from the non-promoter data set.

With different values of coefficients $k1$ and $k2$ we get the results that are depicted in Figure $3-1$. Note that $k_1 = k_2$ is used, and the values ranged from 0.9 to 1.

The results shown in Figure $3-1$ are far from satisfactory. For the changes of k_1 and k_2 from 0.9 to 1, the TP of promoter sequences is in the range 80%~90%. However, only 70%~80% of promoter sequences contain functional TATA-boxes. The FP for intr sequences is about 40%~67%, while FP for cds sequences is about 35%~63%. It is obvious that we could not get any useful result by means of changing k_1 and k_2, as the level of FPs to high.

Thus we will add another assumption to attempt to improve the results. Before we do it, we need to analysis the distribution of the *score*1s.

3.2.3 Distribution of the *score*1 values

When we slide the window along one sequence, we get 24 *score*1s for this sequence. So, in total, we can get 24×878 *score*1s for promoter data (there is 878 promoter sequences in our dataset), 24×8000 *score*1s for either cds or intr data. We collect this data and analyse its distribution.

The distribution of the *score*1s for all the 878 selected promoter sequences, 8000 cds sequences and 8000 intr sequences is shown as Figure $3-2$. For promoter data, the *score*1s are distributed in the range of 128.6 to 855.8; for cds data, the *score*1s are distributed in the range of 148.3 to 794.3; for intron data, the *score*1s are distributed in the range of 136.3 to 764.

From Figure $3-2$ one can observe that, when *score*1 < 600, the number of promoter sequences recognized as containing TATA motif is even lower than for cds or intr. But when *score*1 > 600, the number of cds or intr sequences falsely recognized as containing TATA motif declines to zero much faster than the reduction of TP. This means if we take 600 as a threshold, only very few sequences in non-promoter data will be detected by the program as TATA containing sequences, i.e., FP will be very small; if we take some threshold above 600, then FP may be drop to zero. These facts are depicted in Figure $3-3$.

3.2.4 Another assumption

Analyzing Figure $3-3$ the following idea appears clearly: we can select a threshold in such a way that there are some promoter sequences with *score*1 higher than this selected threshold, while no *score*1 of non-promoter sequences will be above this threshold. Let $LEVscore$ denote this threshold. Using equation (3.3), we may now make another assumption:

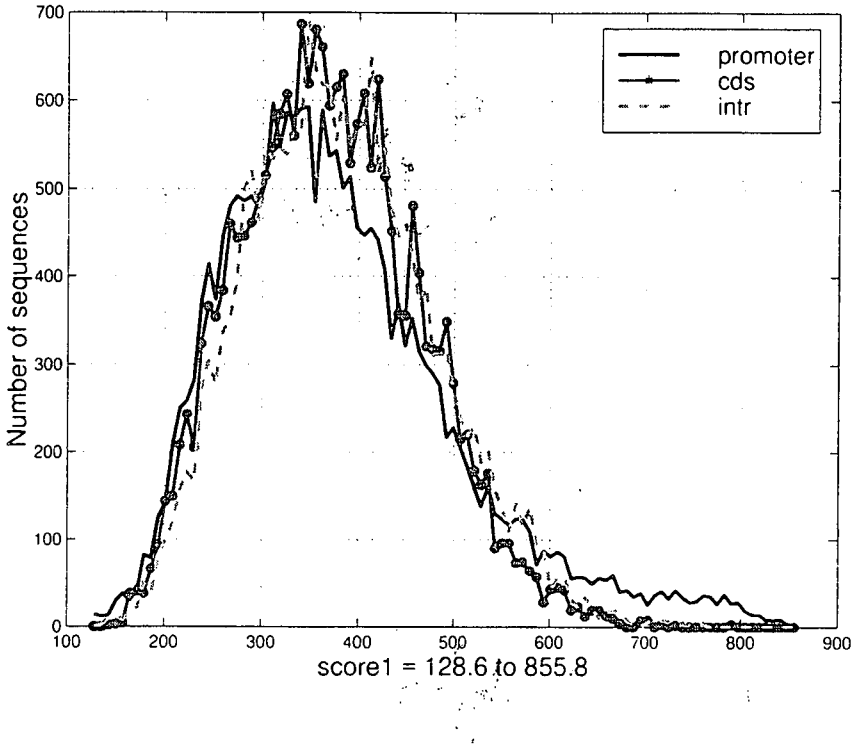

Figure 3-2: Distribution of score1 of promoter, cds and intr

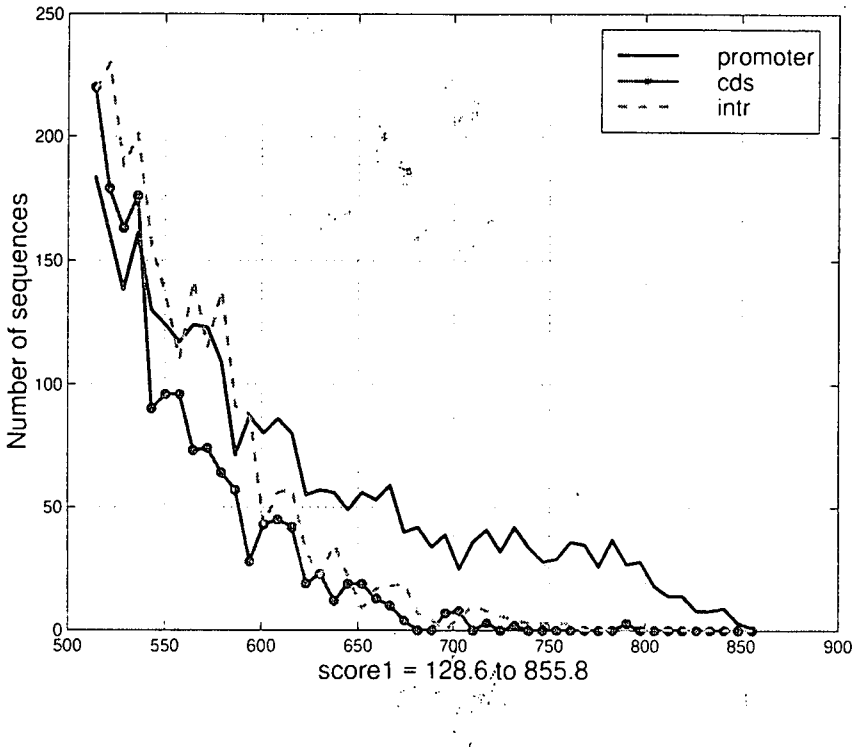

Figure 3-3: *score*1 for promoters, cds and intr data

- the necessary condition for the detection of the TATA motif is

$$score1(k, j) > LEVscore. \tag{3.6}$$

If both equation equations, (3.3) and (3.6), are satisfied, we can consider that the TATA motif (15 nucleotides long) is located on position j.

3.2.5 Results

With both assumptions in place, we made programs (Appendix 8.1.2) to predict TATA motif containing sequences from both 878 promoter sequences and all non-promoter sequences, i.e. 8000 cds sequences and 8000 intr sequences. Results are shown from Figure $3-4$ to Figure $3-7$.

For the results in Figure $3-4$, we select $k_1 = k_2 = 0.9$, and when the threshold changes from 600 to 795, TP declines from 76.4% to 12.8%, FP for intr sequences declines from 19.9% to 0%, while FP for cds declines from 15.6% to 0%. In fact, when threshold is 790, FP for intr sequences had already been 0%.

For the results depicted in Figure $3-5$, we select $k_1 = k_2 = 0.95$. When the threshold changes from 600 to 795, TP declines from 76.1% to 12.8%, FP for intr sequences declines from 19.5% to 0%, and FP for cds sequences declines from 14.8% to 0%. For the threshold if 790, FP for intr was already 0%.

Results in Figure $3-6$, are obtained with $k_1 = k_2 = 0.98$. When the threshold changes from 600 to 795, TP declines from 74.5% to 12.8%, FP for intr sequences declines from 18.9% to 0%,and FP for cds sequences declines from 13.6% to 0%. As previously, for the threshold of 790, FP for intr sequences had already become 0%.

Finally, results in Figure $3-7$ are obtained with $k_1 = k_2 = 1$. The change of the threshold from 600 to 795, resulted in TP change from 73.6% to 12.8%, FP for intr sequences declines from 18.2% to 0%, while FP for cds sequences declines from 12.9%

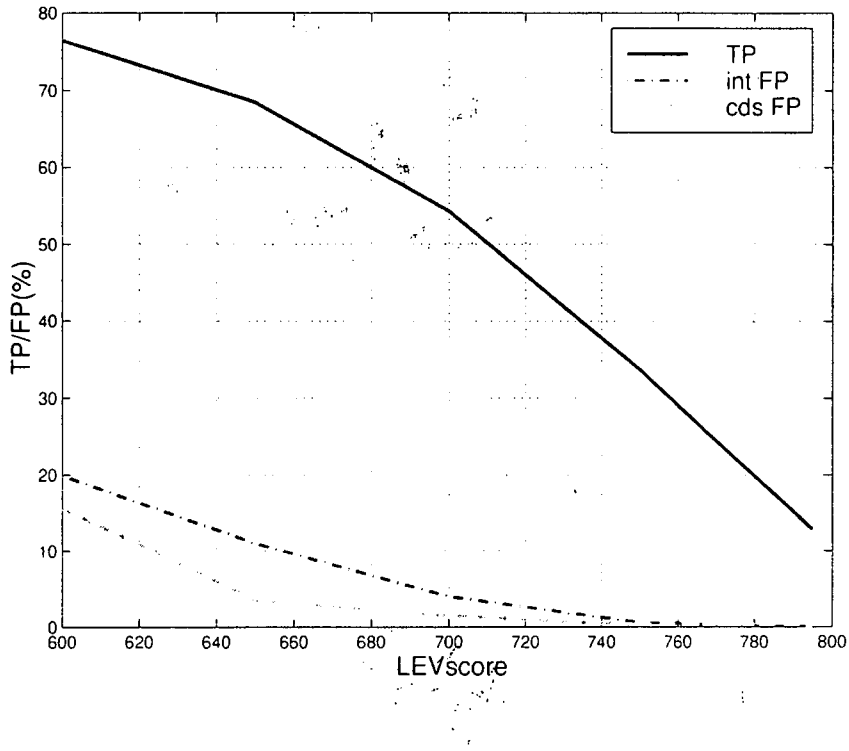

Figure 3-4: Prediction with $k_1 = k_2 = 0.9$

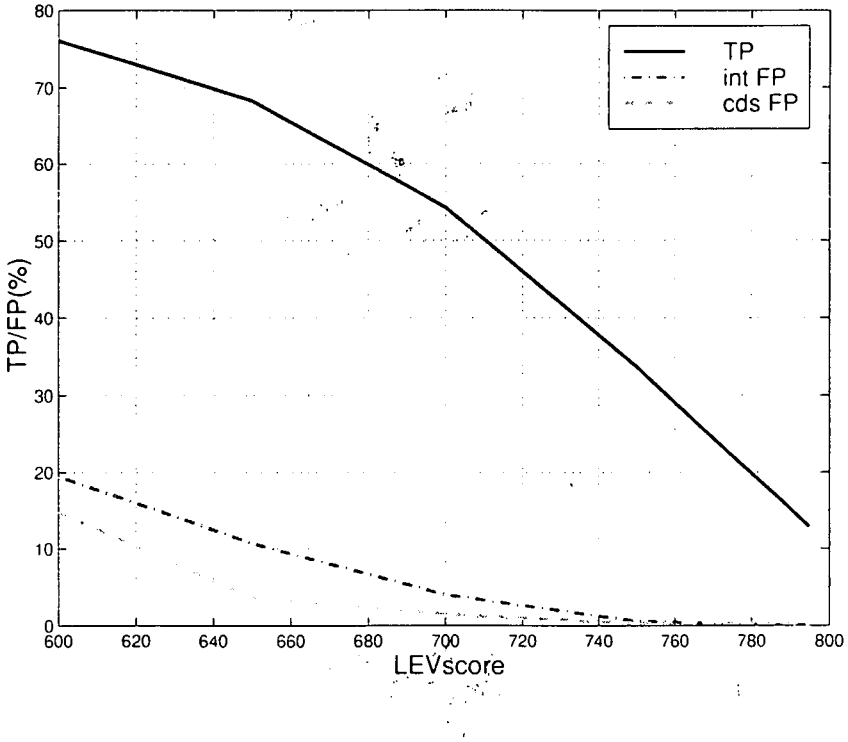

Figure 3-5: Prediction with $k_1 = k_2 = 0.95$

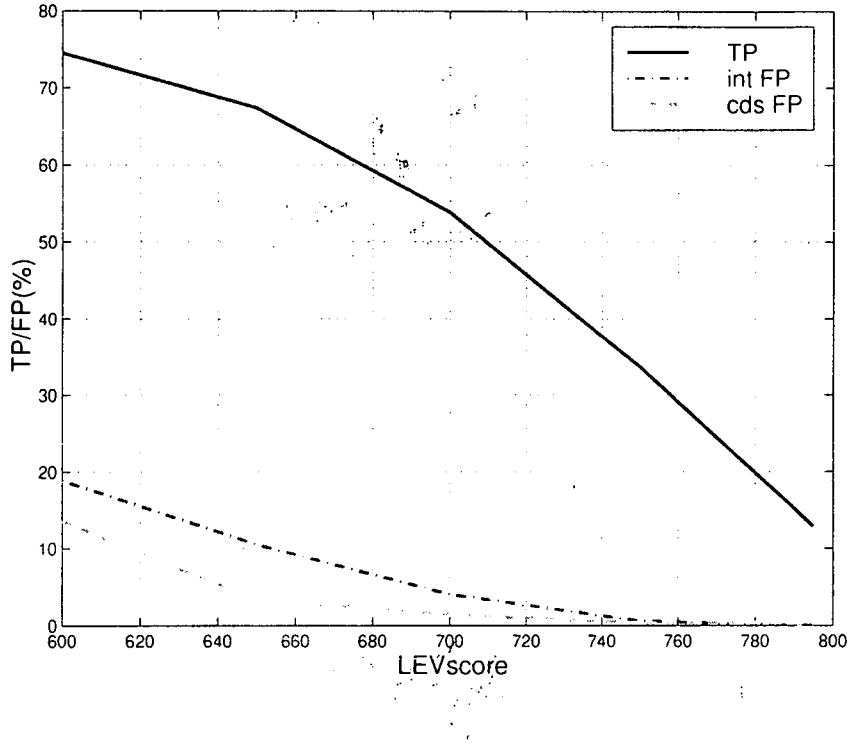

Figure 3-6: Prediction with $k_1 = k_2 = 0.98$

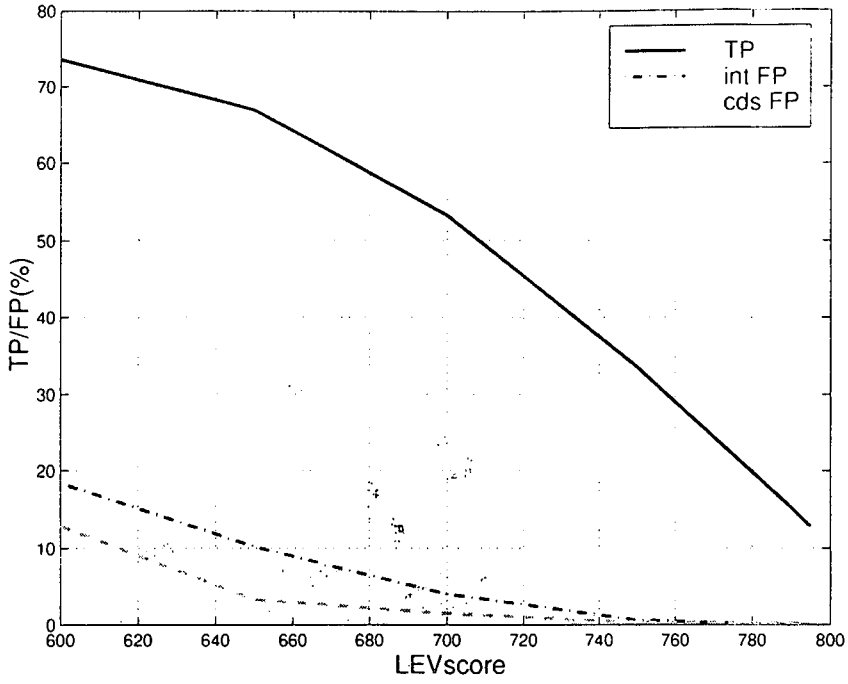

Figure 3-7: Prediction with $k_1 = k_2 = 1$

to 0%. For the threshold of 790, FP for intr sequences was already 0%.

These results show that parameters $k1$ and $k2$ do not affect the final recognition results too much. With different values for $k1$ and $k2$, we can get 0% FP for both cds and intr dataset, while TP for promoter dataset is 12.8%. We also conclude that we could not make FP drop to a reasonably low level and simultaneously achieving reasonably high TP.

Thus, we will have to use another algorithm to detect the presence of the TATA motif. This new algorithm will utilize positional distribution of the TATA-box.

Chapter 4

Development of a new model for TATA motif based on positional clustering

4.1 Algorithm

The original PWM for TATA-box describes statistically the content of the TATA-like motif in a large group of sequences. The four letters A, C, G, and T, can be at any position and their appearance is considered independent of each other. They can appear in any combination with the only exception being that C will never appear on positions $3, 5, 6$, while G will never be on position 6 of the 15 nucleotide-long TATA motif. When we used such a description, we obtained results as indicated in the previous chapter. These results were not very good, and thus we intend to improve the recognition results by developing a different algorithm for motif recognition. This is based on V. Bajić's idea that positional clustering will improve the accuracy of prediction by allowing more precise determination of the PWMs for each cluster. Here we develop that idea to apply it practically in TATA motif recognition.

Because the beginning of a TATA-box may locate at any position from $-40\,bp$ to

$-17\,bp$ upstream of the TSS, so, we will attempt clustering data into groups according to their location relative to the TSS. On this basis we will attempt to get more precise PWMs. Consequently, the TATA motif will be described not by only one PWM, but by a set of PWMs that correspond to the individual distances of the TATA motif from the TSS. In the ideal case of the PWM description, on one particular position only one kind of nucleotide will exist in the motif, and then the probability for one of the four letters A, C, G, T to appear at that position is 100%, while for others it is *zero*. This means, the ideal PWM will have only $1s$ and $0s$ as its elements. In principle, from our database, we can get 24 clusters of TATA motifs, where each cluster corresponds to one particular distance in the range from $-40\,bp$ to $-17\,bp$. This implies that we can obtain 24 new $PWMs$. Determination of the new $PWMs$ is based on the same algorithm as presented in the last chapter to extract TATA containing promoter sequences.

After new $PWMs$ are determined, we develop additional tools to enhance the motif recognition. In what follows, we denote the PWM for the j-th cluster by PWM_j. We still calculate $score1$s for these TATA containing promoter sequences which we have extracted from the promoter dataset. This time, we use new $PWMs$ instead of using the original one. So, we have

$$
\begin{aligned}
score1_{pm}(k,j) &= \sum_{i=1}^{15} p_i(j) \qquad (4.1)\\
k &= 1,2,3,\ldots \text{ corresponds to the number of the TATA-box}\\
&\quad \text{containing sequences in promoter data;}\\
j &= 1,2,3\ldots 24 \text{ corresponds to 24 clusters of TATA-box,}\\
&\quad \text{and also to 24 groups of } PWMs;\\
i &= 1,2,3\ldots 15;
\end{aligned}
$$

where $score1_{pm}(k,j)$ represents the score obtained from data in the window at position j and for the k-th sequence. In this calculation the j-th PWM is used (i.e. the PWM

obtained for data in the j-th cluster).

Then, we calculate the minimum and maximum of $score1_{pm}$ of each group:

$$LB_{pm}(j) = \min(score1_{pm}(.,j)), \qquad HB_{pm}(j) = \max(score1_{pm}(.,j)) \tag{4.2}$$

Now, we can elaborate on the new recognition algorithm:

Assume that a dataset contains k sequences and that each sequence contains at least a 38 nucleotide-long segment (this corresponds to the distance from the start point of the first position of the 15 nucleotide-long window to the end point of the last position of this window). We determine $score1(k,j)$ and $score2(k,j)$ with the corresponding PWM_j for each sequence

$$\begin{aligned} score1(k,j) &= \sum_{i=1}^{15} p_i(j), \qquad score2(k,j) = \prod_{i=1}^{15} p_i(j) \\ k &= 1,2,3,\ldots \text{ corresponds to the total number of sequences} \\ &\quad \text{in the dataset that is analysed;} \\ j &= 1,2,3...24; \\ i &= 1,2,3...15; \end{aligned} \tag{4.3}$$

where $score1(k,j)$ and $score2(k,j)$ represent $score1$ and $score2$, respectively, at position j and for the k-th sequence.

If $score2(k,j) = 0$, then consider that there is no TATA motif at position j for the k-th sequence.

Otherwise, if $score2(k,j) \neq 0$, then let

$$LB(j) = LB_{pm}(j), \qquad HB(j) = HB_{pm}(j), \tag{4.4}$$

and if

$$score1(k,j) \geq LB(j) \text{ and } score1(k,j) \leq HB(j) \tag{4.5}$$

then, the k-th sequence is considered to be a TATA containing sequence, and the TATA motif (15 nucleotides long) is located starting from the position j.

4.2 PWMs obtained from the training set

4.2.1 Determination of the PWMs from the training set

First of all, we divide the promoter data into two subsets, one of which is used for training and from which we get new $PWMs$ and $LB_{pm}(j)$, $HB_{pm}(j)$, $j = 1, 2, ..., 24$. The other is used to test for evaluation on how well TATA motifs are recognized.. The sequences for the training and the test sets are determined by random selection. The promoter training set we denote as $pAtr$ and it contains 640 sequences, while the promoter test set, that we denote as $pAts$, contains 238 promoter sequences. The negative set, the one that contains non-promoters (i.e. cds and intr sequences) contains 8000 cds and 8000 ints sequences.

The programs to determine the new $PWMs$ from training set $pAtr$ are given in the Appendix 8.1.3. We still use the same assumptions as in the previous chapter, i.e. equation (3.3) and equation (3.6) to extract the TATA-box containing sequences. But we have to determine what $LEVscore$ and k_1, k_2 we should select.

From Figure $3-4$ to Figure $3-7$ one can observe that, when $LEVscore = 650$, FP for the cds and intr sequences declines a lot, while TP for promoters stays relatively high. So, in our programs from now on, we chose 650 as the value for the $LEVscore$ to extract TATA-box containing sequences. Since $k1$ and $k2$ do not significantly affect the results, we select them as $k_1 = k_2 = 0.98$.

Finally, after the TATA-box containing sequences for all 24 groups are extracted, we

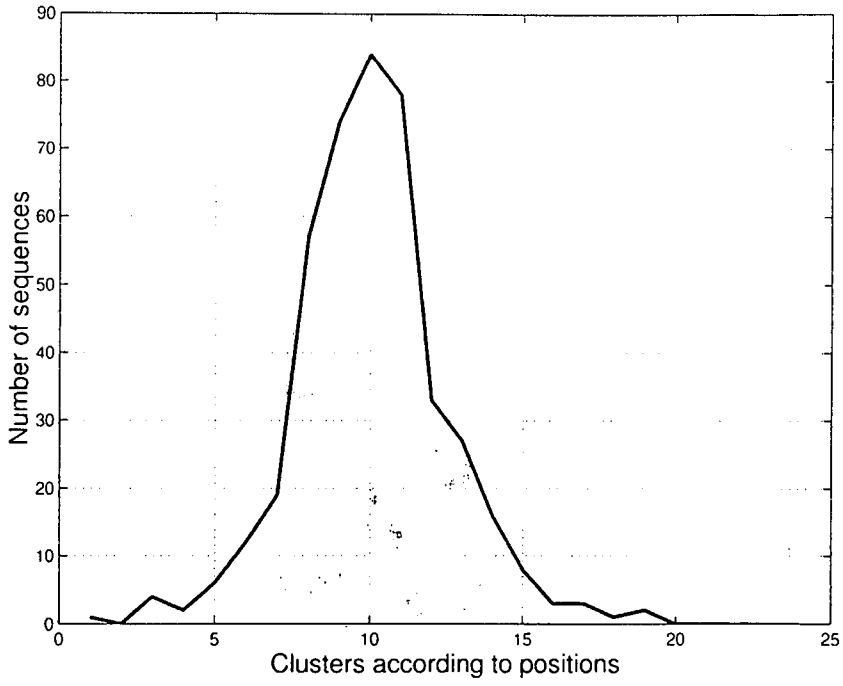

Figure 4-1: Distribution of TATA-box containing sequences according to position from the start of the promoter sequence (all promoter sequences are from -41 to $+10$ realtive to the TSS).

present their positional distribution in Figure $4-1$.

From Figure $4-1$ we can see, no sequence is found in groups 2, 20, 21, 22, 23, 24 which correspond to position -39, -21, -20, -19, -18, -17, relative to the TSS. This means we can only have 18 positional clusters of the TATA motif instead of 24 as initially assumed. Consequently, we can only obtain 18 new $PWMs$.

Now, we calculate new $PWMs$ for the 18 clusters using the following equation:

Position	%A	%C	%G	%T
1	0	100	0	0
2	0	0	0	100
3	0	0	0	100
4	0	0	0	100
5	100	0	0	0
6	100	0	0	0
7	100	0	0	0
8	0	0	100	0
9	100	0	0	0
10	0	100	0	0
11	100	0	0	0
12	0	0	100	0
13	0	100	0	0
14	100	0	0	0
15	100	0	0	0

Table 4.1: PWM_1 for position -40

$$
\begin{aligned}
p^j_{qi} &= \frac{N_{qi}}{N^j} \times 100\% \\
q &= A, C, G, T \\
i &= 1, 2, 3...15 \\
j &= 1, 2, 3....24
\end{aligned}
$$

where p^j_{qi} represents the probability for letter q to appear in position i in PWM_j; N^j represents the number of sequences in the cluster j; N_{qi} represents the number of sequences which have the letter q in position i.

The new $PWMs$ are shown below from Table 4.1 - 4.18. Note that PWM_2, PWM_{20}, PWM_{21}, PWM_{22}, PWM_{23}, PWM_{24} are zero matrices since there were no sequences found on the respective positions.

When we compare these $PWMs$ with the original one, we observe that the least discriminative one (PWM_{10}, Table 4.9) has more zeros than the original one. This

Position	%A	%C	%G	%T
1	0	0	100	0
2	0	0	0	100
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	25	0	0	75
7	75	0	25	0
8	50	0	25	25
9	100	0	0	0
10	0	25	75	0
11	0	25	50	25
12	0	50	0	50
13	0	50	25	25
14	0	25	0	75
15	0	25	75	0

Table 4.2: PWM_3 for position -38

Position	%A	%C	%G	%T
1	0	50	0	50
2	0	0	0	100
3	50	0	50	0
4	0	0	0	100
5	100	0	0	0
6	0	0	0	100
7	100	0	0	0
8	50	0	0	50
9	0	0	100	0
10	50	0	50	0
11	0	50	50	0
12	0	100	0	0
13	0	100	0	0
14	0	100	0	0
15	50	0	0	50

Table 4.3: PWM_4 for position -37

Position	%A	%C	%G	%T
1	0	66.67	33.33	0
2	16.67	16.67	0	66.67
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	100	0	0	0
7	100	0	0	0
8	50	16.67	16.67	16.67
9	16.67	16.67	66.67	0
10	33.33	0	66.67	0
11	16.67	33.33	16.67	33.33
12	0	33.33	50	16.67
13	16.67	33.33	33.33	16.67
14	0	50	33.33	16.67
15	16.67	33.33	16.67	33.33

Table 4.4: PWM_5 for position -36

Position	%A	%C	%G	%T
1	25	50	25	0
2	16.67	8.33	8.33	66.67
3	75	0	0	25
4	0	0	0	100
5	91.67	0	0	8.33
6	83.33	0	0	16.67
7	91.67	8.33	0	0
8	58.33	0	8.33	33.33
9	50	16.67	25	8.33
10	8.33	25	41.67	25
11	25	8.33	41.67	25
12	8.33	50	33.33	8.33
13	16.67	41.67	25	16.67
14	16.67	25	41.67	16.67
15	8.33	50	25	16.67

Table 4.5: PWM_6 for position -35

Position	%A	%C	%G	%T
1	15.79	31.58	42.11	10.53
2	5.26	5.26	5.26	84.21
3	100	0	0	0
4	0	0	0	100
5	94.74	0	0	5.26
6	63.16	0	0	36.84
7	84.21	0	15.79	0
8	52.63	0	5.26	42.11
9	42.11	0	52.63	5.26
10	21.05	47.37	31.58	0
11	42.11	15.79	26.32	15.79
12	21.05	10.53	63.16	5.26
13	10.53	36.84	26.32	26.32
14	31.58	21.05	31.58	15.79
15	15.79	21.05	57.89	5.26

Table 4.6: PWM_7 for position -34

Position	%A	%C	%G	%T
1	12.28	42.11	38.6	7.02
2	1.75	28.07	0	70.18
3	94.74	0	0	5.26
4	0	1.75	1.75	96.49
5	82.46	0	1.75	15.79
6	71.93	0	0	28.07
7	94.74	0	5.26	0
8	75.44	0	5.26	19.3
9	36.84	1.75	56.14	5.26
10	28.07	21.05	42.11	8.77
11	28.07	36.84	22.81	12.28
12	35.09	29.82	26.32	8.77
13	19.3	22.81	42.11	15.79
14	31.58	21.05	38.6	8.77
15	14.04	28.07	35.09	22.81

Table 4.7: PWM_8 for position -33

Position	%A	%C	%G	%T
1	10.81	35.14	48.65	5.41
2	12.16	10.81	4.05	72.97
3	94.59	0	0	5.41
4	0	1.35	0	98.65
5	95.95	0	0	4.05
6	75.68	0	0	24.32
7	97.3	0	1.35	1.35
8	59.46	0	16.22	24.32
9	27.03	9.46	55.41	8.11
10	14.86	28.38	51.35	5.41
11	25.68	32.43	27.03	14.86
12	17.57	31.08	40.54	10.81
13	27.03	28.38	31.08	13.51
14	29.73	29.73	22.97	17.57
15	16.22	16.22	44.59	22.97

Table 4.8: PWM_9 for position -32

Position	%A	%C	%G	%T
1	17.86	46.43	28.57	7.14
2	4.76	9.52	3.57	82.14
3	86.9	0	1.19	11.9
4	0	4.76	0	95.24
5	96.43	0	1.19	2.38
6	76.19	0	0	23.81
7	95.24	1.19	2.38	1.19
8	57.14	0	13.1	29.76
9	26.19	7.14	57.14	9.52
10	7.14	44.05	39.29	9.52
11	20.24	25	41.67	13.1
12	11.9	41.67	40.48	5.95
13	20.24	41.67	27.38	10.71
14	23.81	30.95	29.76	15.48
15	15.48	22.62	41.67	20.24

Table 4.9: PWM_{10} for position -31

Position	%A	%C	%G	%T
1	12.82	43.59	37.18	6.41
2	1.28	12.82	2.56	83.33
3	93.59	0	0	6.41
4	0	0	0	100
5	96.15	0	0	3.85
6	73.08	0	0	26.92
7	92.31	0	5.13	2.56
8	53.85	5.13	21.79	19.23
9	20.51	20.51	51.28	7.69
10	19.23	29.49	37.18	14.1
11	25.64	26.92	41.03	6.41
12	12.82	34.62	39.74	12.82
13	24.36	33.33	26.92	15.38
14	12.82	37.18	32.05	17.95
15	17.95	24.36	39.74	17.95

Table 4.10: PWM_{11} for position -30

Position	%A	%C	%G	%T
1	15.15	42.42	36.36	6.06
2	9.09	3.03	3.03	84.85
3	90.91	0	0	9.09
4	0	0	0	100
5	96.97	0	0	3.03
6	66.67	0	0	33.33
7	87.88	3.03	3.03	6.06
8	51.52	0	33.33	15.15
9	24.24	18.18	48.48	9.09
10	15.15	24.24	39.39	21.21
11	33.33	33.33	30.3	3.03
12	21.21	24.24	45.45	9.09
13	9.09	21.21	42.42	27.27
14	15.15	33.33	42.42	9.09
15	15.15	27.27	57.58	0

Table 4.11: PWM_{12} for position -29

Position	%A	%C	%G	%T
1	7.41	48.15	37.04	7.41
2	3.7	14.81	3.7	77.78
3	92.59	0	0	7.41
4	0	0	0	100
5	96.3	0	0	3.7
6	81.48	0	0	18.52
7	92.59	0	7.41	0
8	51.85	0	25.93	22.22
9	29.63	11.11	51.85	7.41
10	22.22	7.41	66.67	3.7
11	37.04	29.63	29.63	3.7
12	11.11	37.04	37.04	14.81
13	18.52	33.33	29.63	18.52
14	11.11	40.74	22.22	25.93
15	29.63	22.22	37.04	11.11

Table 4.12: PWM_{13} for position -28

Position	%A	%C	%G	%T
1	25	50	18.75	6.25
2	0	6.25	0	93.75
3	87.5	0	0	12.5
4	0	0	0	100
5	93.75	0	6.25	0
6	56.25	0	0	43.75
7	87.5	0	12.5	0
8	62.5	0	6.25	31.25
9	43.75	0	56.25	0
10	0	31.25	25	43.75
11	18.75	25	43.75	12.5
12	25	31.25	37.5	6.25
13	12.5	43.75	25	18.75
14	25	37.5	31.25	6.25
15	25	18.75	25	31.25

Table 4.13: PWM_{14} for position -27

Position	%A	%C	%G	%T
1	25	37.5	12.5	25
2	12.5	0	0	87.5
3	75	0	0	25
4	0	0	0	100
5	100	0	0	0
6	87.5	0	0	12.5
7	100	0	0	0
8	50	0	25	25
9	37.5	0	62.5	0
10	0	62.5	37.5	0
11	25	50	12.5	12.5
12	25	50	25	0
13	0	37.5	25	37.5
14	25	12.5	50	12.5
15	12.5	12.5	50	25

Table 4.14: PWM_{15} for position -26

Position	%A	%C	%G	%T
1	33.33	66.67	0	0
2	0	0	0	100
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	66.67	0	0	33.33
7	100	0	0	0
8	33.33	0	66.67	0
9	33.33	0	66.67	0
10	0	66.67	0	33.33
11	33.33	33.33	33.33	0
12	0	33.33	66.67	0
13	0	33.33	0	66.67
14	33.33	0	66.67	0
15	0	66.67	33.33	0

Table 4.15: PWM_{16} for position -25

Position	%A	%C	%G	%T
1	33.33	0	66.67	0
2	0	33.33	0	66.67
3	66.67	0	0	33.33
4	0	0	0	100
5	100	0	0	0
6	100	0	0	0
7	100	0	0	0
8	33.33	0	33.33	33.33
9	0	33.33	33.33	33.33
10	0	33.33	66.67	0
11	33.33	0	66.67	0
12	0	66.67	0	33.33
13	33.33	33.33	33.33	0
14	0	0	100	0
15	0	33.33	66.67	0

Table 4.16: PWM_{17} for position -24

Position	%A	%C	%G	%T
1	100	0	0	0
2	100	0	0	0
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	100	0	0	0
7	100	0	0	0
8	100	0	0	0
9	0	0	100	0
10	0	100	0	0
11	100	0	0	0
12	0	0	100	0
13	0	0	100	0
14	0	0	0	100
15	0	0	100	0

Table 4.17: PWM_{18} for position -23

Position	%A	%C	%G	%T
1	50	0	50	0
2	0	0	100	0
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	50	0	0	50
7	100	0	0	0
8	50	50	0	0
9	0	50	50	0
10	0	50	50	0
11	50	50	0	0
12	50	0	50	0
13	0	0	100	0
14	0	0	50	50
15	0	100	0	0

Table 4.18: PWM_{19} for position -22

indicates that the new $PWMs$ are more specific (and thus more accurate) in describing properties of the TATA motifs, than when only one PWM is used.

4.2.2 Min-max *score*1 of TATA-box containing sequences

First, we extract the TATA-box containing sequences for each cluster. Then, for each cluster we determine the min-max of *score*1s for them, i.e. $LB_{pm}(j)$ and $HB_{pm}(j)$ by using equation (4.2). $LB_{pm}(j)$ and $HB_{pm}(j)$ are shown in Figure $4-2$. The computer program, that predicts the TATA containing sequences in *pAtr*, *pAtst*, *cds* and *int* sequence sets, based on these bounds and equations (4.4) and (4.5), is given in Appendix 8.1.4. The results of prediction are shown in Table 4.19.

With results from Table 4.19 and equations (3.4), (3.5), we obtain $TP = 50\%$ for *pAtst*, $TP = 69.37\%$ for *pAtr*, $FP = 2.17\%$ for *intr* sequences, and $FP = 0.96\%$ for *cds* sequences. This is much better than what we have got in Chapter 3 (as depicted from Figure $3-4$ to Figure $3-7$) with the original PWM where when for cds sequences $FP \approx 1\%$, while $FP \approx 3\%$ for intr sequences, and $TP \approx 40\%$.

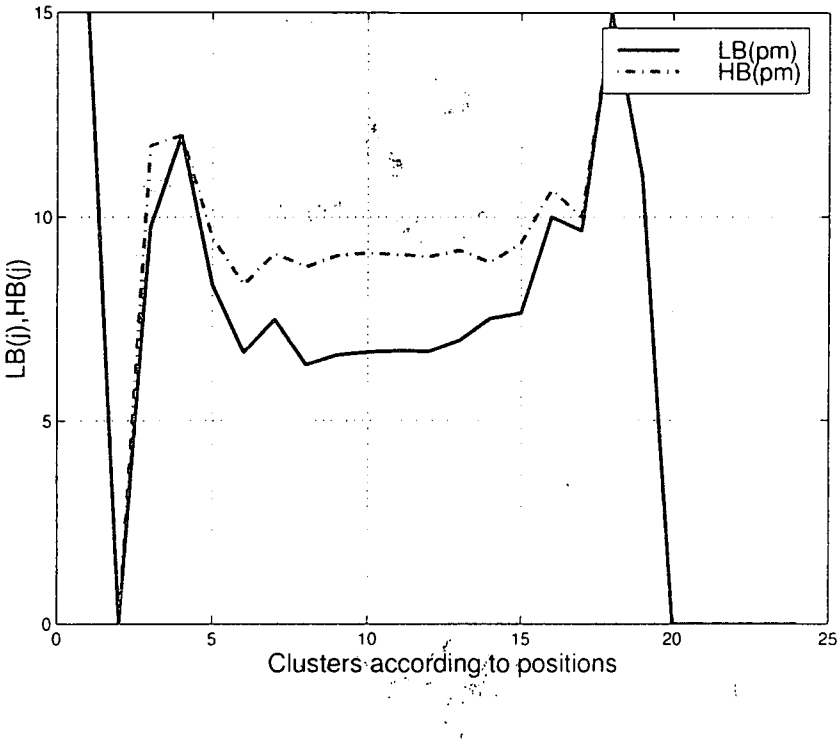

Figure 4-2: Bounds of TATA-box containing sequences of $pAtr$

Group(position)	$LB_{PM}(j)$	$HB_{PM}(j)$	pAtr	pAtst	intr	cds
1(-40)	15	15	1	0	0	0
3(-38)	9.75	11.75	4	0	0	0
4(-37)	12	12	2	0	0	0
5(-36)	8.3333	9.5	6	0	1	0
6(-35)	6.6667	8.3333	16	0	19	8
7(-34)	7.4737	9.1053	20	5	7	1
8(-33)	6.3684	8.7719	68	4	50	25
9(-32)	6.6081	9.0541	80	17	22	14
10(-31)	6.6786	9.119	86	20	22	10
11(-30)	6.7051	9.0769	73	30	23	6
12(-29)	6.697	9.0303	31	16	12	6
13(-28)	6.963	9.1852	13	14	14	4
14(-27)	7.5	8.875	14	11	2	2
15(-26)	7.625	9.375	6	1	2	1
16(-25)	10	10.6667	3	1	0	0
17(-24)	9.6667	10	3	0	0	0
18(-23)	15	15	1	0	0	0
19(-22)	11	11	2	0	0	0

Table 4.19: Prediction with LB(PM) and HB(PM) as bounds

4.2.3 More strict min-max *score*1 results in better prediction

Our goal is to reduce the FP as much as possible. The best result is that the FP for both intr and cds are 0%, while simultaneously TP is kept relatively high. Using equation (4.5) as criterion to reduce FP, we have to analyze the bounds of *score*1 of both TATA-box containing sequences and non-promoter sequences.

Firstly, we need to know min-max *score*1 of cds and intr data, i.e., $LB_{cds}(j)$ and $HB_{cds}(j)$ for cds data, and $LB_{int}(j)$ and $HB_{int}(j)$ for intr data. With equation (4.6) we calculate all *score*1s and *score*2s for cds data set which are denoted by $score1_{cds}$; $score2_{cds}$, and stored in two matrices with k_{cds} rows and 24 columns.

$$
\begin{aligned}
score1_{cds}(k_{cds}, j) &= \sum_{i=1}^{15} p_i, \\
score2_{cds}(k_{cds}, j) &= \prod_{i=1}^{15} p_i, \\
i &= 1, 2, 3...15, \\
j &= 1, 2, 3...24, \\
k_{cds} &= 1, 2, 3...8000,
\end{aligned} \tag{4.6}
$$

where k_{cds} corresponds to the number of sequences in cds sequence data set.

With equation (4.7), we can get all $score1_{int}$ and $score2_{int}$ for intr data set which are in two k_{int} rows j columns matrices respectively

$$
\begin{aligned}
score1_{int}(k_{int}, j) &= \sum_{i=1}^{15} p_i, \\
score2_{int}(k_{int}, j) &= \prod_{i=1}^{15} p_i, \\
i &= 1, 2, 3...15, \\
j &= 1, 2, 3...24, \\
k_{int} &= 1, 2, 3...8000,
\end{aligned} \tag{4.7}
$$

where k_{int} corresponds to the number of sequences in intr sequence data set.

With the same idea as when we derived $LB_{pm}(j)$ and $HB_{pm}(j)$ from $pAtr$, with equation(4.8)

$$
LB_{cds}(j) = \min(score1_{cds}(., j)), \quad HB_{cds}(j) = \max(score1_{cds}(., j)) \tag{4.8}
$$

Group(j) (Position)	LB_{cds}	HB_{cds}	LB_{int}	HB_{int}
1(-40)	NaN	NaN	NaN	NaN
2(-39)	NaN	NaN	NaN	NaN
3(-38)	NaN	NaN	NaN	NaN
4(-37)	NaN	NaN	NaN	NaN
5(-36)	NaN	NaN	8.6667	8.6667
6(-35)	4.0833	7.25	3.5833	8.1667
7(-34)	4.3158	9	4.5263	8.1053
8(-33)	2.3509	7.9825	1.7018	7.7368
9(-32)	2.1622	8.3514	2	7.973
10(-31)	2.0476	8.119	1.6667	7.9405
11(-30)	2.859	8.141	2.7436	7.7436
12(-29)	3.0606	8.4242	3.4242	8.0909
13(-28)	3.0741	7.9259	2.8889	8.2963
14(-27)	4.8125	7.875	4.375	7.75
15(-26)	7	8	7	8.125
16(-25)	NaN	NaN	NaN	NaN
17(-24)	NaN	NaN	NaN	NaN
18(-23)	NaN	NaN	NaN	NaN
19(-22)	NaN	NaN	NaN	NaN
20(-21)	NaN	NaN	NaN	NaN
21(-20)	NaN	NaN	NaN	NaN
22(-19)	NaN	NaN	NaN	NaN
23(-18)	NaN	NaN	NaN	NaN
24(-17)	NaN	NaN	NaN	NaN

Table 4.20: Min-max score1 for cds and int respectively

we get $LB_{cds}(j)$, $HB_{cds}(j)$ from cds data set, and with equation (4.9)

$$LB_{int}(j) = \min(score1_{int}(.,j)), \quad HB_{int}(j) = \max(score1_{int}(.,j)) \tag{4.9}$$

we get $LB_{int}(j), HB_{int}(j)$ from intr data set. The results of recognition res shown as Table 4.20. The NaN indicates that the programs did not find any sequence on respective positions and thus, it was not possible to determine the lower or upper bounds.

We can make a comparison between bounds of $score1$s of TATA-box containing pro-

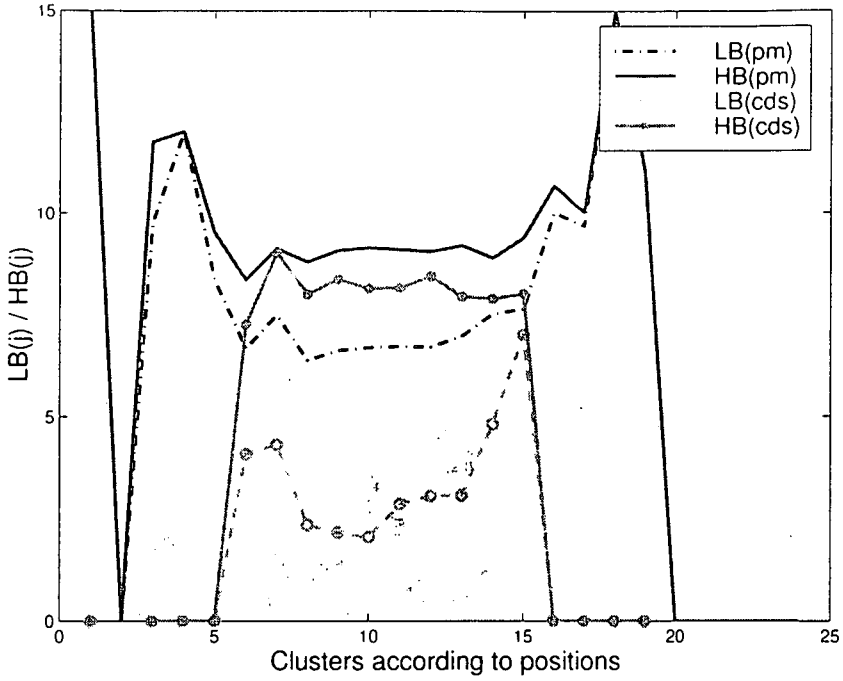

Figure 4-3: Comparison of bounds of *score*1 of TATA containing sequences, and cds sequences

moter sequences, i.e. $LB_{pm}(j)$ and $HB_{pm}(j)$, and bounds of *score*1s of cds sequences, i.e. $LB_{cds}(j)$ and $HB_{cds}(j)$, as given in Figure $4-3$; we will also make a comparison between bounds of *score*1s of TATA-box containing promoter sequences, i.e. $LB_{pm}(j)$ and $HB_{pm}(j)$, and bounds of *score*1s of intr sequences, i.e. $LB_{int}(j)$ and $HB_{int}(j)$, as in Figure $4-4$.

From Figure $4-3$ one can observe that if the low bound $LB(j)$ in equation (4.5) is changed it may reduce FP of cds dataset. And we need to change these bounds for groups 6 to 15. $LB(j)$ for these groups should have maximal value between $HB_{cds}(j)$ and $LB_{pm}(j)$. From Figure $4-4$ we conclude that we also only need to change low bound $LB(j)$ for group 5 to group 15. $LB(j)$ for these groups should be the maximal value between $HB_{int}(j)$ and $LB_{pm}(j)$. We tune $LB(j)$ one by one according to the order

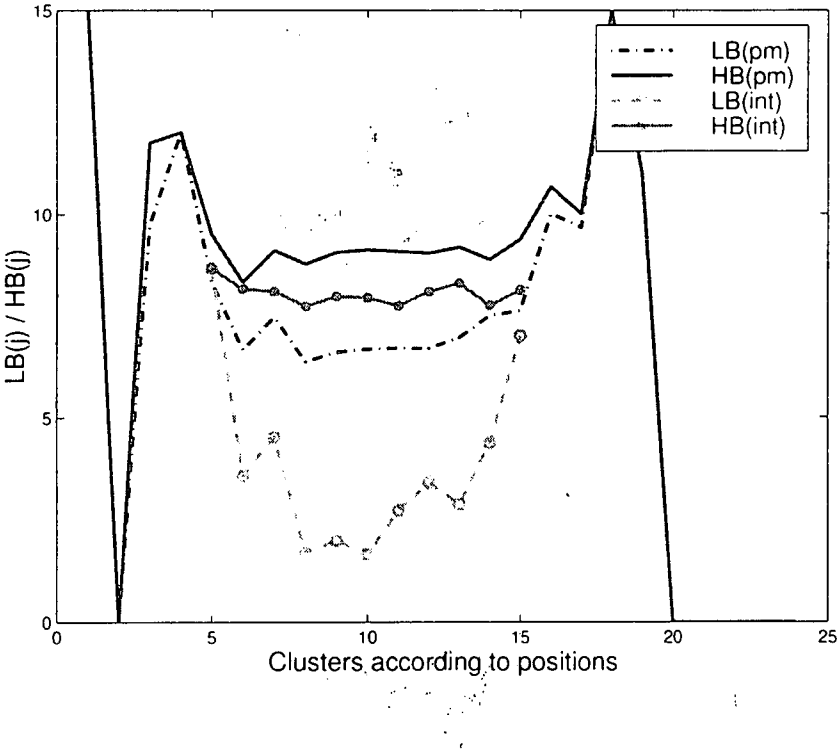

Figure 4-4: Comparison of bounds of *score*1 of TATA containing sequences, and intr sequences

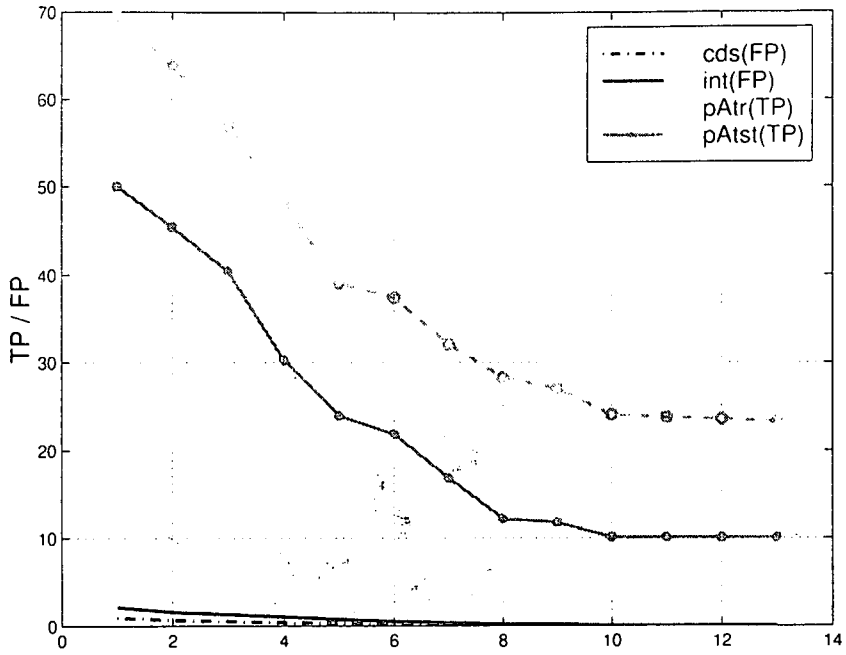

Figure 4-5: Prediction with new PWMs

which causes FP to drop as fast as possible, while simultaneously keeps TP as high as possible. So, according to (4.19), firstly, we tune $LB(8)$, then group 9, 10, 11, 6, 12, 13, 14, 7, 15, 16, 5. Eventually, we get the result as shown both in Figure $4-5$ and Table 4.21.

pAtr(%)	pAtst(%)	cds(%)	int(%)
69.37	50	0.96	2.17
63.91	45.38	0.675	1.59
56.88	40.38	0.525	10325
48.13	30.25	0.3875	1.05
39.06	23.95	0.3125	0.7375
37.34	21.85	0.2125	0.5
32.03	16.81	0.125	0.3375
28.28	12.18	0.05	0.1628
27.03	11.76	0.025	0.125
24.06	10.08	0.0125	0.0375
23.75	10.08	0	0.0125
23.59	10.08	0	0.0125
23.28	10.08	0	0

Table 4.21: Prediction result with new PWMs

Chapter 5

Development of new PWMs from the whole promoter dataset

5.1 PWMs from whole promoter dataset

In the previous chapter we used only a portion of promoter data ($pAtr$) to derive a new model of the TATA motif. However, it may be possible to get a better description of the TATA motif if we use the whole promoter dataset, since that set is larger. We apply the same algorithm as we did in the previous chapter to get the new $PWMs$ from the whole promoter data set. This time, we get 20 groups because there are TATA-box containing sequences found on position -39 and -21. The results are shown in Tables 5.1 to Table 5.17:

5.2 Prediction by using min-max $score1$ of TATA-box containing sequences

Now we calculate the $LB_{pm}(j)$ and $HB_{pm}(j)$ with equation (4.2) for each group. The result is shown in Figure $5-1$.

PWM1 for position -40					PWM2 for position -39				
Position	%A	%C	%G	%T	Position	%A	%C	%G	%T
1	0	100	0	0	1	0	50	50	0
2	0	0	0	100	2	0	0	0	100
3	0	0	0	100	3	100	0	0	0
4	0	0	0	100	4	0	0	0	100
5	100	0	0	0	5	50	0	0	50
6	100	0	0	0	6	50	0	0	50
7	100	0	0	0	7	50	0	0	50
8	0	0	100	0	8	0	0	50	50
9	100	0	0	0	9	0	50	50	0
10	0	100	0	0	10	50	50	0	0
11	100	0	0	0	11	0	0	100	0
12	0	0	100	0	12	100	0	0	0
13	0	100	0	0	13	0	50	50	0
14	100	0	0	0	14	50	0	50	0
15	100	0	0	0	15	0	50	50	0

Table 5.1: PWM1 and PWM2 for position -40, -39

PWM3 for position -38					PWM4 for position -37				
Position	%A	%C	%G	%T	Position	%A	%C	%G	%T
1	0	20	80	0	1	0	50	0	0
2	0	0	0	100	2	0	0	0	100
3	80	0	0	20	3	50	0	50	0
4	0	0	0	100	4	0	0	0	100
5	100	0	0	0	5	100	0	0	0
6	40	0	0	60	6	0	0	0	100
7	80	0	20	0	7	100	0	0	0
8	40	0	40	20	8	50	0	0	50
9	80	0	20	0	9	0	0	100	0
10	0	20	80	0	10	50	0	50	0
11	0	20	60	20	11	0	50	50	0
12	20	40	0	40	12	0	100	0	0
13	0	40	40	20	13	0	100	0	0
14	20	20	0	60	14	0	100	0	0
15	0	20	60	20	15	50	0	0	50

Table 5.2: PWM3 and PWM4 for position -38, -37

Position	%A	%C	%G	%T
1	0	75	25	0
2	12.5	12.5	0	75
3	100	0	0	0
4	12.5	0	0	87.5
5	87.5	0	0	12.5
6	87.5	0	0	12.5
7	100	0	0	0
8	50	12.5	25	12.5
9	25	12.5	62.5	0
10	37.5	0	62.5	0
11	25	25	25	25
12	0	37.5	50	12.5
13	37.5	25	25	12.5
14	0	37.5	37.5	25
15	12.5	25	37.5	25

Table 5.3: PWM_5 for position -36

Position	%A	%C	%G	%T
1	29.41	41.18	23.53	5.88
2	11.76	5.88	5.88	76.47
3	82.35	0	0	17.65
4	0	0	0	100
5	76.47	0	0	23.53
6	82.35	0	0	17.65
7	88.24	5.88	0	5.88
8	52.94	0	5.88	41.18
9	58.82	17.65	17.65	5.88
10	11.76	23.53	41.18	23.53
11	29.41	17.65	35.29	17.65
12	5.88	58.82	29.41	5.88
13	17.65	47.06	23.53	11.76
14	11.76	41.18	29.41	17.65
15	11.76	47.06	29.41	11.76

Table 5.4: PWM_6 for position -35

Position	%A	%C	%G	%T
1	12	24	48	16
2	4	4	4	88
3	100	0	0	0
4	0	0	0	100
5	88	0	4	8
6	64	0	0	36
7	88	0	12	0
8	48	0	4	48
9	52	0	44	4
10	16	44	36	4
11	44	16	28	12
12	20	12	52	16
13	12	44	24	20
14	28	24	28	20
15	24	20	52	4

Table 5.5: PWM_7 for position -34

Position	%A	%C	%G	%T
1	14.29	35.71	42.86	7.14
2	2.86	24.29	2.86	70
3	92.86	0	0	7.14
4	0	2.86	1.43	95.71
5	81.43	0	1.43	17.14
6	71.43	0	0	28.57
7	94.29	0	5.71	0
8	77.14	0	4.29	18.57
9	34.29	2.86	54.29	8.57
10	25.71	24.29	41.43	8.57
11	24.29	38.57	24.29	12.86
12	32.86	31.43	24.29	11.43
13	20	24.29	41.43	14.29
14	30	22.86	35.71	11.43
15	15.71	27.14	35.71	21.43

Table 5.6: PWM_8 for position -33

Position	%A	%C	%G	%T
1	12.24	33.67	48.98	5.1
2	11.22	10.2	6.12	72.45
3	94.9	0	0	5.1
4	0	3.06	0	96.94
5	90.82	0	0	9.18
6	68.37	0	0	31.63
7	97.96	0	1.02	1.02
8	61.22	0	13.27	25.51
9	29.59	8.16	51.02	11.22
10	12.24	30.61	48.98	8.16
11	22.45	38.78	26.53	12.24
12	16.33	33.67	39.8	10.2
13	25.51	27.55	34.69	12.24
14	26.53	34.69	22.45	16.33
15	17.35	15.31	44.9	22.45

Table 5.7: PWM_9 for position -32

Position	%A	%C	%G	%T
1	16.53	41.32	31.4	10.74
2	6.61	9.92	4.96	78.51
3	90.08	0	0.83	9.09
4	0	3.31	0	96.69
5	92.56	0	1.65	5.79
6	74.38	0	0	25.62
7	90.91	2.48	4.13	2.48
8	62.81	0	9.92	27.27
9	26.45	10.74	54.55	8.26
10	11.57	39.67	39.67	9.09
11	18.18	24.79	42.98	14.05
12	11.57	41.32	40.5	6.61
13	19.83	38.84	27.27	14.05
14	19.01	32.23	30.58	18.18
15	21.49	22.31	41.32	14.88

Table 5.8: PWM_{10} for position -31

Position	%A	%C	%G	%T
1	13.27	43.88	34.69	8.16
2	3.06	11.22	3.06	82.65
3	92.86	0	0	7.14
4	0	0	0	100
5	95.92	0	1.02	3.06
6	69.39	0	0	30.61
7	91.84	1.02	5.1	2.04
8	59.18	4.08	18.37	18.37
9	19.39	18.37	52.04	10.2
10	17.35	33.67	34.69	14.29
11	24.49	33.67	35.71	6.12
12	16.33	31.63	37.76	14.29
13	24.49	27.55	34.69	13.27
14	15.31	32.65	34.69	17.35
15	19.39	26.53	37.76	16.33

Table 5.9: PWM_{11} for position -30

Position	%A	%C	%G	%T
1	10.64	42.55	42.55	4.26
2	6.38	4.26	4.26	85.11
3	87.23	0	0	12.77
4	0	2.13	0	97.87
5	97.87	0	0	2.13
6	63.83	0	0	36.17
7	82.98	2.13	10.64	4.26
8	55.32	0	29.79	14.89
9	25.53	12.77	51.06	10.64
10	10.64	34.04	38.3	17.02
11	27.66	40.43	25.53	6.38
12	21.28	25.53	40.43	12.77
13	6.38	23.4	44.68	25.53
14	12.77	27.66	51.06	8.51
15	17.02	23.4	53.19	6.38

Table 5.10: PWM_{12} for position -29

Position	%A	%C	%G	%T
1	11.36	38.64	40.91	9.09
2	2.27	20.45	4.55	72.73
3	86.36	0	0	13.64
4	2.27	4.55	4.55	88.64
5	97.73	0	0	2.27
6	81.82	0	0	18.18
7	93.18	0	6.82	0
8	56.82	0	22.73	20.45
9	34.09	9.09	52.27	4.55
10	18.18	18.18	61.36	2.27
11	27.27	40.91	29.55	2.27
12	13.64	36.36	40.91	9.09
13	20.45	34.09	29.55	15.91
14	11.36	38.64	31.82	18.18
15	18.18	27.27	43.18	11.36

Table 5.11: PWM_{13} for position -28

Position	%A	%C	%G	%T
1	35	40	20	5
2	0	10	0	90
3	90	0	0	10
4	0	0	0	100
5	85	0	5	10
6	65	0	0	35
7	90	0	10	0
8	60	0	15	25
9	40	0	60	0
10	0	35	20	45
11	15	25	50	10
12	35	30	30	5
13	15	50	20	15
14	20	35	40	5
15	20	25	30	25

Table 5.12: PWM_{14} for position -27

Position	%A	%C	%G	%T
1	18.18	36.36	18.18	27.27
2	9.09	0	9.09	81.82
3	81.82	0	0	18.18
4	0	0	0	100
5	100	0	0	0
6	90.91	0	0	9.09
7	90.91	0	9.09	0
8	54.55	0	18.18	27.27
9	36.36	0	54.55	9.09
10	9.09	45.45	45.45	0
11	27.27	45.45	18.18	9.09
12	18.18	54.55	27.27	0
13	9.09	45.45	18.18	27.27
14	18.18	18.18	45.45	18.18
15	9.09	27.27	36.36	27.27

Table 5.13: PWM_{15} for position -26

Position	%A	%C	%G	%T
1	25	50	0	25
2	0	0	0	100
3	100	0	0	0
4	0	0	0	100
5	100	0	0	0
6	87.5	0	0	12.5
7	87.5	0	12.5	0
8	25	0	75	0
9	25	12.5	25	37.5
10	0	62.5	12.5	25
11	12.5	62.5	25	0
12	37.5	12.5	37.5	12.5
13	0	75	0	25
14	25	12.5	62.5	0
15	0	87.5	12.5	0

Table 5.14: PWM_{16} for position -25

Position	%A	%C	%G	%T
1	50	0	50	0
2	0	50	0	50
3	75	0	0	25
4	0	0	0	100
5	100	0	0	0
6	100	0	0	0
7	100	0	0	0
8	25	0	50	25
9	0	25	50	25
10	0	25	75	0
11	25	25	50	0
12	0	75	0	25
13	25	25	50	0
14	0	0	100	0
15	0	50	50	0

Table 5.15: PWM_{17} for position -24

Position	%A	%C	%G	%T
1	33.33	33.33	33.33	0
2	33.33	0	0	66.67
3	100	0	0	0
4	33.33	0	0	66.67
5	100	0	0	0
6	33.33	0	0	66.67
7	100	0	0	0
8	66.67	0	0	33.33
9	33.33	0	66.67	0
10	0	66.67	33.33	0
11	33.33	33.33	33.33	0
12	0	0	66.67	33.33
13	0	33.33	33.33	33.33
14	0	0	0	100
15	0	0	66.67	33.33

Table 5.16: PWM_{18} for position -23

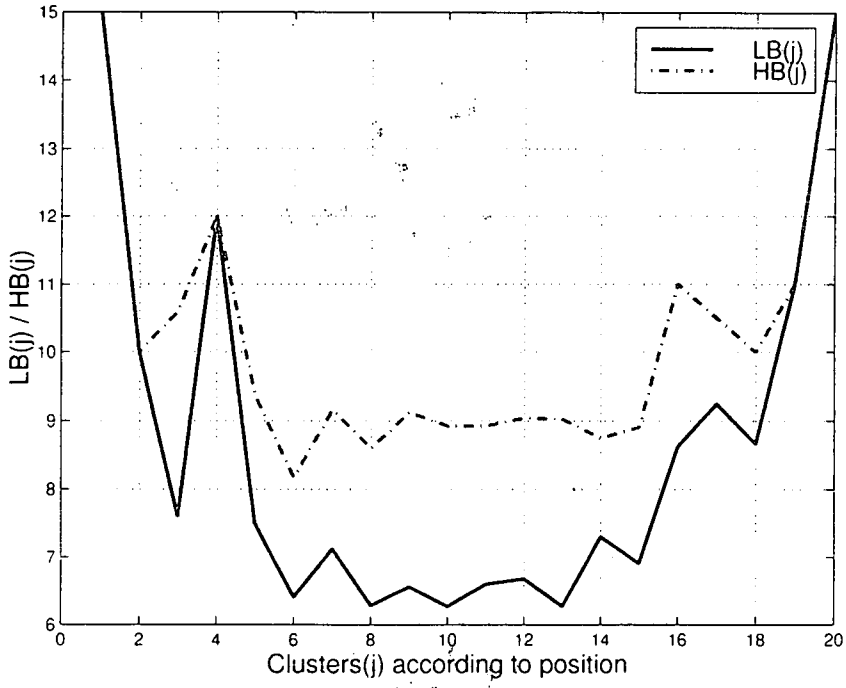

Figure 5-1: Bounds of promoter data

PWM19 for position -22					PWM20 for position -21				
Position	%A	%C	%G	%T	Position	%A	%C	%G	%T
1	50	0	50	0	1	100	0	0	0
2	0	0	100	0	2	0	0	100	0
3	100	0	0	0	3	100	0	0	0
4	0	0	0	100	4	0	0	0	100
5	100	0	0	0	5	100	0	0	0
6	50	0	0	50	6	100	0	0	0
7	100	0	0	0	7	100	0	0	0
8	50	50	0	0	8	0	100	0	0
9	0	50	50	0	9	100	0	0	0
10	0	50	50	0	10	0	100	0	0
11	50	50	0	0	11	0	100	0	0
12	50	0	50	0	12	0	100	0	0
13	0	0	100	0	13	0	0	0	100
14	0	0	50	50	14	0	100	0	0
15	0	100	0	0	15	0	100	0	0

Table 5.17: PWM19 and PWM20 for position -22, -21

Then, we use these bounds directly to predict the TATA containing sequences from both promoter data which have 878 sequences (and which is the same set we used to get new $PWMs$) and non-promoter data (which contains 8000 cds and 8000 intr sequences). The results are shown in Table 5.18. In the column "promoter" we have the number of TATA-box sequences recognized using $LB_{pm}(j)$ and $HB_{pm}(j)$. Similarly in the columns "cds" and "int" we have the number of TATA-box sequences falsely recognized in cds and intr data.

From Table 5.18, one can observe that there is no TATA-box containing sequence found in groups 21, 22, 23, 24. The obtained TP=70.84% for promoters, while FP = 3.49% for intr, and for cds FP = 1.69%.

$Group(Position)$	$LB_{pm}(j)$	$HB_{pm}(j)$	$promoter$	cds	int
$1(-40)$	15	15	1	0	0
$2(-39)$	10	10	2	0	0
$3(-38)$	7.6	10.6	5	0	1
$4(-37)$	12	12	2	0	0
$5(-36)$	7.5	9.375	8	1	2
$6(-35)$	6.4118	8.1765	22	16	27
$7(-34)$	7.12	9.16	28	6	15
$8(-33)$	6.2857	8.6	91	32	53
$9(-32)$	6.5612	9.1224	106	11	23
$10(-31)$	6.2727	8.9256	127	23	36
$11(-30)$	6.602	8.9286	90	9	27
$12(-29)$	6.6809	9.0426	41	6	13
$13(-28)$	6.2727	9.0227	56	25	73
$14(-27)$	7.3	8.75	17	2	2
$15(-26)$	6.9091	8.9091	8	4	6
$16(-25)$	8.625	11	8	0	0
$17(-24)$	9.25	10.5	4	0	0
$18(-23)$	8.677	10	3	0	1
$19(-22)$	11	11	2	0	0
$20(-21)$	15	15	1	0	0

Table 5.18: Number of predicted TATA-box contained sequences

$Group(Position)j$	$LB_{cds}(j) - HB_{cds}(j)$	$LB_{int}(j) - HB_{int(j)}$
1(−40)	0	NaN
2(−39)	0	NaN
3(−38)	0	8 − 8
4(−37)	0	NaN
5(−36)	6.625 − 7.5	5.75 − 8.375
6(−35)	3.2941 − 7.2353	2.8824 − 8.0588
7(−34)	4.36 − 8.76	4.04 − 8.24
8(−33)	2.3143 − 7.9857	1.8286 − 7.6571
9(−32)	2.3061 − 8.3469	2.1429 − 7.7449
10(−31)	2.0413 − 8.0826	1.8512 − 7.7025
11(−30)	2.9082 − 8.2551	2.8061 − 7.6429
12(−29)	1.9574 − 8.4468	1.9149 − 7.8723
13(−28)	2.1136 − 7.8091	2.3864 − 7.9773
14(−27)	3.8 − 7.65	4.25 − 7.65
15(−26)	5 − 7.7273	5 − 8.1818
16(−25)	0	NaN
17(−24)	0	NaN
18(−23)	10.3333 − 10.3333	9.3333 − 9.3333
19(−22)	0	NaN
20(−21)	0	NaN
21(−20)	0	NaN
22(−19)	0	NaN
23(−18)	0	NaN
24(−17)	0	NaN

Table 5.19: low-high bound of cds and intr

5.3 More strict min-max $score1$ results in better prediction

We also want to reduce FP to *zero*. So, we calculate $score1_{cds}(k, j)$ and $score2_{cds}(k, j)$ with equation (4.6) for cds data set, so as to obtain $LB_{cds}(j)$ and $HB_{cds}(j)$ with equation (4.8). Analogously, we calculate $score1_{int}(k, j)$, $score2_{int}(k, j)$ with equation (4.7) for intr data set, and from these we obtain $LB_{int}(j)$, $HB_{int}(j)$ with equation (4.9). The results of recognition are shown in Table 5.19. If NaN appears in some position, this means that no sequence was found at the respective position.

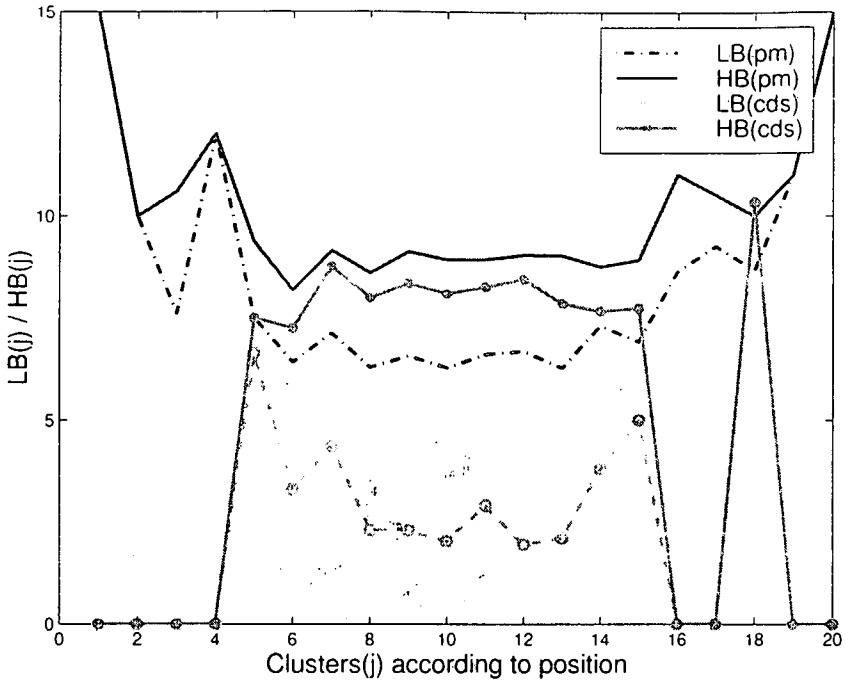

Figure 5-2: Comparison of bounds of TATA-box containing sequences, and cds sequences

We make a comparison between the promoter and cds data and the promoter and intr data, as shown in Figure $5-2$ and Figure $5-3$, respectively.

From Figure $5-2$ we observe that if we want to get *zero* FP of cds data set, we have to only change $LB(j)$ in equation (4.5) from group 5 to group 15, as well as for group 18. $LB(j)$ should be the $\max[LB_{pm}(j), HB_{cds}(j)]$. From Figure $5-3$ we conclude that we also need to change low bound $LB(j)$ from group 5 to group 15, as well as for groups 3 and 18. $LB(j)$ for these groups should be $\max[HB_{int}(j), LB_{pm}(j)]$. We tune $LB(j)$ one by one according to the order which makes FP drop as fast as possible, while simultaneously keeps TP as high as possible. So, according to (5.18), firstly, we tune $LB(13)$, then for the groups $8, 10, 6, 11, 7, 12, 9, 15, 5, 14, 18, 3$. Eventually, we get the result as shown both in Figure $5-4$ and Table 5.20.

When we compare this result with the one obtained in the previous chapter from

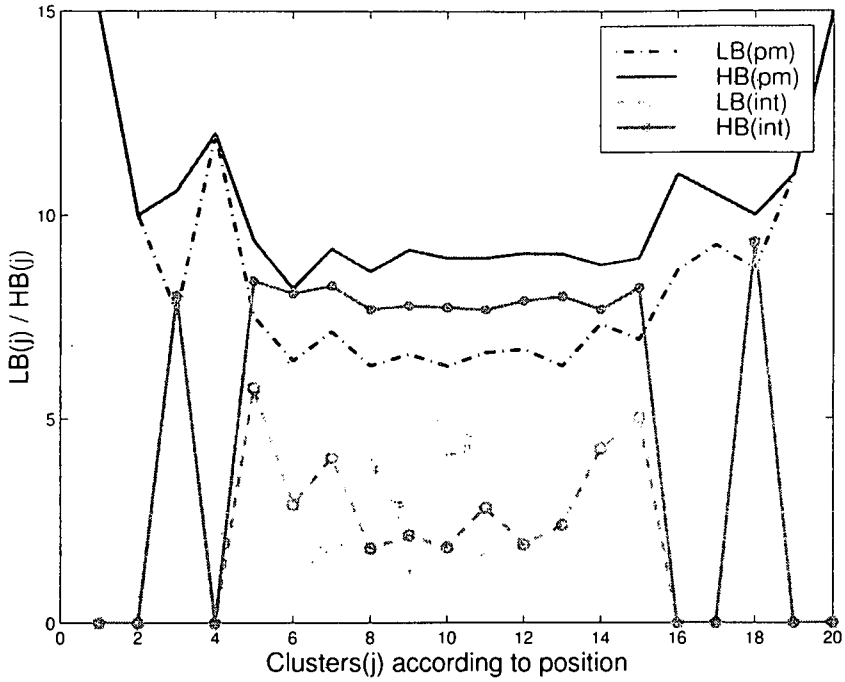

Figure 5-3: Comparison of bounds of TATA-box containing sequences, and int sequences

Promoter(%)	cds(%)	int(%)
70.84	1.69	3.49
66.29	1.39	2.57
60.82	1.01	1.96
50.11	0.74	1.51
48.18	0.54	1.19
40.09	0.43	0.85
37.59	0.37	0.67
31.44	0.26	0.49
20.50	0.1	0.18
19.82	0.0375	0.1
19.48	0.025	0.075
18.79	0	0.025
18.45	0	0.0125
18.34	0	0

Table 5.20: Prediction result with new PWMs

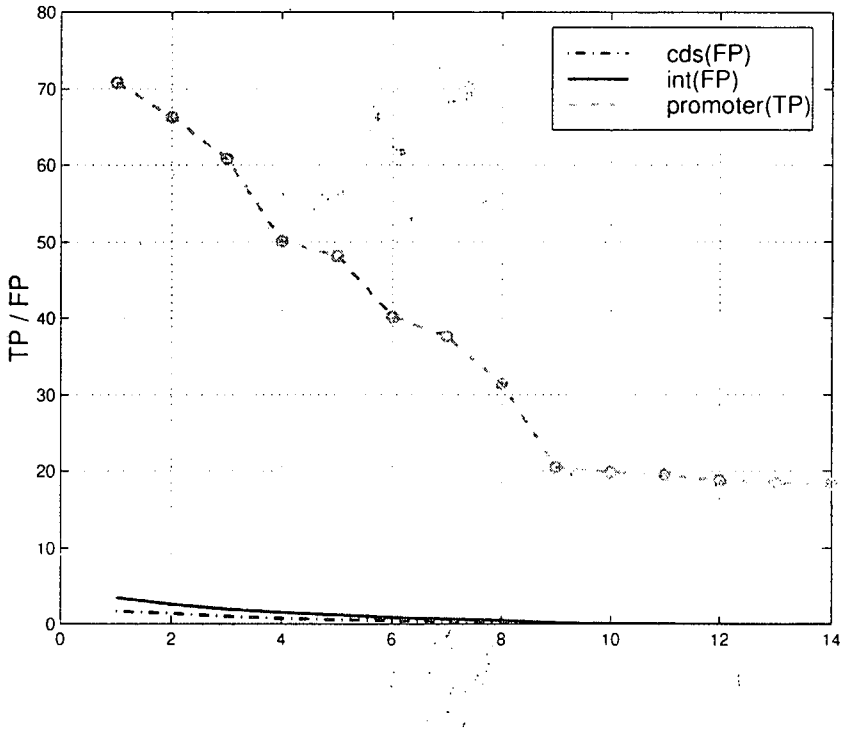

Figure 5-4: Prediction with new PWMs

pAtr data, as shown in Figure 4 – 5 and Table 4.21, we conclude that the results in this chapter are better. One can argue that here, the same data set is used for training and testing. This is true. However, one must also have in mind that description by the $PWMs$ is a statistical one, which does not precisely describe any one sequence. Thus, the larger the training set is, the better the final results could be expected. At the end of the day, different separation of the promoter set to training and testing data could result in different $PWMs$ and thus in different results. We can also observe that the number of positions where TATA motif was found did increase by using the whole promoter set, as compared to the restricted promoter set *pAtr* of Chapter 4. This is one more argument toward using all available data for generating PWM based model.

Chapter 6

Graphical User Interface

6.1 GUI design

Here we present the Graphical User Interface (GUI) made for use with the method developed. Our goal in using this GUI is to predict if an input sequence is a promoter sequence by locating a hypothetical TATA motif in it. The location of the found TATA motif (if it is found) should be presented. During the process of designing GUI, we use the results which are based on Chapter 5.

To design GUI, the basic idea is to make it simple. Our concept for GUI design is in line of the following:

- First of all, we will provide a **Load** button, so that when it is clicked on we can download the data set with sequences;
- Secondly, when we get the data, we need to know how many sequences there are in the data set, and at what length. Thus we add two editable text boxes. One displays the number of sequences, whilst another displays the length of each sequence. We also add two static texts `'Length of sequence'` and `'Number of sequence'` to indicate the usage of the two editable text boxes, respectively;
- Thirdly, we add **Start** button, **Next** button and **End** button;

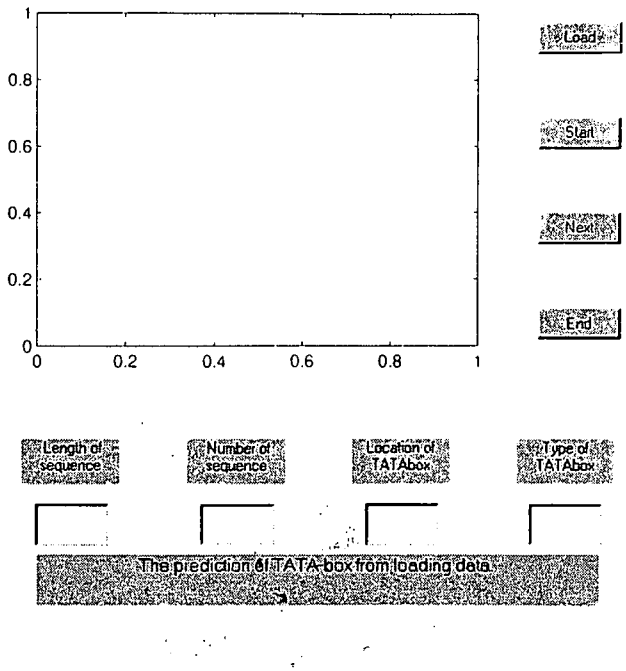

Figure 6-1: The appearance of GUI

- Then, we need another two editable text boxes to display the location of TATA motifs (in the sequence) and the type of TATA box motif (i.e. which cluster it belongs to). We also need two static texts `'Location of TATAbox'` and `'Type of TATAbox'` to indicate these two editable text boxes. We put one static text on the bottom to explain the goal of this GUI.

- Finally, we want to view the scored values of prediction for the motif, i.e. how these scores distribute along the sequence. So we add **axes** for plotting the figure to show this.

Up to now, our GUI has been designed as shown in Figure $6-1$:

The **Start** button and **Next** button are grey because we disabled them (set its Enable property to 'off') to avoid an error result which is caused by clicking the **Start** or **Next** before the **Load** button is clicked. Only after loading the data, the **Start** button gets re-enabled; and only after **Start**, the **Next** button gets re-enabled. Figure $6-2$ and Figure $6-3$ are after click **Start** and **Next** buttons, respectively.

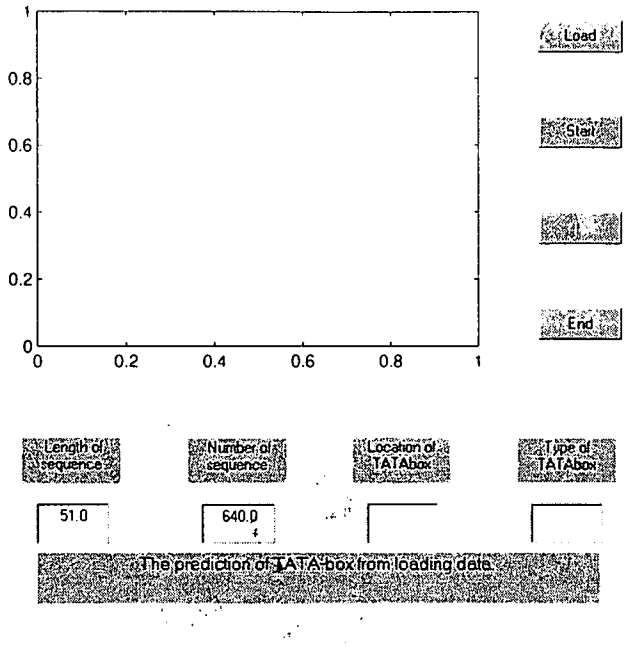

Figure 6-2: GUI after click LOAD

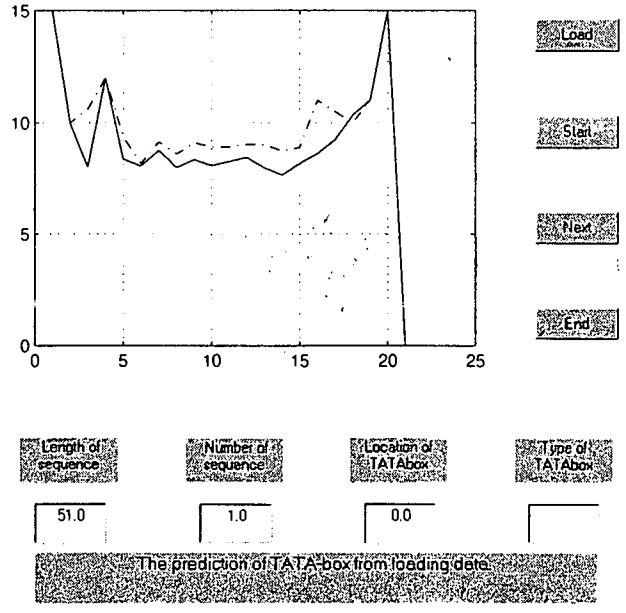

Figure 6-3: GUI after click START

6.2 Callbacks

Now, we need to manage what happens when a button is depressed, or when we type in editable text boxes. For this we need to make callbacks for each graphical handle.

- When one clicks on the **Load** button, data is loaded from disk, and the number of sequences in the whole set and the length of the sequences are displayed (normally, sequences in the same data set are with the same length) (see Figure 6 – 2);

- When one clicks on the **Start** button, the analysis of the first sequence begins. The editable text box which is below the static text `'Number of sequence'` displays the position of the sequence under analysis. That is, if the first sequence of the set is under analysis, display 1, and so on. If there is no TATA motif found, display 0 in the editable text box which is below the static text `'Location of TATAbox'`; if the TATA-box is found, display the location of TATA motif in the editable text box which is below the static text `'Location of TATAbox'` and display the type of the TATA motif (which cluster of 20 possible) in the editable text box which is below the static text `'Type of TATAbox'` (see Figure 6 – 2). Because we developed our method from 38 nucleotide-long subsegments of sequences, the 20 types of the TATA-boxes are defined by their location on these subsegments. So when we check the long sequence, we firstly divide the long sequence into short regions, each containing 38 nucleotides. So, if the editable text below static text 'location of TATAbox' displays 1, TATA motif exists in the first subregion of 38 nucleotides, and so on.

- When one clicks the **Next** button, the next sequence is analyzed and in the editable text box below the static text `'Number of sequence'` its position is displayed.

- Further, if we want to analyze a specific sequence among the whole data set, then, for example, when we type 12 in the editable text box which is below static text

'Number of sequence', the program will check sequence $No.12$ and the result will come out (see Figure $6-4$).

- When we click the **End** button, the window closes.

To speed up our GUI, we use Function Callbacks. That is, instead of giving long strings of commands to callbacks, we only give short function names to each callback. The long string of commands are in the function file. For example, we list the function LOAD which is the callback of the **Load** button.

```
functionLOAD
loadseq_nik;
[mm, m] = size(pAtr);
k = ones(mm, 1);
set(gcbf, /UserData/, pAtr)
sequencelength
LocalSetDisplay(m)
StHndl = findobj(gcbf, /String/, /Start/);
set(StHndl, /Enable/, /on/)
```

Then we send all the callbacks to one single function named `TATACOMPONENT.m`. We use switchyard programming to prevent function proliferation.

```
function tatacomponent(action)
switch (action)
case 'Load'
.load seq_nik;
[mm,m]=size(pAtr);
k=ones(mm,1);
set(gcbf,'UserData',pAtr)
sequencelength
LocalSetDisplay(m)
```

```
StHndl=findobj(gcbf,'String','Start');
set(StHndl,'Enable','on')
case'Start'
.
.
.
case'Next'
.
.
.
```

The whole function `tatacomponent.m` is available in Appendix.

Figure 6 – 4 shows the case when we analyze sequence *No.*12. It contains a TATA motif which is found in the first region of 38 nucleotides, and the cluster to which the TATA-box belongs is 13, which means that the TATA motif starts from the nucleotide *No.*13 in this first region.

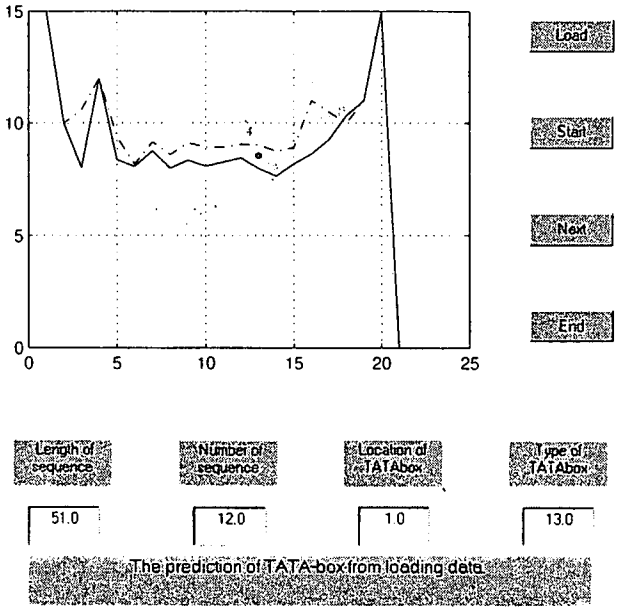

Figure 6-4: Analysis of one particularly selected sequence

Chapter 7

Conclusions

In this study we applied a specific statistical signal processing method, the so-called position weight matrix approach, to recognize patterns/motifs contained in discrete signals that have finite set of values. Our approach was based on the assumption that specific patterns in specific classes of signals have some positional distribution relative to a reference point, and that patterns on different positions have different characteristics. We also assumed that such different pattern characteristics can be conveniently described by the $PWMs$. We did apply this approach to recognition of a specific important motif in DNA sequences, the TATA-box. The approach was based on positional clustering of the TATA-box and determination of the PWMs for different clusters. The testing of the final results has been made on the problem of Eukaryotic Polymerase II promoter recognition.

By applying positional clustering and describing each cluster by the PWM, a more precise description of the characteristics of TATA-box motifs is obtained. The price for this was the increased number of $PWMs$, instead of only one PWM. However, this has resulted in considerably increased recognition accuracy.

Our method is tested both on the TATA-box containing promoter data set and non-promoter data set. For the promoter test set which contains 238 sequences is $TP = 10.08\%$ when for the whole non-promoter test set (8000 cds sequences and 8000 intr sequences) $FP = 0\%$. On the training set which contains 640 promoter sequences, the

new $PWMs$ produce $TP = 23.44\%$ when $FP = 0\%$ for both cds and intr sequences. This is quite good result.

7.1 Advantages of the new approach

The development of new $PWMs$ by means of positional clustering of motifs allows for a general approach of determining the more precise statistical models of motifs, than if only cumulative PWM is used for motif description. This was definitely shown in the case of TATA-box motifs, but there is no restriction to use it more generally. It is applicable for all discrete signal with finite set of values. The method can be easily implemented in the programs that do search of promoters in uncharacterized DNA.

7.2 Limitations

Although we have demonstrated the advantages of this method on a specific difficult example, there are several limitations which need to be solved in the further research. Firstly, during the process of clustering, we use some algorithm to extract TATA-like components contained in sequences. In fact, we are not sure if all of them are functional TATA-like motifs, and the approach is more computational. This problem can be resolved if proper TATA motif context is analyzed and if the training and test sets are composed of only experimentally verified TATA motifs.

Secondly, the results on the test set are always worse than on the training set. This is a general problem of pattern recognition method and it depends directly on the features that are used for recognition of patterns.

Thirdly, some of the clusters may have a very small number of sequences. This then may cause problems in accurate determination of the $PWMs$ for such clusters.

Chapter 8

Appendix: Matlab programs

All programs are written in the Matlab language. Matlab is a registered Trademark of MathWorks Inc., USA.

8.1 Script files

8.1.1 program1.m:Prediction based on original PWM with one assumption

All data is in the file *seq_nik.mat*. There are 878 promoter sequences which are split into two subsets *pAtr* and *pAtst*. The 8000 cds sequences were contained in five distinct subsets *cds*0, *cds*1, *cds*2, *cds*3 and *cds*4, and, similarly, 8000 intr sequences were contained in five subsets *int*0, *int*1, *int*2, *int*3 and *int*4. Each of these subsets had 1600 sequences of length 51 nucleotides. Using LEV1 and LEV2 parameters as threshold levels, this script predicts sequences that contain a hypothetical TATA box.

```
% script program1
load seq_nik;     % Load data;
SEQ = [pAtr pAtst];     % Promoter dataset;
%SEQ = [int0 int1 int2 int3 int4];   % intr dataset;
```

```
%SEQ = [cds0 cds1 cds2 cds3 cds4];      % cds dataset;
[n,m] = size(SEQ);     % The size of the data under analysis;
suma = 0;     % Initiation of the number of TATA box containing sequences;
 for k = 1:m
 p1 = SEQ(:,k);
 [score] = tata1(p1);     % Invoke function tata1.m;
 kk = sum(score);
 if kk>0
 suma = suma+1;
 end
 end
 disp([suma/m]);
```

8.1.2 Program2.m: Prediction based on original PWM with both assumption

The only difference between *program2.m* and *program1.m* is that, instead of invoking *tata1.m*, we invoke another program, *tata2.m*.

```
% script program2
loadseq_nik;                          % Load data;
SEQ = [pAtrpAtst];                    % Promoter dataset;
%SEQ = [int0 int1 int2 int3 int4];     % intr dataset;
%SEQ = [cds0 cds1 cds2 cds3 cds4];    % cds dataset;
[n,m] = size(SEQ);                    % The size of the data under analysing;
suma = 0;          % Initiation of the number of TATA box containing sequences;
 for k = 1:m
 p1 = SEQ(:,k);
 [score] = tata2(p1);                 % Invoke function tata2.m;
 kk = sum(score);
```

```
if kk>0
suma = suma+1;
end
end
disp([suma/m]);
```

8.1.3 Program3.m: Calculate new PWMs

% Script program3.m

% This script calculates new PWMs

% MM – A matrix that contain all new PWMs. It has 96 rows and 15 columns, and contains 24 PWMs each of which occupies 4 rows.

```
load seq_nik;
SEQ = [pAtr pAtst];
[mm,m] = size(SEQ);
N = zeros(96,15);
t = zeros(1,24);
MM = zeros(96,15);
for k1 = 1:m
 p1 = SEQ(:,k1);
 [score,Ms1b,Ms2b] = tata2(p1);     % Invoke function file tata2.m
xc = p1(Ms1b:Ms1b+14);
 kk = sum(score);
 if (kk>0) & (Ms1b == Ms2b)
 for i = 1:24
 if Ms1b == i
 NN = N(4*(i-1)+1:4*i,:);
 NN = new_N(xc,NN);                 % Invoke function file new_N.m;
 N(4*(i-1)+1:4*i,:)  = NN;
```

```
    t(i) = t(i)+1;
    break
    end
    end
    end
  end
  for i = 1:24
   MM(4*(i-1)+1:4*i,:)  = N(4*(i-1)+1:4*i,:)/t(i);
  end
```

8.1.4 Program4.m: prediction with new PWMs calculated from pAtr

```
for j = 1:12                    % In tuning LB to get 0% FP, 12 of them have to be
```

changed. They are tuned one by one.

```
    load minmax_pA_cds;      % load LB_cds and HB_cds;
    cds = Da;
    load minmax_pA_int;       % load LB_int and HB_int;
    int = Da;
    load new_minmax_pAtr;     % load LB_pm and HB_pm;
    pp = [cds;int;Xiao];
    n = max(pp);
    load seq_nik;
    SEQ = [pAtst];


    x3 = 2;
    Num = zeros(1,24);
    score = zeros(1,24);
    [mm,m] = size(SEQ);
```

```
load new_minmax_pAtr;
LB = Xiao;
  HB = Da;
ii = [8 9 10 11 6 12 13 14 7 15 16 5];    % the order of tuning LB(j);
e = zeros(12);
LB(ii(1:j)) = n(ii(1:j))+0.0001;
for k1 = 1:m
p1 = SEQ(:,k1);
for i = 1:24
[score1(i),score2(i)] = new_newscore(p1(i:i+14),i,x3);
% invoke function   new_newscore(x1,x2,x3);
if score2(i) == 0
Num(i) = Num(i);
elseif (score2(i)~=0) & (score1(i)>=LB(i)) & (score1(i)<=HB(i))
Num(i)=Num(i)+1;
break
end
end
end
N=sum(Num);
e(j)=([N/m]);
disp([e(j)]);
clear
end
```

8.1.5 Program5.m: predicion with new PWMs calculated from whole promoter dataset

for j=1:13 % In tuning LB to get 0% FP, 13 of them have to be changed, and they are tuned one by one.

```
load minmax_cds;
cds=Da;
load minmax_int;
int=Da;
load new_minmax;
pp=[cds;int;Xiao];
n=max(pp);
load seq_nik;
SEQ=[pAtst pAtr];

x3 =1;
Num=zeros(1,24);
score=zeros(1,24);
[mm,m]=size(SEQ);
load new_minmax;
LB = Xiao;
HB = Da;
ii = [13 8 10 6 11 7 12 9 15 5 14 18 3];   % the order of tuning LB;
e = zeros(13);
LB(ii(1:j)) = n(ii(1:j))+0.0001;
for k1 = 1:m
p1 = SEQ(:,k1);
for i = 1:24
[score1(i),score2(i)] = new_newscore(p1(i:i+14),i);  % invoke function;
```

```
    if score2(i) == 0
    Num(i) = Num(i);
    elseif (score2(i)~=0)&(score1(i)>=LB(i))&(score1(i)<=HB(i))
    Num(i) = Num(i)+1;
    break
    end
    end
    end
    N = sum(Num);
    e(j) = ([N/m]);
    disp([e(j)]);
    clear
   end
```

8.2 Function files

8.2.1 function tatacomponent(action)

```
function tatacomponent(action)
```

% This is the callback function for myGUI of TATAbox

% All the sequences should be put in a data file called seq_nik

% Promoter sequences for training are in the data file should be pAtr;

% This is the general case. Sequences could be at any length.

```
switch (action)
case 'Load'
```

% When loading data from disk, 'Number of sequences' shows how many sequences there are

% 'Length of sequence'shows the length of the sequences (normally, all the sequences

% in the same data file with the same length)

```
load seq_nik;
 [mm,m]=size(pAtr);
 mm=mm-14;
 k=ones(mm,1);
 set(gcbf,'UserData',pAtr)
 sequencelength
 LocalSetDisplay(m)
 StHndl=findobj(gcbf,'String','Start');
 set(StHndl,'Enable','on')
case 'Start'
% when press the 'Start' button, first sequence is handled.
% 'Number of sequence' shows the location of the sequence among all sequences
 SEQ=get(gcbf,'UserData');
 [mm,m]=size(SEQ);
 k1=1;
 p1=SEQ(:,k1);
 [i,N]=prediction(p1);
 TATAboxlocation(i)
 TATAboxType(N)
 LocalSetDisplay(k1)
 NxHndl=findobj(gcbf,'String','Next');
 set(NxHndl,'Enable','on')
case 'Next'
% Each time when press the 'Next' button, 'Number of sequence' increases by 1
 SEQ=get(gcbf,'UserData');
 NmHndl=findobj(gcbf,'Tag','sequencenumber');
 String=get(NmHndl,'String');
 k1=eval(String);
```

```
    p1=SEQ(:,k1+1);
    [i,N]=prediction(p1);
    TATAboxlocation(i)
    TATAboxType(N)
    LocalSetDisplay(k1+1)
  case 'NumberofSequence'
  % When one sequence is selected, prediction of TATA-box from this particular
  % sequence is made automatically
  SEQ=get(gcbf,'UserData');
    NmHndl=findobj(gcbf,'Tag','sequencenumber');
    String=get(NmHndl,'String');
    k1=eval(String);
    p1=SEQ(:,k1);
    [i,N]=prediction(p1);
    TATAboxlocation(i)
    TATAboxType(N)
    LocalSetDisplay(k1)
  case 'End'
  close(gcbf)
  end
  function sequencelength          % Subfunction for displaying the length of
the sequences.
  SEQ=get(gcbf,'UserData');
  [mm,m]=size(SEQ);
  LgHndl=findobj(gcbf,'Tag','sequencelengthtext');
  set(LgHndl,'String',sprintf('%3.1f',mm))
  function TATAboxlocation(i)    % Subfunction for displaying the location of the
TATA-box
```

```
LcHndl=findobj(gcbf,'Tag','TATAboxlocationtext');
set(LcHndl,'String',sprintf('%3.1f',i));
function TATAboxType(N)          % Subfunction for displaying the type of the TATA-
box. There are 23 types.
TpHndl=findobj(gcbf,'Tag','TATAboxTypetext');
set(TpHndl,'String',sprintf('%3.1f',N));

function LocalSetDisplay(m)    % Subfunction for displaying the location of the
sequence among all sequences
LiHndl=findobj(gcbf,'Tag','sequencenumber');
set(LiHndl,'String',sprintf('%3.1f',m))
function [ii,N]=prediction(p1)  % Subfunction for predicting the TATA-box for
each sequence
SEQ=get(gcbf,'UserData');
[mm,m]=size(SEQ);
load minmax_for_gui;
tt=1:24;
plot(tt,Xiao,tt,Da,'r-.');
hold on
grid on
flg=0;
for j=1:(mm-38)
if flg==1
break
end
for i=1:24
[score1(i),score2(i)]=new_newscore(p1(i:i+14),i);
if (score2(i)~=0)&(score1(i)>=Xiao(i))&(score1(i)<=Da(i))
```

```
N=i;flg=1;ii=j;
plot(i,score1(i),'g.');
hold off
break
else
ii=0;
end
end
end
```

8.2.2 function [score] = tata1(x)

```
function [score] = tata1(x)
%function [score] = tata1(x)
% x - input sequences as a matrix - each sequence is a column
% score - overall assessment of the presence of the TATA box at a particular position;
assessment is based on the Position Weight Matrix (PWM) published in [15]
tt = 1:24;
for n = tt,
 [score1(n),score2(n),flg] = tata_score(x(n:n+14));    % Invoke function tata_score
end
% normalization
k1 = 0.9; k2 = 0.9;
% k1 = 0.95; k2 = 0.95;
% k1 = 0.98; k2 = 0.98;
% k1 = 1; k1 = 1;
[LEV1,Ms1] = max(score1);
[LEV2,Ms2] = max(score2);
if LEV2 == 0
```

```
 score = 0;
else
 score = (score1>=k1*LEV1)&(score2>=k2*LEV2);
end
```

8.2.3 Function [score1, score2, flg] = tata_score(x)

```
function [score1, score2, flg] = tata_score(x)
```

% x - input sequence as vector column has 15 elements

% score1, score2 - overall assessment of the presence of the TATA box at a particular position

% score1 - obtained as sum of weights

% score2 - obtained as product of weights

```
a = 0.1260; g = 0.0806; t = 0.1335; c = 0.1340;
```

PWM = [15.7 4.1 90.5 0.8 91 68.9 92.5 57.1 39.8 14.4
21.3 21.1 21.1 21.1 17.5 19.8;
37.3 11.8 0 2.6 0 0 0.8 0.5 11.3
34.7 37.8 32.6 30.3 27.5 26 ;
39.1 4.6 0.5 0.5 1.3 0 5.1 11.3 40.4
38.6 32.9 32.9 32.9 35.7 36 ;
8 79.4 9 96.1 7.7 31.1 1.5 31.1 8.5
12.3 8 13.4 15.7 19.3 18.3];

% PWM is defined

```
flg = 0;
```

% score calculation

```
score1 = 0;
score2 = 1;
for i = 1:15
if x(i) == a,
```

```
p_current = PWM(1,i);
elseif x(i) == c,
p_current = PWM(2,i);
elseif x(i) == g,
p_current = PWM(3,i);
elseif x(i) == t,
p_current = PWM(4,i);
else
p_current = NaN;
end
score1 = score1 + p_current;
score2 = score2 * p_current/20;
end
```

8.2.4 function [score, Ms1, Ms2] = tata2(x)

```
function [score] = tata2(x)
% function [score] = tata2(x)
% x - input sequences as a matrix - each sequence is a column
% score - overall assessment of the presence of
% the TATA box at a particular position
% flg = 0 - all is OK, flg = 1 - there was error in input data
tt = 1:24;
for n = tt,
 [score1(n),score2(n),flg] = tata_score(x(n:n+14));
end
% normalization
k1 = 0.98; k2 = 0.98;
LEVscore = 650;
```

```
score1log = score1>LEVscore;
[LEV1,Ms1] = max(score1);
[LEV2,Ms2] = max(score2);
if LEV2 == 0
 score = 0;
else
score = (score1>=k1*LEV1)&(score2>=k2*LEV2)&score1log;
end
```

8.2.5 function [N] = new_N(x1, x2)

```
function [N] = new_N(x1,x2)
   % N - postion matrix of each group of sequences
   % x - input sequence as vector column with 15 elements
   a = 0.1260; g = 0.0806; t = 0.1335; c = 0.1340;
   for i = 1:15
    if x1(i) == a,
    N(1,i) = x2(1,i)+1;
    elseif x1(i) == c,
    N(2,i) = x2(2,i)+1;
    elseif x1(i) == g,
    N(3,i) = x2(3,i)+1;
    elseif x1(i) == t,
    N(4,i) = x2(4,i)+1;
    else
    disp('error');
    end
   end
```

8.2.6 function [score1, score2] = new_newscore(x1,x2,x3)

```
function [score1,score2] = new_newscore(x1,x2,x3)
   a = 0.1260; g = 0.0806; t = 0.1335; c = 0.1340;

   % define PWM
   if x3 == 1;
    load new_PWMS;
   elseif x3 ==2;
    load new_PWMS_pAtr;
   end
   % initiation
    score1 = 0;
    score2 = 1;
   % score calculation
    for i = 1:15
    if x1(i)==a,
    p_current = MM(4*(x2-1)+1,i);
    elseif x1(i)==c,
    p_current = MM(4*(x2-1)+2,i);
    elseif x1(i)==g,
    p_current = MM(4*(x2-1)+3,i);
    elseif x1(i)==t,
    p_current = MM(4*x2,i);
    else
    p_current = NaN;
    end
    score1 = score1 + p_current;
    score2 = score2*p_current;
```

Bibliography

[1] Alexandrov, N. N. and A. A. Mironov (1990) Application of a new method of pattern recognition in DNA sequence analysis: a study of E.coli promoters, *Nucl. Acids Res.*, **18:** 1847-1852.

[2] Anderberg, M. R. (1973) Cluster Analysis for Applications. *Academic Press.*

[3] Anderson, J., A. Pellionisz, and E. Rsenfeld (1990) *Neurocomputing 2: Directions for Research.* Cambridge Mass.: MIT Press.

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

[5] Bajić, V. B. and I. V. Bajić (1999) Artificial Neural Networks in DNA Regulatory Region Recognition: The Case of Promoters. *Part2 of Analysis of DNA Sequences with Artificial Neural Networks - in Biosensors and Biocomputing tutorial track of IJCNN'99.*

[6] Bajić, V. B. (2000) Comparing the success of different prediction software in sequence analysis: A review, *Briefings in Bioinformatics*, Vol. 1, No. 3.

[7] Bajić, V. B. and I V Bajić (2000) Neural network system for promoter recognition (20 pages), Chapter in *Future Directions for Intelligent Systems and Information Science* (Nik Kasabov, Ed.), Physica-Verlag, New York.

[8] Beebee, T. and J. Burke (1988) *Gene structure and transcription,* IRL Press, Oxford, UK.

[9] Bernier, F., P. Soucy, and V. Luu-The (1996) Human phenol sulfotransferase gene contains two alternative promoters: Structure and expression of the gene, *DNA Cell Biol.,* **5:**367-375.

[10] *http://www.bic.nus.edu.sg/.*

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

[12] 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.

[13] Bucher, P., J. W. Fickett, and A. Hatzigeorgiou (1997) Computational analysis of transcriptional regulatory elements: a field in flux, *Bioinformatics online, Vol.12, Issue. 5, pp361-362, Oct. 1997. http://www3.oup.co.uk/bioinformatics/hdb/Volume_12/Issue_05/html/120361_gml.html.*

[14] Burge, C. (1997) Identification of genes in human genomic DNA (Ph. D thesis, Stanford University, Stanford, CA).

[15] Burge, C. and S. Karlin (1998) Finding the genes in genomic DNA, *Current Opinion on Structural Biology, 8:346-354.*

[16] Burge, C. and S. Karlin (1997) Prediction of complete gene structures in human genomic DNA, *J. Mol. Biol.,* **268:**79-94.

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

[18] Chen, Q. K., G. Z. Hertz, and G. Stromo (1995) MATRIX SEARCH 1.0: A computer program that scans DNA sequences for transcriptional elements using a database of weight matrices, *Comp. Appl. Biosci.*, **11**:563-566.

[19] Chen, Q. K., G. Z. Hertz, and G. D. Stormo (1997) PromFD 1.0: A computer program that predicts eukaryotic pol II promoters using strings and IMD matrices, *Comp. Appl. Biosci.*, **13**:29-35.

[20] Claverie, J. M. and S. Audic (1996) The statistical significance of nucleotide position-weight matrix matches, *Comp. Appl. Biosci.*, **12**:431-440.

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

[22] Claverie, J., O. Poirot and F. Lopez (1997) The difficulty of identifying genes in anonymous vertebrate sequences, *Computer & Chemistry*, **21(4)**:203-214.

[23] Clifford, H. T. (1975) An introduction to Numerical Classification. *Academic Press.*

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

[25] Crowley, E. M., K. Roeder, and M. Bina (1997) A statistical model for locating regulatory regions in genomic DNA, *J. Mol. Biol.*, **268**:8-14.

[26] Davies, E. R. (1993) Electronics noise and signal recovery, *Academic Press.*

[27] Demeler, B. and G. W. Zhou (1991) Neural network optimization for E.coli promoter prediction, *Nucl. Acids Res.*, **19:** 1593-1599.

[28] *http://www.accessexcellence.org/AB/GG/structure.html*

[29] Dong, S. and D. B. Searls (1994) Gene structure prediction by linguistic methods, *Genomics,* **162:** 705-708.

[30] *http://www.epd.isb-sib.ch/.*

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

[32] Fickett, J. W. and R. Guigo (1996) Computational gene identification, in *Internet for Molecular Biologist* (R. R. Miller, S. R. Swindell and G. Myers, Eds.), Horizon Scientific Press.

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

[34] Fields, C. and C. Soderlund (1990) GM: a practical tool for automating DNA sequence analysis, *Computer Applic. Biosci.,* **6:** 263-270.

[35] Frech, K., K. Quandt, and T. Werner. (1997a) Software for the analysis of DNA sequence elements of transcription, *Comp. Appl. Biosci.,* **13:**89-97.

[36] Gelfand, M. S. (1995) Prediction of function in DNA sequence analysis, *J. Comp. Biol.,* **2:**87-115.

[37] Gelfand, M. S., A. A. Mironov, and P. A. Pevzner (1996) Gene recognition via spliced sequence alignment, *Proc. Nat. Acad. Sci.,* **93:** 9061-9066.

[38] Gordon, A. D. (1981) Classification. *Monograph on applied probability and statistics* , published by Chapman & Hall, London.

[39] Gupta, M. S. (1977) Electrical noise: fundamentals & sources , *IEEE Press.*

[40] *http://www.ncbi.nlm.nih.gov/disease/*

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

[42] Jain, A. K., R. P. W. Duin and J. Mao (2000) Statistical Pattern Recognition: A Review. *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 1, January.

[43] Krogh, A. (1997) Two methods for improving performance of a HMM and their application for gene finding, in ISMB97, 179-186.

[44] Krogh, A. (1998) Gene finding: putting the parts together, in *Guide to Human Genome Computing* (Martin Bishop, Ed.), 2nd edition, Academic Press.

[45] Lewin, B. (1997). *Gene.*VI edn. Oxford University Press, Oxford.

[46] http://linkage.rockefeller.edu/wli/gene/.

[47] Marven, C. and G. Ewers (1994) A Simple Approach to Digital Sinal Processing. *Texas Instruments.*

[48] O'Shea-Greenfield, A. and S. T. Smale (1992) Rolès of TATA and Initiator elements in determining the start site location and direction of RNA Polymerase II transcription, *J. Biol. Chem.*, **267(2):** 1391-1402.

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

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

[51] Prestridge, D. S. (1995) Predicting Pol II promoter sequences using transcription factor binding sites, *J. Mol. Biol.*, **249:**923-32.

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

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

[54] Smale, S. T. (1997) Generality of a functional initiator consensus sequence, *Gene*, **182:** 13-22.

[55] Smale, S. T. (1997) Transcription initiation from TATA-less promoters within eukaryotic protein coding genes, *Biochim. Biophys. Acta.*, **1351:**73-88.

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

[57] Vaseghi, S.V. (1996) Advanced Signal Processing and Digital Noise Reduction. *Wiley and Teubner* .

[58] Wang, X. and V. B. Bajic (1999) Clustering DNA Data for Better PWM Determination. *International Conference on Artificial Inteligence: Development and practice of artificial intelligence techniques (ed. Bajic, V. B. and D. Sha), pp. 109-114,* IAAM-SAD, Durban, SA.

[59] Wolfe, S. L. (1995) *An Introduction to Cell and Molecular Biology*, Wadsworth Publishing Company, Belmont, California, USA.