PREDICTION OF DNA BINDING PROTEINS & CLASSIFICATION FROM SEQUENCE DERIVED FEATURES USING ANN AND HIDDEN MARKOV MODEL

By

GOUTHAM KONERU- 031514
SRIMATH KANDALAM SRIKANTH-031510

MAY-2007

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF BIOTECHNOLOGY & BIOINFORMATICS
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT, SOLAN, H.P, INDIA

CERTIFICATE

This is to certify that the work entitled, **"Prediction of DNA Binding Proteins and Classification from Sequence Derived Features using ANN nad Hidden Markov Model"** submitted by **Goutham Koneru(031514)** & **Srimath Kandalam Srikanth(031510)** in partial fulfillment for the award of degree of Bachelor of Technology in Bio-Informatics of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

[illegible]

Dr. Pradeep Kumar Naik
(Project Coordinator)
Senior Lecturer
Dept. of Bioinformatics and Biotechnology
Jaypee University of Information Technology
Waknaghat, Solan, Himachal Pradesh, India

ACKNOWLEDGMENT

It has been a great pleasure working under the able guidance of faculty & staff in the Department of Bioinformatics and Biotechnology at Jaypee University of Information Technology, during our study as a B.Tech student. Thanks a lot to Dr. Pradeep Kumar Naik, our project advisor and committee chair, for his financial and scientific support; otherwise, this research work would never have been possible

Many people have contributed to this project in a variety of ways over the past few months. To the individuals who have helped me, We again express our appreciation. We also acknowledge the many helpful comments received from our teachers of the concerned departments. We are indebted to all those who provided reviews & suggestions for improving the results and the topics covered in our project, and We extend our apologies to anyone We may have failed to mention.

K. Goutham

[illegible]

Goutham Koneru(031514) Srimath Kandalam Srikanth(31510)

TABLE OF CONTENTS

CERTIFICATE...	2
ACKNOWLEDGMENT..	3
LIST OF FIGURES...	6
LIST OF ABBREVIATIONS..	7
ABSTRACT..	8
Chapter 1: Introduction... • Nucleic Acid and Protein interactoin... • What are DNA Binding proteins................................... • Classifiation of DNA binding proteins... • Need of prediction and classification of DNA binding proteins... • Machine Learning classification........................ • Reference... • Objective...	1-29 10 10 12 20 20 27 29
Chapter 2: DBPPred: A tool for binary prediction of DNA Binding/Non-DNA Binding proteins from sequence derived features using ANN • Abstract... • Introduction... • Prediction model... • Results... • References...	30-46 31 32 33 36 37

Chapter 3: DBPPred: A Tool for prediction and classification of DNA Bindng proteins into four major classes using ANN from sequence derived features	47-71
• Abstract……………………………………………………	48
• Introduction………………………………………………	48
• Materials and method……………………………………	50
• Results……………………………………………………	53
• Discussion…………………………………………………	54
• References…………………………………………………	56
Chapter 4: DBPPred: A Tool for prediction and classification of DNA Bindng proteins into four major classes using HMM model……………………………………………	72-92
• Methods……………………………………………………	73
• A model of transition probability matrix generated for two way classification HMM ie.,DNA binding/Non DNA binding	78
• Viterbi Algorithm…………………………………………	81
• Results……………………………………………………	82
• Discussion…………………………………………………	89
• References…………………………………………………	90
• Conclusion…………………………………………………	92
Publications………………………………………………	
Appendix……………………………………………………	94

LIST OF FIGURES

Figure 1. Showing DNA binding protein interaction

Figure 2. Showing DNA binding protein interaction with Helix-Turn-Helix motif .

Figure 3. Showing DNA binding protein interaction with Leucine zipper.

Figure 4. Showing DNA binding protein interaction with Zinc finger motif .

Figure 5. Showing DNA binding protein interaction with homeobox domain .

Figure 6. Typical Artificial Neural network setup

Figure 7. Averaged values of 85 sequence derived parameters calculated independently within studied sets of dna binding and non dna biding.

Figure 8. Configuration of artificial neural network used to develop binary primary sequence descriptor model .

Figure 9. ROC curves for binary ANN network systems.

Figure 10. Configuration of artificial neural network used to develop binary primary sequence descriptor model for dna binding / non-dna binding proteins.

Figure11.Basic HMM model used in our study

Figure 12: A Markov chain for modeling a protein sequence. B and ε are the begin and end states, respectively.

Figure 13: HMM for the dishonest casino problem.

Figure14.Model of 2 state HMM

Figure15. Model for 4 state HMM

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
pI	Iso-electric Point
PSSM	Position Specific Scoring Matrix
PSI-BLAST	Position specific iterative BLAST
EC number	Enzyme Commission number
ORF	Open Reading Frame
PEPSTAT	Protein Statistics
SNNS	Stuttgart Neural Network Simulator
MCC	Mathew Correlation constant
ROC	receiver operating characteristic
PPV	Positive Predictive Values
CNS	Central Nervous System

ABSTRACT

The prediction of DNA Binding proteins and their classification into four major classes is one of the most important problems in biological world. Many previous algorithms have been developed to solve this intricacy. But, none of them has been able to provide a reliable method. The previous methods used a much complex method for prediction i.e from its 3-dimensional structure. It proved out to be a time consuming and a costly method with lot of limitation. A large number of data are constantly being generated thanks to several genome-sequencing projects throughout the world. However, the gap between the growth rate of biological sequences and the capability to characterize experimentally the roles and functions associated with these new sequences is constantly increasing. This results in an accumulation of raw data that can lead to an increase in our biological knowledge only if computational characterization tools are developed. The family of DNA-binding proteins is one of the most populated and studied amongst the various genomes of bacteria, archaea and eukaryotes. Understanding the molecular details of protein-DNA interactions is critical for deciphering the mechanisms of gene regulation. We focus here on the annotation of novel protein as DNA/Non-DNA Binding and if it is a DNA Binding protein its classification into four major classes.

In this study we have developed a machine learning approach for the identification of amino acid residues involved in protein-DNA interactions. A generic approach to this problem consists of transferring the annotation from sequences of known DNA Binding Proeteins to uncharacterised proteins. These approaches seemed initially to hold quite a bit of promise, that sequence derived features and HMM directly dictate the protein function. Classes of newly found DNA Binding sequences are usually determined either by biochemical analysis of eukaryotic and prokaryotic genomes or by microarray chips. However, with the explosion of protein sequences entering into databanks, it is highly desirable to explore the feasibility of selectively classifying newly found DNA Binding protein sequences into their respective classes by means of an automated method. This is indeed important because knowing which family or subfamily a protein belongs to may help deduce its catalytic mechanism and specificity, giving clues to the relevant biological function. Sequence similarity metrics are a useful approach to provide functional annotation, but its use is sometimes limited, prompting the development and use of

machine learning methods (MLMs). MLMs also have a certain degree of flexibility regarding data inputs, allowing them to expand progressively to meet the requirements of rapidly accumulating mountain of data generated from genomics research. Hence, in this study we have developed a one layer artificial neural network; the one layer is for prediction of 4 classes of DNA Binding/non-DNA binding from the protein sequence only using sequence derived features and a two layer Hidden Markov Model;the first layer is for binary prediction of DNA Binding/Non DNA Binding proteins.If it is predicted as DNA Binding protein in the second layer it is to classify the DNA binding proteins into its respective class out of 4 major classes. Both the methods are perfectly trained and validated and is predicting the result with more than 70% accuracy. DBPPred: A tool develop for prediction of DNA binding protein using the above mentioned machine learning techniques has been uploaded in the University Web Site (http://www.juit.ac.in/DBPPred.HTML)

Chapter 1
Introduction

Nucleic Acid & Protein interaction

Nucleic acids are the storage of genetic information and this information has to be accessible and inherited. The regulation of gene expression and the replication of the genome are central mechanisms to provide a 'translation' of the genetic blueprint into the machinery of life - the proteins. Nucleic acids need proteins for storage, replication, and transcription purposes. The specificity of the transcription and replication requires recognition on the molecular level between protein structures and nucleic acid structures. The genetic code has to be readable. The reading of the code is a conformationally specific interaction between amino acids and nucleic acids. Surface properties of the macromolecules involved are the essential key in the recognition process. Electrostatic interaction, hydrogen bonding capability and hydrophobic effects are of importance. Complementarity in surface profiles is the essential mechanism that provides the specificity.

What are DNA Binding Proteins

DNA-binding proteins are proteins that comprise any of many DNA-binding domains and thus have a specific or general affinity to DNA. A DNA-binding domain includes any protein motif that binds to double- or single-stranded DNA with affinity to a specific sequence or set thereof or a general affinity to DNA. DNA-binding domains are included in many proteins involved in the regulation of gene expression (including transcription factors), proteins involved in the packaging of DNA within the nucleus (such as histones), nucleic acid dependent-polymerases involved in DNA replication and transcription, or any of many accessory proteins which are involved in these processes. Proteins that bind DNA and are involved in replication or transcription do so in a sequence specific way. Transcription factors are dimers when active, i.e., they bind to DNA upon dimerization and are inactive in the monomeric form. Dimerization is a regulatory mechanism of controlling transcription factor activity. There are 3 common features most DNA binding proteins have in common:

1. the major groove is the binding site of proteins through □ -helices; the dimensions of the major groove is 12Å wide and 8Å deep
2. the minor groove of B-DNA is 5Å wide, 8Å deep, and is generally too narrow to fit entire □ -helices, but is recognized by □ -sheet structures of TATA box binding proteins
3. sequence specific DNA binding proteins generally do not disrupt the base pairs of the DNA, but do distort backbone conformation by bending the double helix

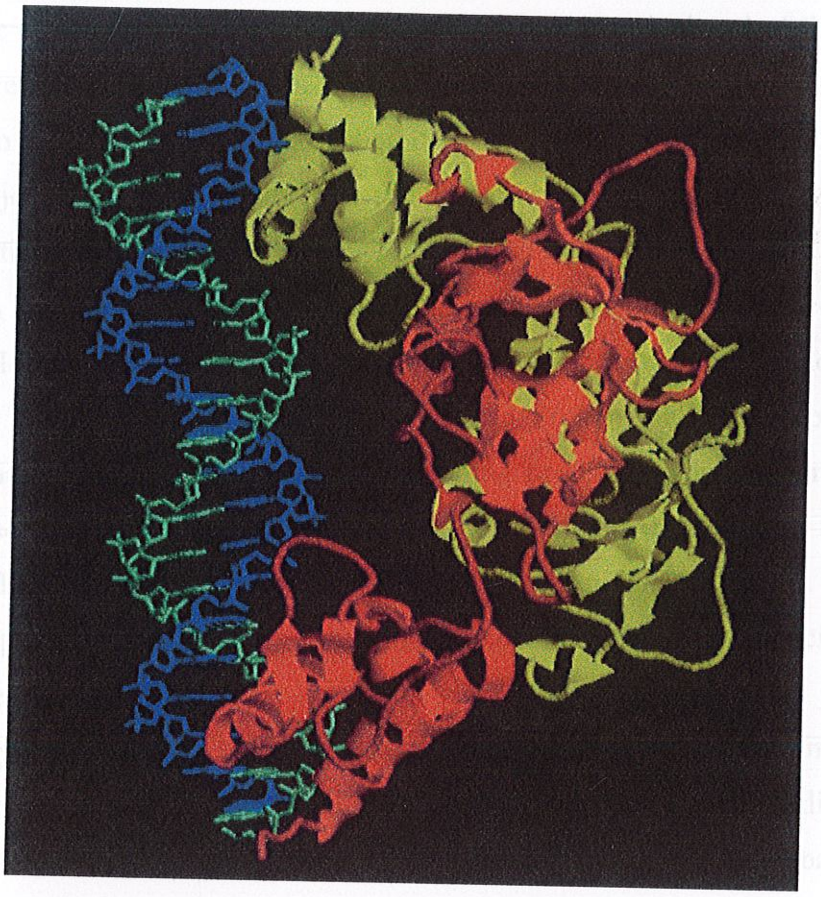


Figure 1. Showing DNA binding protein interaction.

Classification of DNA binding proteins

Helix-turn-helix motif (prokaryotic systems) :

In proteins, the **helix-turn-helix (HTH)** is a major structural motif capable of binding DNA. It is composed of two α helices joined by a short strand of amino acids and is found in many proteins that regulate gene expression.Its discovery was based on similarities between the genes for Cro, CAP, and λ repressor, which share a common 20-25 amino acid sequence that facilitates DNA recognition. In particular, recognition and binding to DNA is done by the two α helices, one occupying the N-terminal end of the motif, the other at the C-terminus. In most cases, such as in the Cro repressor, the second helix contributes most to DNA recognition, and hence it is often called the "recognition helix". It binds to the major groove of DNA through a series of hydrogen bonds and various Van der Waals interactions with exposed bases. The other α helix stabilizes the interaction between protein and DNA, but does not play a particularly strong role in its recognition.The Helix-Turn-Helix motif consists of two a helices and a short extended amino acid chain between them. The more carboxyl-terminal helix can fit into the major groove of DNA. This motif is found in hundreds of DNA-binding proteins, including l repressor, tryptophan repressor, catabolite activator protein (CAP), octamer transcription factor 1 (Oct-1) and heat shock factor (HSF), Homeotic genes are known to play a critical role in fly development. Their products contain a homeodomain which is a special form of the helix-turn-helix motif. The helix-turn-helix motif is the common DNA recognition motif in prokaryotes. The motif resembles that of an EF hand described in calmodulin . The F-helix is the recognition helix and the side chains give the specificity of binding. Sometimes more than one protein compete for the same sequence. As examples serve bacteriophages l , P22, and 434, where repressors and activators affect transcription. They can recognize the same DNA fragment. The phage 434 *cI repressor* and *cro* have been shown *in vitro* to bind to the same 20bp DNA fragment. They have different binding interaction visualized by the two close, but not identical structures as determined by X-ray crystallography. They differ in their affinity for the same sequence, or DNA conformation, respectively through H-bonds, salt bridges and Van der Waals interactions. The *relative concentrations* of all proteins, therefore, determine which one is bound to the binding element most of the time, which in turn affects polymerase binding to the DNA determining if and in which direction transcription will occur. The ratio of all DNA binding proteins therefore determines the rate of transcription controlled by the DNA sequences in question.

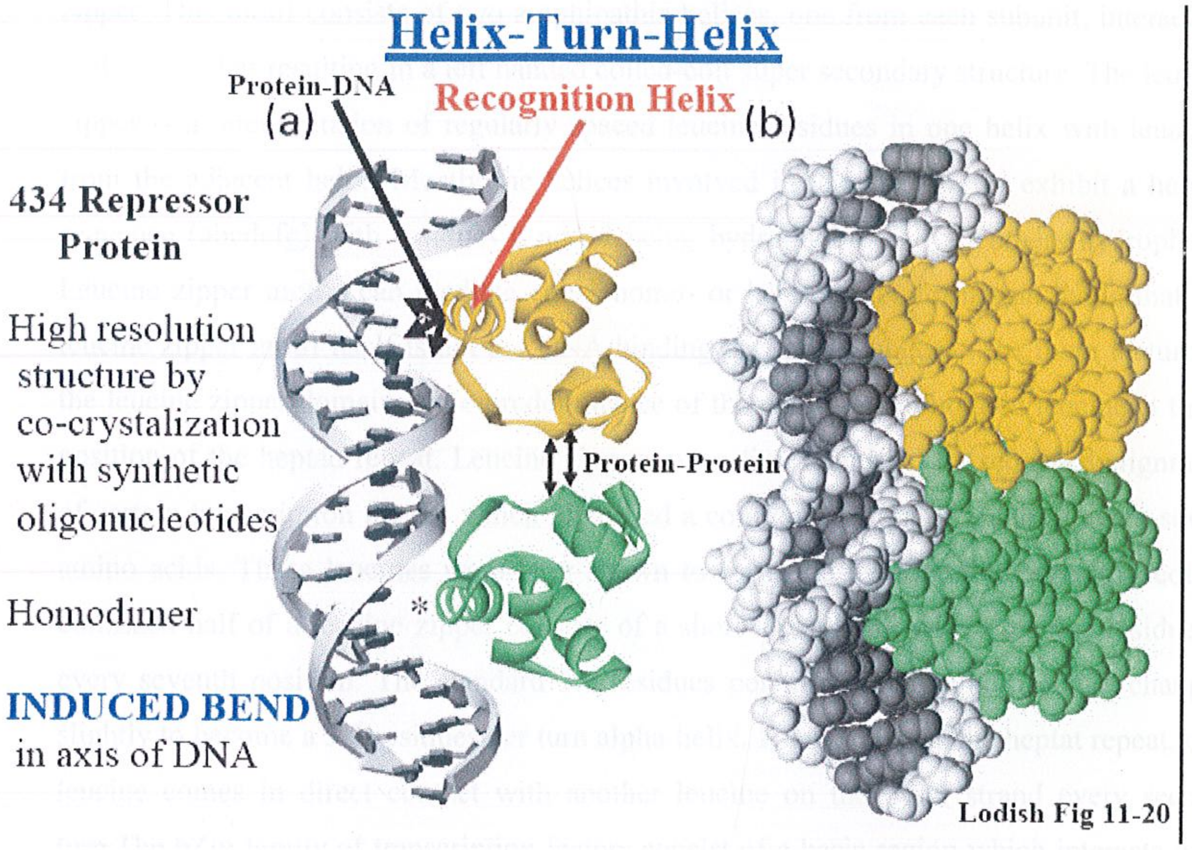


Figure 2. Showing DNA binding protein interaction with Helix-Turn-Helix motif .

Leucine-zipper (Eukaryotic transcription factors) :

In some transcription factors the dimer binding site with the DNA forms a so called leucine zipper. This motif consists of two amphipathic helices, one from each subunit, interacting with each other resulting in a left handed coiled-coil super secondary structure. The leucine zipper is a interdigitation of regularly spaced leucine residues in one helix with leucines from the adjacent helix. Mostly the helices involved in leucine zippers exhibit a heptad sequence (abcdefg) with residues a and d being hydrophobic and all others hydrophilic. Leucine zipper motifs can mediate either homo- or heterodimer formation. Note that the leucine zipper motif itself is not the DNA binding part of the helices. The main feature of the leucine zipper domain is the predominance of the common amino acid leucine at the d position of the heptad repeat. Leucine zippers were first identified by sequence alignment of certain transcription factors which identified a common pattern of leucines every seven amino acids. These leucines were later shown to form the hydrophobic core of a coiled coil.Each half of a leucine zipper consists of a short alpha-helix with a leucine residue at every seventh position. The standard 3.6 residues per turn alpha-helix structure changes slightly to become a 3.5 residues per turn alpha-helix. Known also as the heptat repeat, one leucine comes in direct contact with another leucine on the other strand every second turn.The bZip family of transcription factors consist of a basic region which interacts with the major groove of a DNA molecule through hydrogen bonding, and a leucine zipper region which is responsible for dimerization. Leucine zipper regulatory proteins include fos and jun (the AP1 transcription factor), important regulators of normal development. If they are overproduced or mutated in a vital area, they may generate cancer. These proteins interact with the DNA as dimers (homo- or hetero-) and are also called basic zipper proteins (bZips).

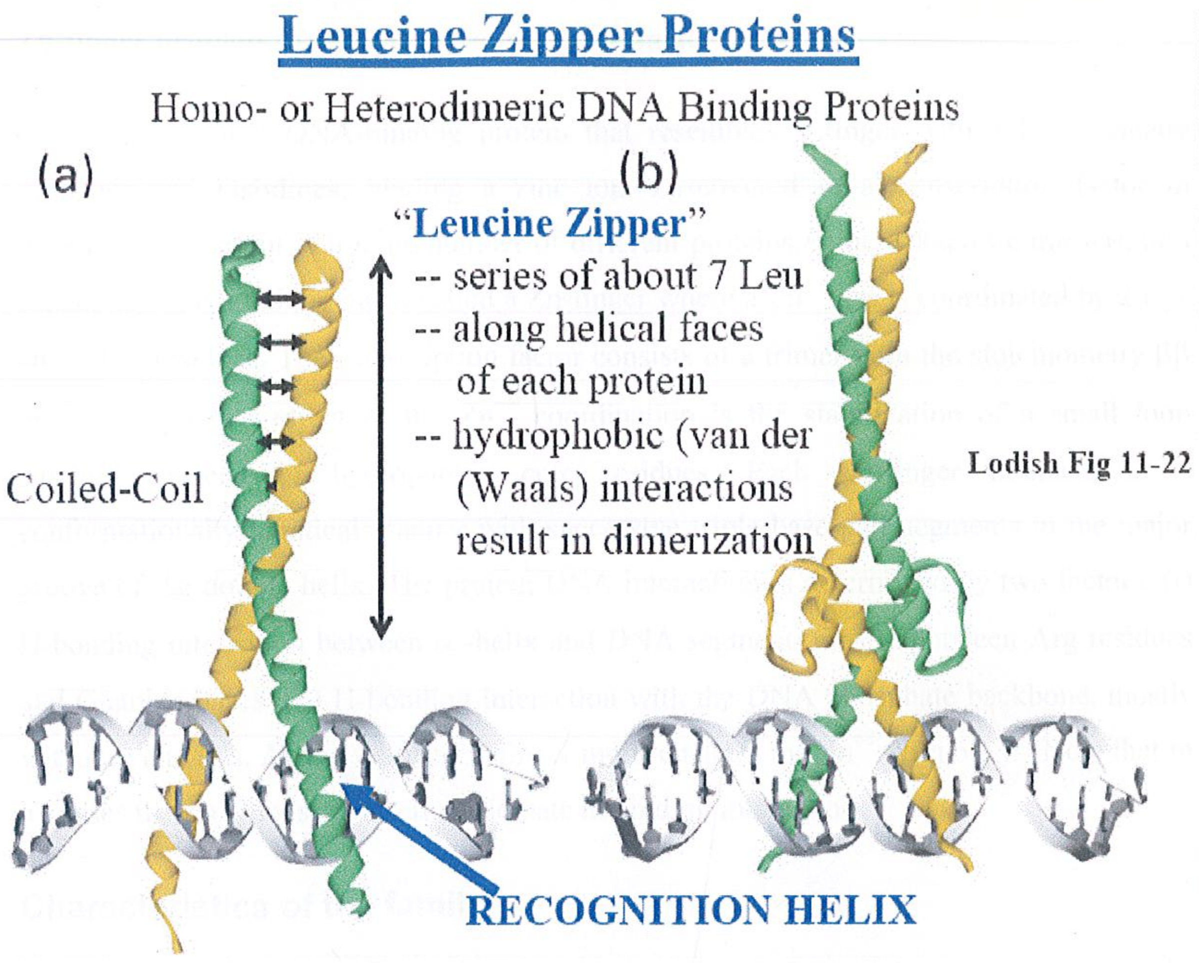

Leucine Zipper Proteins
Homo- or Heterodimeric DNA Binding Proteins
(a)
(b)
"Leucine Zipper"
-- series of about 7 Leu
-- along helical faces of each protein
-- hydrophobic (van der (Waals) interactions result in dimerization
Coiled-Coil
Lodish Fig 11-22
RECOGNITION HELIX

Figure 3. Showing DNA binding protein interaction with Leucine zipper.

Zn-finger proteins (Eukaryotic transcription factors) :

Configuration of a DNA-binding protein that resembles a finger with a base, usually cysteines and histidines, binding a zinc ion. Discovered in a transcription factor in Xenopus but present in a large number of different proteins.Some eukaryotic transcription factors showed a unique motif called a Zn-finger where a Zn^{++} ion is coordinated by 2 Cys and 2 His residues. The transcription factor consists of a trimer with the stoichiometry ββ 'α . The apparent effect of the Zn^{++} coordination is the stabilization of a small loop structure instead of hydrophobic core residues. Each Zn-finger interacts in a conformationally identical manner with successive triple base pair segments in the major groove of the double helix. The protein-DNA interaction is determined by two factors: (i) H-bonding interaction between α -helix and DNA segment, mostly between Arg residues and Guanine bases. (ii) H-bonding interaction with the DNA phosphate backbone, mostly with Arg and His. An alternative Zn-finger motif chelates the Zn^{++} with 6 Cys. Note that in all cases the Zn^{++} does not itself participate in binding interaction.

Characteristics of the family:

Function: The DNA-binding motif is found as part of transcription regulatory proteins.

Structure: One of the most abundant DNA-binding motifs. Proteins may contain more than one finger in a single chain; each motif consists of 2 antiparallel beta-strands followed by by an alpha-helix. A single zinc ion is tetrahedrally coordinated by conserved histidine and cysteine residues, stabilising the motif.

Binding: Fingers bind to 3 base-pair subsites and specific contacts are mediated by amino acids in positions -1, 2, 3 and 6 relative to the start of the alpha-helix. Contacts mainly involve one strand of the DNA. Where proteins contain multiple fingers, each finger binds to adjacent subsites within a larger DNA recognition site thus allowing a relatively simple motif to specifically bind to a wide range of DNA sequences.

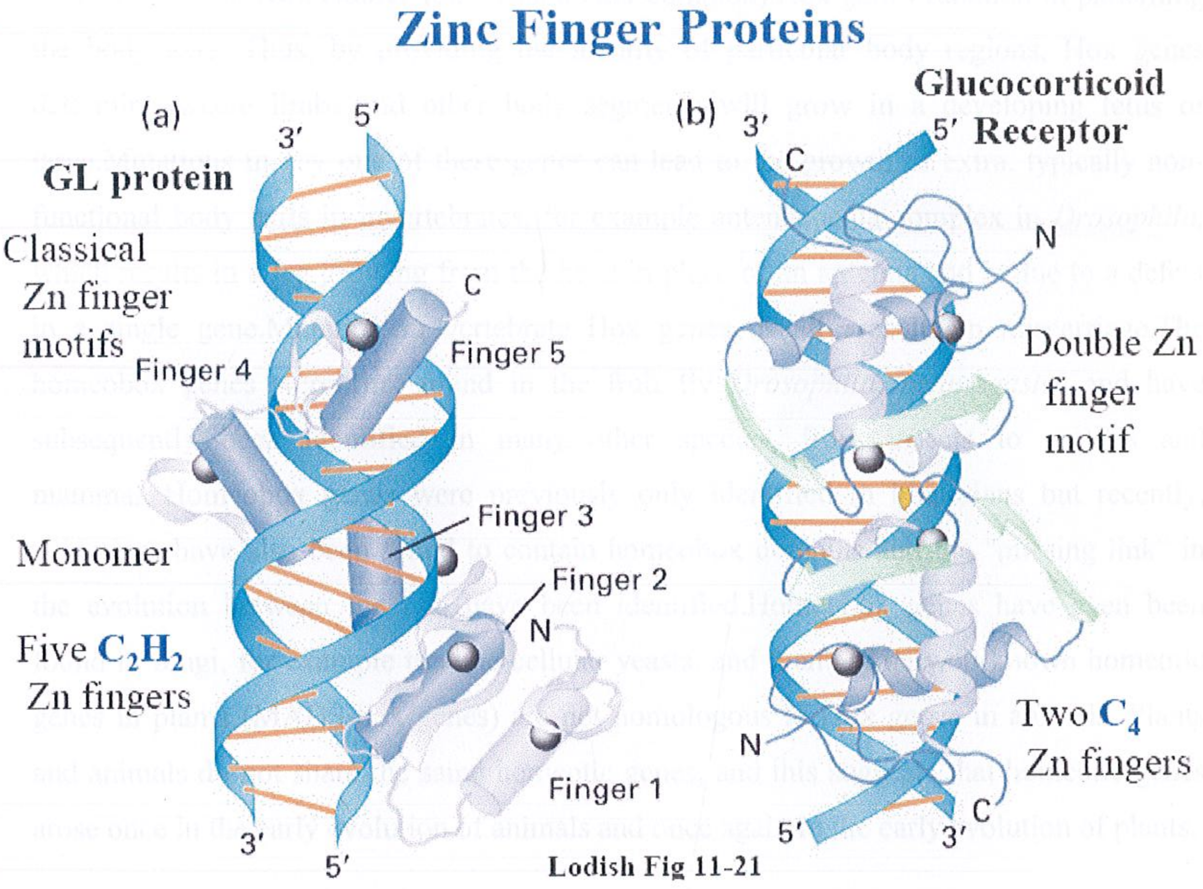


Figure 4. Showing DNA binding protein interaction with Zinc finger motif .

Homeobox proteins

A homeobox is about 180 base pairs long; it encodes a protein domain (the homeodomain) which can bind DNA.Homeobox genes encode transcription factors which typically switch on cascades of other genes, for instance all the ones needed to make a leg. The homeodomain binds DNA in a specific manner.However, the specificity of a single homeodomain protein is usually not enough to recognize only its desired target genes. Most of the time, homeodomain proteins act in the promoter region of their target genes as complexes with other transcription factors, often also homeodomain proteins. Such complexes have a much higher target specificity than a single homeodomain protein.A particular subgroup of homeobox genes are the **Hox genes**, which are found in a special gene cluster, the **Hox cluster** (also called Hox complex).Hox genes function in patterning the body axis. Thus, by providing the identity of particular body regions, Hox genes determine where limbs and other body segments will grow in a developing fetus or larva.Mutations in any one of these genes can lead to the growth of extra, typically non-functional body parts in invertebrates, for example antennapedia complex in *Drosophila,* which results in a leg growing from the head in place of an antenna and is due to a defect in a single gene.Mutation in vertebrate Hox genes usually results in miscarriage.The homeobox genes were first found in the fruit fly *Drosophila melanogaster* and have subsequently been identified in many other species, from insects to reptiles and mammals.Homeobox genes were previously only identified in bilaterians but recently, cnidarians have also been found to contain homeobox domains and the "missing link" in the evolution between the two have been identified.Homeobox genes have even been found in fungi, for example the one-cellular yeasts, and plants. The well known homeotic genes in plants (MADS-box genes) are not homologous to Hox genes in animals. Plants and animals do not share the same homeotic genes, and this suggests that homeotic genes arose once in the early evolution of animals and once again in the early evolution of plants.

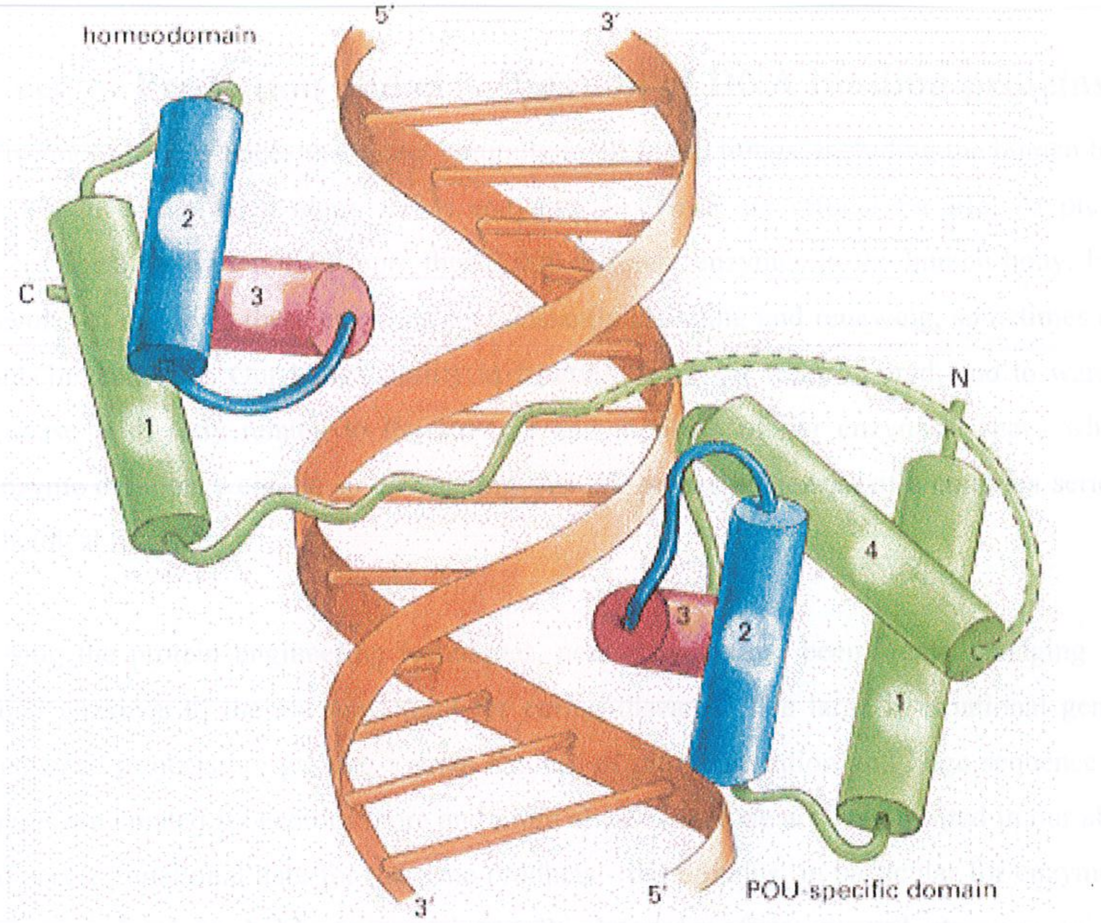


Figure 5. Showing DNA binding protein interaction with homeobox domain .

Need of Prediction and classification of DNA binding proteins

Enzymes are substances that occur naturally in all living things, including the human body. If it's an animal or a plant, it has enzymes. Enzymes are critical for life. At present, researchers have identified more than 3,000 different enzymes in the human body. Every second of our lives these enzymes are constantly changing and renewing, sometimes at an unbelievable rate. Our body's ability to function, to repair when injured, and to ward off disease is directly related to the strength and numbers of our enzymes. That's why an enzyme deficiency can be so devastating. All life processes consist of a complex series of chemical reactions.

Using the protein engineering techniques, new enzymes are been created, ranging from food enzymes to the enzymes used for curing diseases. The large international genome sequence projects are gaining a great amount of public attention and huge sequence data bases are created it becomes more and more obvious that we are very limited in our ability to access functional data for the gene products - the proteins, in particular for enzymes. It seems quite improbable to experimentally determine function and structure of each candidate protein. So a revolutionary method is needed to solve this computation catastrophe. Primary sequence of these proteins are readily available, therefore a method using the sequence derived features will prove a much valuable and a cost effective process of determining and classifying these proteins into broader enzyme/non-enzyme and specifically into 6 major classes as defined by international enzyme commission.

Machine Learning in Classification

As a broad subfield of artificial intelligence, **machine learning** is concerned with the design and development of algorithms and techniques that allow computers to "learn". At a general level, there are two types of learning: inductive, and deductive. Inductive machine learning methods extract rules and patterns out of massive data sets. The major focus of Machine learning research is to extract information from data automatically by computational and statistical methods, hence, machine learning is closely related to data mining and statistics but also theoretical computer science [Christopher M. Bishop (2007)]. Machine learning has a wide spectrum of applications including natural language processing, syntactic pattern recognition, search engines, medical diagnosis, bioinformatics and cheminformatics, detecting credit card fraud, stock market analysis, classifying DNA sequences, speech and handwriting recognition, object recognition in computer vision, game playing and robot locomotion.

NEURAL NETWORKS:

Neural Network or more appropriately Artificial Neural Network is basically a mathematical model of what goes in our mind (or brain). The brain of all the advanced living creatures consists of neurons, a basic cell, which when interconnected produces what we call Neural Network. The sole purpose of a Neuron is to receive electrical signals, accumulate them and see further if they are strong enough to pass forward. The basic functionality lies not in neurons but the complex pattern in which they are interconnected. NNs are just like a game of chess, easy to learn but hard to master. In the same way, a single neuron is useless. Well, practically useless. It is the complex connection between them and values attached with them which makes brains capable of thinking and having a sense of consciousness (much debated).

Basic working principle of a neuron

A neuron is basically a cell which accumulates electrical signals with different strengths. What it does more is that it compares the accumulated signal with one predefined value unique to every neuron. This value is called bias. Function of a neuron could be explained in the following diagram.

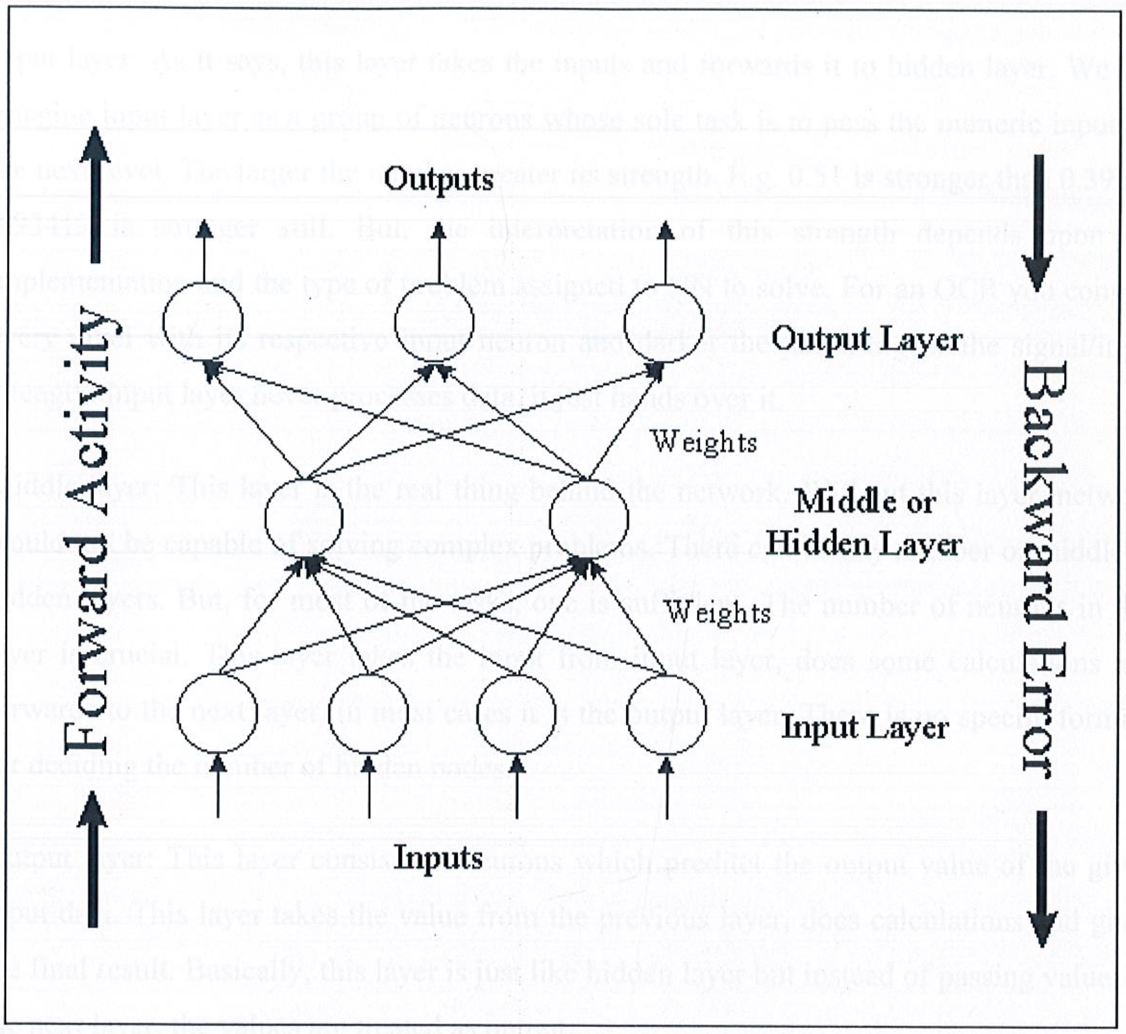


FIGURE 6. Typical Artificial Neural Network Setup (Caudill and Butler, 1992a).

Image Source: http://www.interwet.psu.edu/f41.gif

The circles in the image represent neurons. This network or more appropriately this network topology is called feed-forward multi layered neural network. It is the most basic and most widely used network. The network is called multi layered because it consists of more than two layers. The neurons are arranged in a number of layers, generally three. They are input, hidden/middle and output layers. This network is feed-forward, means the values are propagated in one direction only. There are many other topologies in which values can be looped or move in both forward and backward direction. But, this network allows the movement of values only from input layer to output layer. The functions of various layers are explained below:

Input layer: As it says, this layer takes the inputs and forwards it to hidden layer. We can imagine input layer as a group of neurons whose sole task is to pass the numeric inputs to the next level. The larger the number greater its strength. E.g. 0.51 is stronger than 0.39 but 0.93412 is stronger still. But, the interpretation of this strength depends upon the implementation and the type of problem assigned to NN to solve. For an OCR you connect every pixel with its respective input neuron and darker the pixel, higher the signal/input strength. Input layer never processes data, it just hands over it.

Middle layer: This layer is the real thing behind the network. Without this layer, network would not be capable of solving complex problems. There can be any number or middle or hidden layers. But, for most of the tasks, one is sufficient. The number of neurons in this layer is crucial. This layer takes the input from input layer, does some calculations and forwards to the next layer, in most cases it is the output layer. There is no specifc formula for deciding the number of hidden nodes.

Output layer: This layer consists of neurons which preditct the output value of the given input data. This layer takes the value from the previous layer, does calculations and gives the final result. Basically, this layer is just like hidden layer but instead of passing values to the next layer, the values are treated as output.

Dendrites: These are straight lines joining two neurons of consecutive layers. They are just a passage (or method) through which values are passed from one layer to the next. There is a value attached with dendrite called weight. The weight associated with dendrites basically determines the importance of incoming value. A weight with larger value determines that the value from that particular neuron is of higher significance. To achieve this we do is multiply the incoming value with weight. So no matter how high the value is, if the weight is low the multiplication yields the final low value.

Training: Training is the most important part of a neural network and the one consisting of the most mathematics. It uses Backpropagation method for training the NN. The best example illustrating this principle is Charles Darwin(what?). Yes, at the time when he wrote 'On the Origin of Species', DNA was not known. So, he propounded the evolution without even knowing the method of how it is done i.e. how traits are passed on from parents to offspring. Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimises the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. Evolutionary methods, simulated annealing, and expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks. This training procedure must be repeated for larger number of samples so that the NN can produce accurate results for untrained input samples.

Applications: The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

Real life applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.

Application areas include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

Hidden Markov Model

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. A HMM can be considered as the simplest dynamic Bayesian network.In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states.Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following and bioinformatics.

Architecture of a hidden Markov model

The diagram below shows the general architecture of an HMM. Each oval shape represents a random variable that can adopt a number of values. The random variable x(t) is the value of the hidden variable at time t. The random variable y(t) is the value of the observed variable at time t. The arrows in the diagram denote conditional dependencies.From the diagram, it is clear that the value of the hidden variable x(t) (at time t) only depends on the value of the hidden variable x(t − 1) (at time t − 1). This is called the Markov property. Similarly, the value of the observed variable y(t) only depends on the value of the hidden variable x(t) (both at time t).

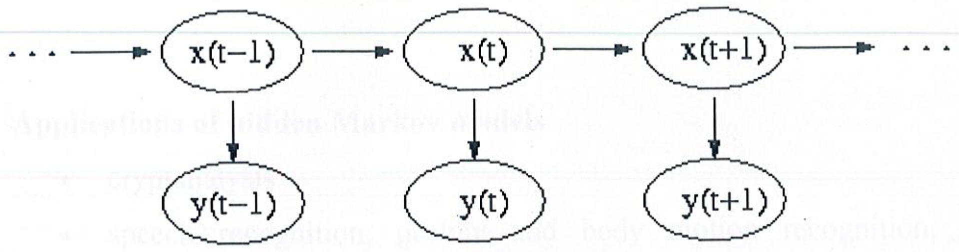


Figure.7. Architecture of a hidden Markov model

Probability of an observed sequence

The probability of observing a sequence $Y = y(0), y(1), \dots, y(L-1)$ of length L is given by

$$P(Y) = \sum_X P(Y \mid X)P(X),$$

where the sum runs over all possible hidden node sequences $X = x(0), x(1), \dots, x(L-1)$. Brute force calculation of P(Y) is intractable for most real-life problems, as the number of possible hidden node sequences is going to be extremely high. The calculation can however be sped up enormously using an algorithm called the forward-backward procedure .

Using hidden Markov models

There are three canonical problems associated with HMMs:

- Given the parameters of the model, compute the probability of a particular output sequence. This problem is solved by the forward-backward algorithm.
- Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This problem is solved by the Viterbi algorithm.
- Given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. In other words, train the parameters of the HMM given a dataset of sequences. This problem is solved by the Baum-Welch algorithm.

Applications of hidden Markov models

- cryptanalysis
- speech recognition, gesture and body motion recognition, optical character recognition
- machine translation
- musical score following
- bioinformatics and genomics
    - prediction of protein-coding regions in genome sequences
    - modeling families of related DNA or protein sequences
    - prediction of secondary structure elements from protein primary sequences

References

- National Cancer Institute: Vaginal Cancer (public domain)
- [Stenchever: Comprehensive Gynecology, 4th ed., Copyright © 2001 Mosby, Inc.]
- The Image Processing Handbook by John C. Russ, ISBN 0849372542 (2006)

  Fundamentals of Image Processing by Ian T. Young, Jan J. Gerbrands, Lucas J. Van Vliet,

  Paperback, ISBN 90-75691-01-7 (1995)
- Image Analysis and Mathematical Morphology by Jean Serra, ISBN 0126372403 (1982)
- Front-End Vision and Multi-Scale Image Analysis by Bart M. ter Haar Romeny, Christopher M. Bishop (2007) Pattern Recognition and Machine Learning, Springer ISBN 0-387-31073-8.
- Neural Computing and Applications, Springer-Verlag. (address: Sweetapple Ho, Catteshall Rd., Godalming, GU7 3DJ)
- Bhagat, P.M. (2005) Pattern Recognition in Industry, Elsevier. ISBN 0-08-044538-1
- Bishop, C.M. (1995) Neural Networks for Pattern Recognition, Oxford: Oxford University

Press. ISBN 0-19-853849-9 (hardback) or ISBN 0-19-853864-2 (paperback)

- Duda, R.O., Hart, P.E., Stork, D.G. (2001) Pattern classification (2nd edition), Wiley,
  ISBN 0-471-05669-3
- Gurney, K. (1997) An Introduction to Neural Networks London: Routledge.
- ISBN 1-85728-673-1 (hardback) or ISBN 1-85728-503-4 (paperback)
- Haykin, S. (1999) Neural Networks: A Comprehensive Foundation, Prentice Hall,

  ISBN 0-13-273350-1

Objective:

Rational classification of proteins encoded in sequenced genomes is critical for making the genome sequences maximally useful for functional and evolutionary studies. The family of DNA-binding proteins is one of the most populated and studied amongst the various genomes of bacteria, archaea and eukaryotes and the method presented here is an approach to their classification. Sequence similarity metrics are a useful approach to provide functional annotation, but its use is sometimes limited, prompting the development and use of machine learning methods (MLMs). MLMs also have a certain degree of flexibility regarding data inputs, allowing them to expand progressively to meet the requirements of rapidly accumulating mountain of data generated from genomics research.

Hence, in this study an attempt has been taken to develop an automated tool using machine learning technique for annotation of protein sequence with following objectives.

1. To extract sequence derived features and selection of important features from protein sequence to be used for prediction and classification of dna binding proteins.

2. To develop and optimized the artificial neural network for prediction of dna binding (4 major classes) /non dna binding using sequence derived features.

3. To develop a two layer Hidden markov model for binary prediction and as well as classification of dna binding proteins into 4 major classes.

4. To validate the develop neural network model for predicting and classification of dna binding proteins in some organisms.

5. To develop an automated tool for prediction and classification of dna binding proteins from the protein sequence and to upload in the university web server for public uses.

Chapter 2

DBPPred: A tool for binary prediction of DNA Binding/Non-DNA Binding proteins from sequence derived features using ANN

(The developed tool is uploaded in the university web server and the tool is communicated for publication in *Protein: Structure, Function and Bioinformatics,* Wiley Publisher.)

ABSTRACT

The problem of predicting the DNA binding and Non DNA binding from protein sequence information is still an open problem in bioinformatics. It is further becoming more important as the number of sequenced information grows exponentially over time. Sequence similarity metrices are a useful approach to provide functional annotation, but its use is sometimes limited, prompting the development and use of machine learning methods. We describe a novel approach for predicting the DNA Binding and Non DNA Binding from its amino-acid sequence using artificial neural network (ANN) and hidden markov model (HMM).The ANN used in this study is free-forward neural network with a standard back propogation traning algorithm. Using 85 sequence features alone we have been able to achieve 70% correct prediction of proteins into DNA binding/non DNA binding (in the set of 500 proteins). For the complete set of 85 parameteres using 5-fold cross-validated classification, ANN model reveal a superior model (accuracy=72.19 , Q_{pred} = 69.36 , sensitivity = 80.06, specificity = 74.56).The second module is based on hidden markov model (HMM). The HMM used in this study is a simple statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. Using the extracted model parameters we have been able to achieve 70% correct prediction of proteins into DNA binding/non DNA binding. HMM model reveal a superior model (accuracy = 69.76 , Q_{pred} = 68, sensitivity = 75.35, specificity = 68.32).

1. INTRODUCTION

The rapid progress in genome analysis has made available the complete genome sequences of many organisms. Subsequent annotation of the genes, enabling their function to be inferred from sequence homology, is an important next step in the post-sequence analysis of genomes. In that regard, X-ray crystallographic and NMR spectroscopic analyses of DNAbinding proteins, which play key roles in the regulation of gene expression, have provided valuable information about the general features of protein–DNA interactions. In recent years, for example, Luscombe and Thornton (2002) analysed amino acid conservation and the effects of mutations on the binding specificity within protein–DNA complexes. Pabo and Nekludova (2000) developed geometrical models for characterizing side chain–base interactions and in related studies, Mandel-Gutfrend and Margalit (1998) and Mandel- Gutfrend et al. (1998) demonstrated the importance of hydrogen bonding and hydrophobic and CH ??·O interactions to protein–DNA interactions. Nadassy et al. (1999) analysed the importance of the interface surface area between the protein and the DNA for protein–DNA recognition. In addition, our systematic analysis of the contacts between amino acids and base pairs in a set of protein–DNA complexes has enabled us to construct models with which to predict the DNA target sites of regulatory proteins. Our analysis of base–amino acid interactions and have applied it successfully to cognate, non-cognate, symmetric and asymmetric binding of DNA-binding proteins. Still, the mechanism underlying protein–DNA recognition is not yet completely understood. In the present work, we analysed the characteristic features of DNA-binding proteins and their binding sites in order to determine the factors that distinguish binding proteins from non-binding ones and binding residues from all others. The data sets used for this analysis included (i) nonredundant protein–DNA complexes containing information about the structure and location of binding sites within DNAbinding proteins and (ii) a non-redundant database of DNAbinding proteins for which structural information and the location of DNA-binding sites were not known.

The family of DNA-binding proteins is one of the most populated and studied amongst the various genomes of bacteria, archea and eukaryotes. Most of these proteins, such as the eukaryotic and prokaryotic transcription factors, contain independently folded units (domains) in order to accomplish their recognition with the contours of DNA. It is now clear that the majority of these DNA-binding scaffolds which are in general relatively small, less than 100 amino acid residues, belong to a large number of structural families with characteristic sequences and three-dimensional designs or conformations (Branden and Tooze, 1999).

Hence, in this study we have develop two different methods viz., neural networks and Hidden markov model which extract valuable information from protein sequence only for prediction into DNA Binding/Non-DNA Binding proteins. The neural network used sequence derived features derived from PEPSTAT (EMBOSS suite), which would be useful for the systematic analysis of small or medium size protein sequences.HMM model was developed for classification of DNA Binding proteins Results are discussed, assessing the benefits of using this methodology in binary prediction of DNA Binding / Non-DNA Binding proteins. The preliminary results suggest that sequence derived feature can be used as a fast and effective classification methodology for proteins.

2. PREDICTION MODEL FOR ANN

2.1. Training data

To discriminate between the dna binding and non dna binding proteins a set of 500 proteins , consisting of 250 non redundant dna-binding proteins and the same number of non redundant non-dna binding proteins, were used for training and testing.The dna-binding proteins dataset used in this study is obtained from the PDB database (http://www.rcsb.org). The dna binding protein datasets used in this study consists of almost equal number of enzyme dna binding proteins for each 4 major classes (62 zinc finger, 62 leucine zipper, 62 helix-turn-helix, 62 homeo box). The pairwise sequence identities in the datasets are less than 70% for both dna-binding and non-dna binding protein classes.Sequences of all the proteins along with their name and pdb id are included in the supplementary material.

2.2. Sequence derived parameters calculation and selection

To build a binary ANN model enabling effective prediction of enzymes/non-enzymes we initially calculated 85 parameters (Table.1.) from the protein sequence alone using PEPSTAT (EMBOSS suite) ftp://emboss.open-bio.org/pub/EMBOSS (Rice et al.,

2000) for all 500 protein sequences. The average values of these 85 parameters independently calculated for enzymes and non- enzymes have been plotted onto Figure 8. It showed clear distinction between dna binding and non-dna binding proteins based on 85 parameters. The normalized values have been then used to generate ANN models for binary prediction.

2.3. Fivefold cross-validation

A prediction method is often developed by cross-validation or jack-knife method (Chou and Zhang, 1995). Because of the size of the dataset, the jack-knife method (individual testing of each protein in the data set) was not feasible. So a more limited cross-validation technique has been used, in which the dataset is randomly divided into five subsets, each containing equal number of protein sequences. Each set is a balanced set that consist of 50 percent of dna binding proteins and 50 percent non-dna binding proteins. The data set has been divided into training and testing set. The training set consists of five subsets. The network is validated for minimum error on testing set to calculate the performance measure for each fold of validation. This has been done five times to test for each subset. The final prediction results have been averaged over five testing sets.

2.4. ANN model for prediction of dna binding/non-dna binding using sequence derived features

. In this study we have used the standard back-propagation ANN configuration consisting of 85 inputs and 1 output node in order to discriminate between dna binding /non-dna binding proteins from the training sets (Figure.9.). For each sequence in the training and testing sets, we have transformed 85 network input parameters into the normalized values varying from 0 to 1. Similarly, the output parameters from the ANN were normalized to [0:1] range. The number of nodes in the hidden layer was varied from 1 to 30 in order to find the optimal network that allows most accurate separation of dna binding/non-dna binding proteins. in the training sets (Table .2.). During the learning phase, a value of 1 was assigned for the dna binding sequence and 0 for non-dna binding sequence. For each configuration of the ANN (with 1, 3, 5, 7, 9 and so on upto 30hidden nodes respectively) 119 independent training runs were performed to evaluate the average predictive power of the network. The corresponding counts of the false/true positive and negative predictions were estimated using 0.1 and 0.9 cut-off values for dna binding and non-dna binding proteins respectively. Thus, a protein sequence from the testing set was considered correctly predicted by the ANN only when its output value ranged from 0.9 to 1.0. For each non-dna binding protein of the testing set the correct prediction was assumed if the corresponding ANN output lies between 0 and 0.1. Thus, all network output values ranging from 0.2 to 0.9 have been ultimately considered as incorrect predictions (rather than undetermined or non-defined).

2.5. Performance measures

The prediction results of ANN model developed in the study were evaluated using the following statistical measures.

1. *Accuracy of the methods*: The accuracy of prediction for neural network models were calculated as follows:

$$Q_{ACC} = \frac{P+N}{T}, \text{ where } T = (P+N+O+U)$$

Where *P* and *N* refer to correctly predicted DNA binding and Non-DNA binding, and *O* and *U* refer to over and under predictions, respectively.

2. Sensitivity (Q_{sens}) and specificity (Q_{spec}) of the prediction methods are defined as:

$$Q_{sens} = \frac{P}{P+U}$$

$$Q_{spec} = \frac{N}{N+O}$$

3. Q_{Pred} (Probability of correct prediction) and Q_{obs} (Percentage over coverage) are defined as:

$$Q_{pred} = \frac{P}{P+O} \times 100$$

The receiver operating characteristic (ROC) curve is also used to evaluate the prediction accuracy of our system using both sequence derived features.

3. RESULTS

3.1. Predictability of DNA binding proteins with sequence derived features

The ANN model (85-7-1) is trained with the sequence derived features (85 parameters) calculated using PEPSTAT. When applying a fivefold cross-validation test using five data sets, we found that the network reached an overall accuracy of 70.79 ± 6.86 %. The prediction results are presented in (Table .3.). The other performance measures are: Qpred = 65.789 ± 11.012 %, sensitivity = 80.70 ± 3.73 % and specificity = 70.66 ± 10.39 %. The value of the learning parameter was set to 0.1. Training was performed for 119 epochs for both the networks, after which the learning has been terminated when the error reached a stable value; differences between errors in subsequent steps become sufficiently small. Table.4. reveals the predictability of dna binding and non-dna binding of the network. Out of 100 in each cross validation set 50-70 sequences were correctly predicted as dna binding. However,out of 100 in non-dna binding claa; 45-60 were correctly predicted as non-dna binding. . Prediction performance measures were averaged over five sets. Figure 9 features averaged frequencies of the output values for the five testing sets used in the study. As it can readily be seen from the graph, the vast majority of the predictions have been contained within (0.0 – 0.1) for non-enzymes and (0.9 – 1.0) for enzymes in case of sequence derived module. This illustrates that 0.1 and 0.9 cut-offs values provide very adequate separation of two bioactive classes using ANN. All network output values ranging from 0.1 to 0.9 have been ultimately considered as incorrect predictions (rather than undetermined or non-defined).

3.2. Evaluation of prediction accuracy

From a practical point of view the most important aspect of a prediction method is its ability to make correct predictions. As prediction methods are never perfect, one always faces the dilemma of choosing between making few false-positive predictions and having a high sensitivity, that is, correctly identifying as many positive examples as possible. This tradeoff can be visualized as what is known as the receiver output characteristic (ROC) curve, in which the sensitivity is plotted as a function of the 1-specificity by varying the score threshold used for making positive predictions. Figure .10. shows the ROC curves for the two predictors included in our method. Performance of the network has been evaluated by calculating the area under the ROC curve. The area under the curve is 0.90 for sequences derived features; revealing a better discrimination of network system.

3.3. An application of the model

The reliability of developed model of binary prediction of dna binding and non-dna binding proteins was tested on the 500 annotated protein sequences of two organisms downloaded form GenBank (Benson et al. 2002). The predicted result is shown in Table .5. Prediction on new sequences is done by first running the PEPSTAT (EMBOSS) program to obtain the sequence-derived features, which are subsequently used as input for the prediction of dna binding and non-dna binding proteins.

References

Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389–3402.

Anfinsen, C.B. 1973 Principles that govern the folding of protein chains. *Science* 181, 223–230.

Baker, E.N., Arcus, V.L., and Lott, J.S. 2003. Protein structure prediction and analysis as a tool for functional genomics. *Applied Bioinformatics* 2, (3 Suppl), s3-s10.

Benson, D., Karsch-Mizrachi, I., Lipman, D., Ostell, J., Rapp, B., and Wheeler, D. 2002. GenBank. *Nucleic Acids Res.* 30, 17–20.

Boberg, J., Salakoski, T. & Vihinen, M. 1995. Accurate prediction of protein secondaryst ructural class with fuzzyst ructural vectors. *Protein Eng.* 8, 505–512.

Bork, P., and Koonin, E.V. 1998. Predicting functions from protein sequences - where are the bottlenecks? *Nat Genet.* 18, 313–318.

Bu, W.S., Feng, Z.P., Zhang, Z.D. and Zhang, C.T. 1999. Prediction of protein (domain) structural classes based on amino-acid index. *Eur. J. Biochem.* 266, 1043–1049.

Cai, C.Z., Wang, W.L., Sun, L.Z., and Chen, Y.Z. 2003. Protein function classification via support vector machine approach. *Math Biosci.* 185(2),111-122.

Chou, P.Y. 1989. Prediction of protein structural classes from amino acid composition. In Prediction of Protein Structures and the Principles of Protein Conformation (Fasman, G.D., ed.), pp. 549–586. Plenum Press, New York.

Chou, K.C. and Zhang, C.T. 1995. Prediction of protein structural classes. *Crit. Rev. Biochem. Mol. Biol.* 30, 275–349.

Chou, K.C., Liu, W.M., Maggiora, G.M. and Zhang, C.T. 1998. Prediction and classification of domain structural classes. *Proteins: Struct. Funct. Genet.* 31, 97–103.

desJardins, M., Karp, P.D, Krummenacker, M., Lee, T.J., and Ouzonis, C.A.1997. Prediction of enzyme classification from protein sequence without the use of sequence similarity, *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Halkidiki, Greece.

Dobson, P.D., and Doig, A.J. 2003. Distinguishing Enzyme Structures from Non-Enzymes Without Alignments. *J. Mol. Biol*. 330, 771-783.

Eisenberg, D., Marcotte, C.A., Xenarios, I., and Yeates, T.O. 2000. Protein function in the post-genomic era. *Nature* 405, 823 – 826.

Jensen, L.J., Skovgaard, M., and Brunak, S. 2002. Prediction of novel archaeal enzymes from sequence-derived features. *Protein Sci.* 11, 2894-2898.

King, R.D, Paul, H., and Clare, A. 2004. Confirmation of data mining based predictions of protein function. *Bioinformatics* 20, 1110-1118.

King, R.D., Karwath, A, Clare, A., and Dehaspe, L 2000. Acurate prediction of protein functional class from sequence in the *M. tuberculosis* and *E. coli* genomes using data mining. *Yeast-Comparative and Functional Genomics* 17(4), 283 – 293.

Klein, P. 1986. Prediction of protein structural class bydi scriminant analysis. *Biochem. Biophys. Acta.* 874, 205–215.

Nagl, S. 2003. Function prediction from protein sequence. In Orengo, C.A., Jones, D.T. Thornton, J.M. (eds). *Bioinformatics - Genes, proteins and computers.* BIOS Scientific publishers. Oxford. 298 pp.

Nishikawa, K. and Ooi, T. 1982. Correlation of amino acid composition of a protein to its tructural and biological characters. *J. Biochem.* 91, 1821–1824.

Nishikawa, K., Kubota, Y. and Ooi, T. 1983. Classification of proteins into groups based on amino acid composition and other characters. I. Angular distribution. *J. Biochem.* 94, 981–995.

Nishikawa, K., Kubota, Y. and Ooi, T. 1983. Classification of the proteins into groups based on amino acid composition and other characters. II. Grouping into four types. *J. Biochem.* 94, 997–1007.

Nakashima, H., Nishikawa, K. and Ooi, T. 1986. The folding type of a protein is relevant to the amino acid composition. *J. Biochem.* 99, 152–162.

Pasquier, C., Promponas, V., and Hamodrakas, S.J. 2001. PRED-CLASS: Cascading Neural networks for generalized protein classification and genome wide applications. *Proteins* 44, 361-369.

Pellegrini, M., Marcotte, E., Thompson, M., Eisenberg, D., and Yeates, T. 1999. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc. Natl. Acad. Sci.* 38, 667–677.

Rice, P., Longden, I., and Bleasby, A. 2000. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics* 16, (6) pp276—277.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., and Schomburg, D. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Res.* Jan 1, 32(Database issue):D43, 1-3.

Tian, W., and Skolnick, J. 2003. How well is enzyme function conserved as a function of pairwise sequence identity?. *J. Mol. Biol.* 333, 863-882.

Wu, C., Berry, M., Shivakumar, S., and McLarty, J. 1995. Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with singular Value Decomposition. *J Mach. Learn.* 21 N(1-2), 177-193.

Zell, A., and Mamier, G. 1997. Stuggart Neural Network Simulator version 4.2. Universty of Stuttgart, Stuttgart, Germany.

Table.1. 85 'Pepstat(EMBOSS)' primary sequence descriptors used in the Study.

Sequence derived parameters	DNA bindnig		Non-DNA binding		Sequence derived parameters	DNA binding		Non-DNA binding	
	Max	Min	Max	Min		Max	Min	Max	Min
Charge	0.207588	0.00182	0.20947	0.00419	N_Mole %	0.7186	0.1200	0.9091	0.2300
Isoelectric Point	0.11811	0.09159	0.1209	0.09186	N_DayhoffStat	0.1671	0.0987	0.2114	0.1078
A_Mole %	0.104656	0.0427	0.1288	0.03857	P_Mole %	0.9572	0.3450	3.6556	0.5680
A_DayhoffStat	0.29032	0.019	0.33257	0.027	P_DayhoffStat	0.1841	0.0089	0.703	0.02908
B_Mole %	0.275	0.024	0.376	0.036	Q_Mole %	0.585	0.0871	1.5106	0.1098
B_DayhoffStat	0.928	0.494	0.979	0.41	Q_DayhoffStat	0.15	0.0098	0.3873	0.0129
C_Mole %	0.18828	0.02881	0.21186	0.03	R_Mole %	1.0682	0.0088	2.1256	0.0187
C_DayhoffStat	0.2189	0.0335	0.2464	0.045	R_DayhoffStat	0.218	0.02389	0.434	0.0452
D_Mole %	0.1989	0.0017	0.0902	0.0011	S_Mole %	0.9035	0.1796	2.2034	0.0012
D_DayhoffStat	0.0292	0.001	0.0109	0.0009	S_DayhoffStat	0.1291	0.0257	0.3148	0.0389
E_Mole %	1	0.00659	2.0339	0.0089	T_Mole %	1.0497	0.3091	1.4352	0.1203
E_DayhoffStat	0.3448	0.02154	0.7013	0.0154	T_DayhoffStat	0.1721	0.0507	0.2353	0.0092
F_Mole %	0.8147	0.0154	1.206	0.0015	V_Mole %	0.15	0.04484	0.17647	0.0289
F_DayhoffStat	0.1481	0.0152	0.2193	0.0652	V_DayhoffStat	0.2273	0.0679	0.2674	0.0546
G_Mole %	1.018	0.0147	1.8615	0.0254	W_Mole %	0.4598	0.00245	0.4839	0.0254
G_DayhoffStat	0.1697	0.0215	0.3102	0.0145					
H_Mole %	0.9195	0.1277	1.0044	0.0596	W_DayhoffStat	0.3537	0.0021	0.3722	0.0215
H_DayhoffStat	0.2554	0.0355	0.279	0.0101	X_Mole %	0.4562	0.025	0.3262	0.0254
I_Mole %	0.25	0.00769	0.36923	0.00503					
I_DayhoffStat	0.2976	0.0092	0.4396	0.006	X_DayhoffStat	0.5263	0.0562	0.3215	0.025
K_Mole %	0.6513	0.00894	1.0271	0.021	Y_Mole %	0.6135	0.0159	2.4615	0.0521
K_DayhoffStat	0.3257	0.0456	0.5136	0.0598					
L_Mole %	1	0.2077	1.0377	0.0089	Y_DayhoffStat	0.1804	0.0154	0.724	0.00987
L_DayhoffStat	0.2222	0.0462	0.2306	0.0564	Z_Mole %	0.2222	0.0089	0.3262	0.0154
M_Mole %	1.018	0.0591	2.0455	0.00115					
M_DayhoffStat	0.1542	0.00213	0.3099	0.0002	Z_DayhoffStat	0.894	0.1256	0.265	0.03652
Charged Mole %	0.19444	0.03139	0.19101	0.0321	Tiny Mole %	0.6	0.15569	0.6389	0.16239
Basic Mole %	0.2628	0.0424	0.2581	0.0021	Small Mole %	0.75	0.4012	0.77119	0.32479
Acidic Mole %	0.5169	0.0456	1.2346	0.0268	Aliphatic Mole %	0.31481	0.14808	0.32903	0.02542
Aromatic Mole %	0.24521	0.04918	0.29231	0.08541	Acidic	0.1159	0.02569	0.13489	0.03654
Non-polar Mole %	0.85	0.45521	0.86154	0.31818	Basic	0.1869	0.04521	0.12365	0.02564
Polar Mole %	0.54479	0.15	0.68182	0.13846	Charged	0.30125	0.00586	0.25634	0.01245
Aromatic	0.10256	0.04956	0.1558	0.07852	Aliphatic	0.25692	0.08956	0.1667	0.00698

* The parameters are scaled down by appropriate scaling values.

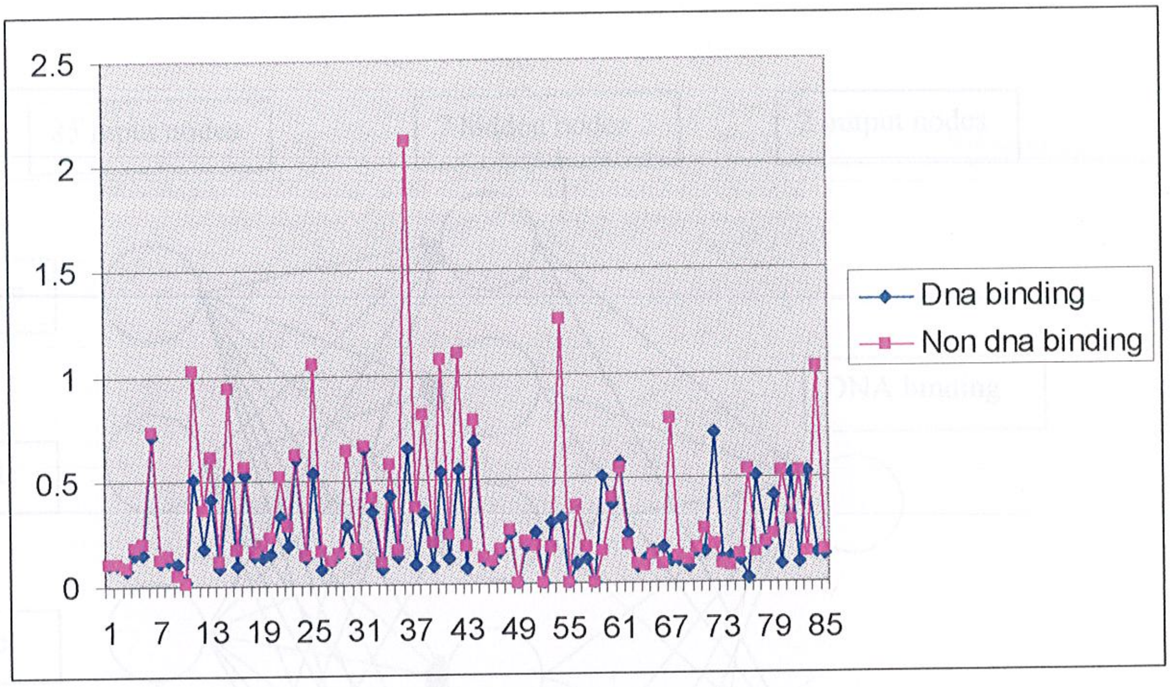


Figure 8. Averaged values of 85 sequence derived parameters calculated independently within studied sets of dna binding and non dna biding.

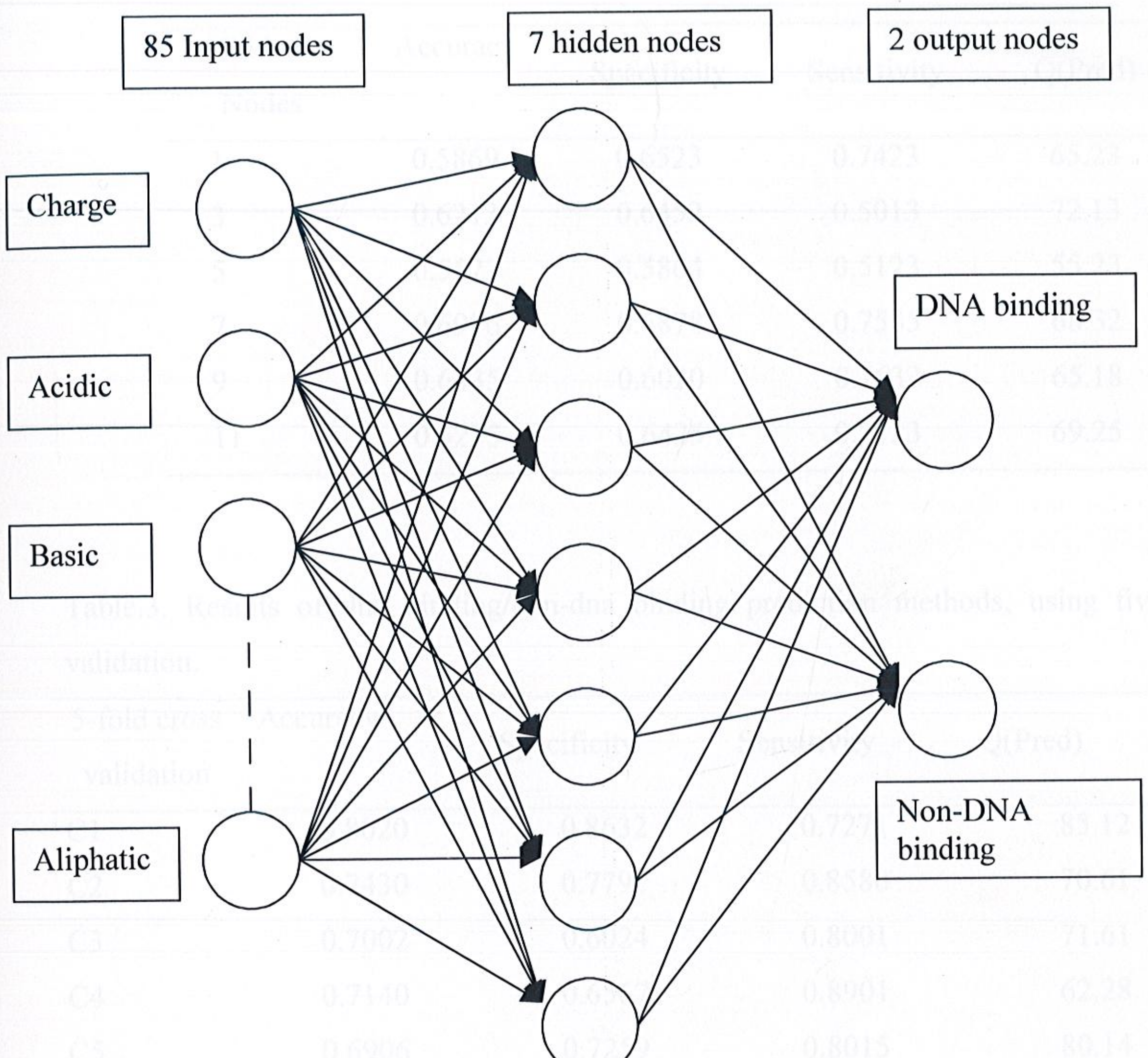


Figure 9. Configuration of artificial neural network used to develop binary primary sequence descriptor model

Table 2. Parameters of specificity, sensitivity, accuracy and positive predictive values for prediction of dna binding and non-dna binding proteins by the artificial neural network with the varying number of hidden nodes.

Hidden Nodes	Accuracy	Specificity	Sensitivity	Q(Pred)
1	0.5869	0.6523	0.7423	65.23
3	0.6213	0.6452	0.5013	72.13
5	0.5522	0.5864	0.5123	55.23
7	0.6976	0.6878	0.7535	68.32
9	0.6435	0.6020	0.7632	65.18
11	0.6235	0.6425	0.7123	69.25

Table.3. Results of dna binding/non-dna binding prediction methods, using five fold cross validation.

5-fold cross validation	Accuracy	Specificity	Sensitivity	Q(Pred)
C1	0.8020	0.8632	0.7271	85.12
C2	0.7430	0.7791	0.8580	70.61
C3	0.7002	0.6024	0.8001	71.61
C4	0.7140	0.6567	0.8901	62.28
C5	0.6906	0.7259	0.8015	80.14
Mean	0.7299 ± 0.0686	0.7254 ± 0.0639	0.8153 ± 0.0673	73.952 ± 13.123

Table 4. Output values from the neural network for the fivefold cross validation set's of

Testing 5 fold cross validation	Number of dna binding proteins correctly predicted (out of 100)	Number of non-dna binding proteins correctly predicted (out of 100)	Prediction range (dna binding)	Prediction range (non-dna binding)
Using sequence features				
C1	67	52	0.6726-1.00	0.00-0.5240
C2	50	56	0.5079-1.00	0.00-0.5658
C3	42	53	0.4257-1.00	0.00-0.5386
C4	35	64	0.3592-1.00	0.00-0.6486
C5	47	58	0.4748-1.00	0.00-0.5836

Table .5. The data set size and breakdown on organisms.

Organism	Annoted protein sequences	Assigned as DNA binding (using derived sequence features)	Assigned as DNA binding (using HMM model)
E.coli	250	185	162
Herpes virus	250	192	173
Total	500	377	335

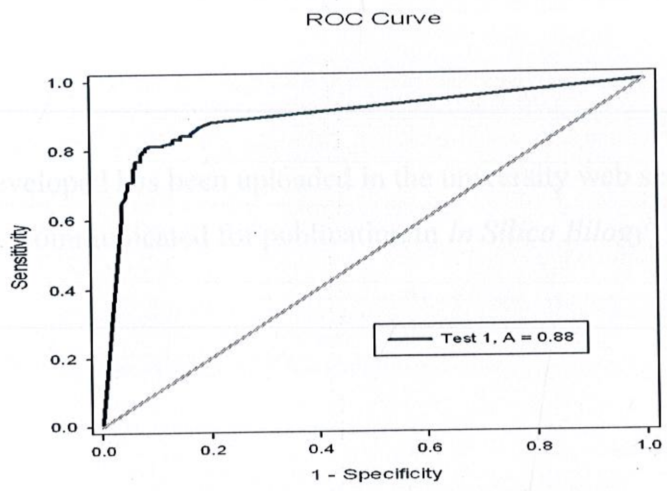


Figure 10. ROC curves for binary ANN network systems.

Chapter 3

DBPPred: A Tool for prediction and classification of DNA Bindng proteins into four major classes using ANN from sequence derived features.

(The tool developed has been uploaded in the university web server and it is communicated for publication in *In Silico Bilogy*)

Abstract

Classes of newly found DNA binding protein sequences are usually determined either by biochemical analysis of eukaryotic and prokaryotic genomes or by microarray chips. These experimental methods are both time-consuming and costly. With the explosion of protein sequences entering into databanks, it is highly desirable to explore the feasibility of selectively classifying newly found Dna binding protein sequences into their respective classes by means of an automated method. This is indeed important because knowing which family or subfamily a protein belongs to may help deduce its catalytic mechanism and specificity, giving clues to the relevant biological function. In this study, we have developed a prediction method for detection and classification of DNA binding proteins from sequence alone The method does not make use of sequence similarity; rather, it relies on predicted protein features and simple physical/chemical properties. The tool has been validated in two different organisms and is proved to be very useful in classification of DNA binding proteins with good accuracy.

Introduction

A large number of data are constantly being generated thanks to several genome-sequencing projects throughout the world. However, the gap between the growth rate of biological sequences and the capability to characterize experimentally the roles and functions associated with these new sequences is constantly increasing [1]. This results in an accumulation of raw data that can lead to an increase in our biological knowledge only if computational characterization tools are developed. **DNA-binding proteins** are proteins that comprise any of many DNA-binding domains and thus have a specific or general affinity to DNA.DNA-binding proteins include transcription factors which modulate the process of transcription, nucleases which cleave DNA molecules, and histones which are involved in DNA packaging in the cell nucleus.We focus here on the annotation of protein as DNA binding/non-DNa binding and if it is DNA binding its classification into four major classes.

A generic approach to this problem consists of transferring the annotation from sequences of known DNA binding proteins to uncharacterized proteins [2]. The transfer mechanism might be subdivided in two steps: (i) to establish the list of known DNA binding proteins with significant sequence similarity to the uncharacterized sequence; (ii) to select the known sequence(s) from which the annotation is transferred [3]. The first step is usually performed with sequence alignment tools such as FASTA [4] or BLAST [5]. When sensitivity is critical, alternative tools such as PSI-BLAST [6] and hidden Markov models [7] can be used. Finding homologous proteins can also be accomplished using alignment independent sequence comparison tools, which have been developed to overcome the limitation arising from the assumption of contiguity between homologous segments [8,9]. Then, the challenge is the selection of true homologues from the list of similar sequences. Most of the above tools provide a score measuring the degree of similarity between the sequences compared. A simple criterion to single out a homologue is to choose the most similar sequence i.e. the highest scoring sequence. More elaborate methods have been designed to enhance the precision and reliability of the annotation process. These rely on the combination of the annotations of more than one homologue [10-13] .

However, annotating and assigning as dna binding/non-dna binding and their further classification into four major classes from their primary sequences requires highly automated computational methods linking experimental data. These methods must be able to discriminate the distinct catalytic function encapsulated in the protein's structure or in its primary sequences. To this end, the machine learning methods (MLMs) seem to be best suited for the task. Compared to the similarity-based methods such as BLAST or FASTA (Altschul *et al.*, 1990) and phylogeny-based method such as ClustalW, MLMs are widely applicable, and now frequently used in annotation of biological sequence analysis with relatively good accuracy. MLMs also have a certain degree of flexibility regarding data inputs, allowing them to expand progressively to meet the requirements of rapidly accumulating mountain of data generated from genomics research. The most often used methods of MLMs are support vector machine (SVM), neural network (NN), hidden markov model (HMM), decision tree (DT) and so on. Among these, NNs are particularly attractive due to its ability for pattern recognition (Raghava 2004), to handle large or small datasets, large input spaces (Narayanan et al. 2002), and its greater accuracy compared to simple BLAST or HMM methods (Bhasin and Raghava, 2004a; 2004b). Currently, there is no reliable systematic way for recognizing and classifying enzymes. Jensen et al. (2006) reported a method which classifies the enzymes into six major classes according to their sequence derived features, that is the co translational and posttranslational modifications, secondary structure, and simple physical/chemical properties. The limitation of this method is that it is confined only to archeal and they have developed six neural networks, each one for each of the six major classes of enzymes. Other methods make use of the structural features to classify enzymes into six major classes (Chou and Elrod 2003). The natural encoding of the primary structure is a string of letters. However, this encoding is not appropriate for NNs, since it demands numerical (preferably, normalized) inputs. Therefore, proteins have to be encoded in a more suitable way. Proteins-including enzymes, in general, are composed by a variable number of amino acids, from tens to thousands. The encoding process proposed here allows differently sized enzymes to be processed by a predefined, fixed-size NN based on fixed number of sequence derived features from the protein sequence. The method does not make use of sequence similarity. Strategically, we have develop a neural network, two-layer, fully automated computational method capable of recognizing enzymes first, and then classifying them into their subfamilies based on their protein sequences. A user-friendly program EnzymePred2 has been developed on the basis of this study to assist readers to distinguish enzymes and to annotate their subfamilies.

MATERIALS AND METHODS

Dataset for prediction of DNA binding/non-DNA binding

The sequence data on positive examples of dna biniding proteins used were obtained from the PDB database (Schomburg *et al.*, 2004). containing 400 protein sequences assigned to four classes according to their structural features. These sequences were used as positive examples for prediction as enzymes. The sequences data on negative examples were obtained from the PDB database .Sequences related to enzymes were removed from the original dataset. A non-redundant treatment was applied (same as for positive datasets) such that no sequence had similarity higher than 25% to any others. Thus, 400 non-dna binding sequences were optimized as negative examples.

Dataset for classification of enzymes into four major classes

The above mentioned 400 protein sequences of DNA binding protein were then grouped into four major classes Class 1 (Homeobox domain) consist of 100 sequences, Class II (Zinc) consist of 100 sequences, Class III (Leucine zipper) having 100 sequences, class IV (Helix-Turn-Helix) with 100 sequences, class V (None) consist of 400 sequences. They were used for construction of neural networks training and validating the model for classification of novel enzymes into four classes.

Neural network architecture

In this study we have used the standard back-propagation ANN configuration consisting of 85 inputs and 5 output nodes with one hidden layer.First layer of neural network is used for prediction of DNA binding/non-DNA binding from the protein sequence, whereas, the second layer is used for classifying the predicted dna binding protein into out of four major classes. The architecture of 1st neural network consisting of 85 inputs, 7 hidden nodes and 1 output node, whereas the 2nd neural network consisting of 85 inputs, 11 hidden nodes and five output nodes (each node is specified for each class of enzyme) (Figure 3.a). For each sequence in the training and testing sets, we have transformed 85 network input parameters into the normalized values varying from 0 to 1. Similarly, the output parameters from the ANN were in the range of 0 to 1. During the learning phase, a value of 1 was assigned for the DNA binding sequence and 0 for non-DNA binding sequence. For configuration of the ANN, 100 independent training runs were performed to evaluate the average predictive power of the network. The corresponding counts of the false/true positive and negative predictions were estimated using 0.1 and 0.9 cut-off values for non-dna binding and dna binding respectively. Thus, an enzyme from the testing set was considered correctly predicted by the ANN only when its output value ranged from 0.9 to 1.0. For each non-dna binding of the testing set the correct prediction was assumed if the corresponding ANN output lies between 0 and 0.1. Thus, all network output values ranging from 0.2 to 0.9 have been ultimately considered as incorrect predictions (rather than undetermined or non-defined). For classifying the predicted dna binding into one.

Sequence derived parameters calculation

To build a binary ANN model enabling effective prediction of dna binding/non-dna binding we initially calculated 85 parameters (Table 6 and 7) from the protein sequence alone using PEPSTAT (EMBOSS suite) ftp://emboss.open-bio.org/pub/EMBOSS (Rice et al., 2000) for all 800 protein sequences. The average values of these 85 parameters independently calculated for dna binding and non- dna binding have been plotted onto Figure 3.a. Similarly all the predicted 85 parameters for each class of dna binding proteins independently is given in Table 6 and 7 & the average value of the parameters is given in Figure 3.a. It showed clear distinction between dna binding and non-dna binding based on 85 parameters. The normalized values have been then used to generate ANN models for binary prediction.

The input to second filtering network is the same input values used for the first layer and the predicted enzyme is classify into its particular class based on the maximum value obtained from the defined out put node for each class. For example to classify the predicted dna binding into class 1 the predicted output value is 1, 0, 0, 0, 0 and so on.

Fivefold cross-validation

A prediction method is often developed by cross-validation or jack-knife method (Chou and Zhang, 1995). Because of the size of the dataset, the jack-knife method (individual testing of each enzyme in the data set) was not feasible. So a more limited cross-validation technique has been used, in which the dataset is randomly divided into five subsets, each containing equal number of enzyme sequences. Each set is a balanced set that consist of 50 percent of dna binding and 50 percent non-dna binding. The data set has been divided into training and testing set. The training set consists of five subsets. The network is validated for minimum error on testing set to calculate the performance measure for each fold of validation. This has been done five times to test for each subset. The final prediction results have been averaged over five testing sets.

RESULTS

Predictability of DNA binding proteins and their classification into four classes

The ANN model develop in this study (85-7-1) is trained with the sequence derived features (85 parameters) calculated using PEPSTAT. When applying a fivefold cross-validation test using five data sets, we found that the network reached an overall accuracy of 72.19 ± 6.86 %. The prediction results are presented in Table .8.. The other performance measures are: Qpred = 69.466 ± 17.084 %, sensitivity = 83.70 ± 6.73 % and specificity = 74.56 ± 13.39 %. The value of the learning parameter was set to 0.1. Training was performed for 100 epochs for both the networks, after which the learning has been terminated when the error reached a stable value; differences between errors in subsequent steps become sufficiently small. Table 9 revealed the predictability of enzymes and non-enzymes of the network. Out of 100 dna binding proteins in each cross validation set 40-70 dna binding were correctly predicted as enzymes. However, out of 100 in non-dna binding class; 35-62 were correctly predicted as non-non-dna binding. Prediction performance measures were averaged over five sets. Figure 3.a features averaged frequencies of the output values for the five testing sets used in the study. As it can readily be seen from the graph, the vast majority of the predictions has been contained within (0.0 – 0.1) for non-dna binding and (0.9 – 1.0) for dna binding in case of sequence derived module. This illustrates that 0.1 and 0.9 cut-offs values provide very adequate separation of two bioactive classes using ANN. All network output values ranging from 0.1 to 0.9 have been ultimately considered as incorrect predictions (rather than undetermined or non-defined).

An application of the model

The reliability of developed model of binary prediction of dna binding and non-dna binding and their classification into four major classes was tested on the annotated protein sequences of two organisms downloaded form GenBank (Benson et al. 2002). The predicted results are shown in Table 10. Prediction on new sequences is done by first running the PEPSTAT (EMBOSS) program to obtain the sequence-derived features, which are subsequently used as input for the prediction of dna binding/non-dna binding and the same values are again used for classification of the dna binding proteins if it is predicted as dna binding protein from first layer.

4. DISCUSSION

The tool EnzymePred2 has been developed in this study using two layered neural network based on sequence derived features. The results demonstrate that the developed ANN-based model for binary prediction of enzymes/non-enzymes and classification of enzymes into six major classes is adequate and can be considered an effective tool for 'in silico' screening. The results also demonstrate that the sequence derived parameters readily accessible from the protein sequences only, can produce a variety of useful information to be used '*in silico*'; clearly demonstrates an adequacy and good predictive power of the developed ANN model. There is strong evidence, that the introduced sequence features do adequately reflect the structural properties of proteins. The structure of a protein is an important determinant for the detailed molecular function of proteins, and would consequently also be useful for prediction of enzymes/non-enzymes and for their classification. Based on the analysis of limited sequence features from protein sequences, differences in the parameters between enzymes and non-enzymes have previously been shown to exist and used for prediction of enzymes/non-enzymes in archaeal (Jensen et al., 2002). This agrees well with our result that sequence derived features can be used for predicting enzymes. This observation is not surprising considering that the calculated parameters should cover a very broad range of proprieties of bound atoms and molecules related to their size, polarizability, electronegativity, compactness, mutual inductive and steric influence and distribution of electronic density, etc. As it can be seen that the average value for both the classes were clearly separated on the graph and, hence, the selected 61 parameters should allow building an effective ANN model for binary prediction (Fig.11). The average value of all the 61 sequence derived parameters used in the study for all the six classes of enzymes clearly separated and could be suitable for classification.

The ANN with 61 input-nodes, 4 hidden-nodes and 1 output nodes has allowed the recognition of 79 % of enzymes and 79 % of non-enzymes, on average (Table 8) and has also demonstrated very good separation on positive (enzymes) and negative (non-enzymes) predictions. The second layer of neural network with 61 input-,32 hidden and 6 output nodes able to correctly classify 67 % of the enzymes (Table 10) This result revealed a good prediction with accuracy of > 65 %. Further, the reliability of developed ANN model for prediction of enzyme and non-enzyme and their further classification were tested on the complete annotated protein sequences of three organisms downloaded form GenBank (Benson et al., 2002). The predicted results was shown in Table 12

Presumably, accuracy of the approach operating by the sequence derived features can be improved even further by expanding the parameters or by applying more powerful classification techniques such as Support Vector Machines or Bayesian Neural Networks. Use of merely statistical techniques in conjunction with the sequence parameters would also be beneficial, as they will allow interpreting individual parameter contributions into "enzymes/non-enzymes-likeness".

The results of the present work demonstrate that the sequence derived features with ANN appear to be a very fast protein classification mechanism providing good results, comparable to some of the current efforts in the literature. The developed ANN-based model for enzymes/non-enzymes prediction and their classification into different classes can be used as a powerful tool for filtering through the collections of genome sequences to discover novel enzymes.

References

1. Janssen P, Audit B, Cases I, Darzentas N, Goldovsky L, Kunin V,Lopez-Bigas N, Peregrin-Alvarez JM, Pereira-Leal JB, Tsoka S,Ouzounis CA: Beyond 100 genomes. *Genome Biol* 2003, 4:402.

2. Andrade MA, Sander C: Bioinformatics: from genome data tobiological knowledge. *Curr Opin Biotechnol* 1997, 8:675-683.

3. Karp PD: What we do not know about sequence analysis and sequence databases. *Bioinformatics* 1998, 14:753-754.4. Pearson WR: Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol* 1990, 183:63-98.

4. Shah I, Hunter L: Predicting enzyme function from sequence: a systematic appraisal. *Proc Int Conf Intell Syst Mol Biol* 1997, 5:276-283.

5. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: Basic local alignment search tool. *J Mol Biol* 1990, 215:403-410.

6. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997, 25:3389-3402.

7. Krogh A, Brown M, Mian IS, Sjolander K, Haussler D: Hidden Markov models in computational biology. Applications to protein modeling. *J Mol Biol* 1994, 235:1501-1531.

8. Vinga S, Almeida J: Alignment-free sequence comparison-a review. *Bioinformatics* 2003, 19:513-523.

9. Vries JK, Munshi R, Tobi D, Klein-Seetharaman J, Benos PV, Bahar I: A sequence alignment-independent method for protein classification. *Appl Bioinformatics* 2004, 3:137-148.

10. Abascal F, Valencia A: Automatic annotation of protein function based on family identification. *Proteins* 2003, 53:683-692.

11. Krebs WG, Bourne PE: Statistically rigorous automated protein annotation. *Bioinformatics* 2004, 20:1066-1073.

12. Leontovich AM, Brodsky LI, Drachev VA, Nikolaev VK: Adaptive algorithm of automated annotation. *Bioinformatics* 2002, 18:838-844.

13. Tatusov RL, Galperin MY, Natale DA, Koonin EV: The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res* 2000, 28:33-36.

14. Andrade MA, Brown NP, Leroy C, Hoersch S, de Daruvar A, Reich C, Franchini A, Tamames J, Valencia A, Ouzounis C, Sander C: Auto- mated genome sequence analysis and annotation. *Bioinformatics* 1999, 15:391-412.

15. Wilson CA, Kreychman J, Gerstein M: Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *J Mol Biol* 2000, 297:233-249.

16. Kyrpides NC, Ouzounis CA: Whole-genome sequence annotation: 'Going wrong with confidence'. *Mol Microbiol* 1999, 32:886-887.

17. Bork P, Koonin EV: Predicting functions from protein sequences--where are the bottlenecks? *Nat Genet* 1998, 18:313-318.

18. Devos D, Valencia A: Intrinsic errors in genome annotation. *Trends Genet* 2001, 17:429-431.

19. Gerlt JA, Babbitt PC: Can sequence determine function? *Genome Biol* 2000, 1:REVIEWS0005.

20. Gilks WR, Audit B, De Angelis D, Tsoka S, Ouzounis CA: Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics* 2002, 18:1641-1649.

21. Cheng BY, Carbonell JG, Klein-Seetharaman J: Protein classification based on text document classification techniques. *Proteins* 2005, 58:955-970.

22. des Jardins M, Karp PD, Krummenacker M, Lee TJ, Ouzounis CA: Prediction of enzyme classification from protein sequence without the use of sequence similarity. *Proc Int Conf Intell Syst Mol Biol* 1997, 5:92-99.

23. Karchin R, Karplus K, Haussler D: Classifying G-protein coupled receptors with support vector machines. *Bioinformatics* 2002, 18:147-159.

24. Fillinger S, Boschi-Muller S, Azza S, Dervyn E, Branlant G, Aymerich S: Two glyceraldehyde-3-phosphate dehydrogenases with opposite physiological roles in a nonphotosynthetic bacterium. *J Biol Chem* 2000, 275:14031-14037.

25. Sigrist CJ, Cerutti L, Hulo N, Gattiker A, Falquet L, Pagni M, Bairoch A, Bucher P: PROSITE: a documented database using patterns and profiles as motif descriptors. *Brief Bioinform* 2002, 3:265-274.

26. Wen Z, Morrison M: The NAD(P)H-dependent glutamate dehydrogenase activities of Prevotella ruminicola B(1)4 can be attributed to one enzyme (GdhA), and gdhA expression is regulated in response to the nitrogen source available for growth. *Appl Environ Microbiol* 1996, 62:3826-3833.

27. Itkor P, Tsukagoshi N, Udaka S: Nucleotide sequence of the rawstarch- digesting amylase gene from Bacillus sp. B1018 and its strong homology to the cyclodextrin glucanotransferase genes. *Biochem Biophys Res Commun* 1990, 166:630-636.

28. Shah I, Hunter L: Predicting enzyme function from sequence: a systematic appraisal. *Proc Int Conf Intell Syst Mol Biol* 1997, 5:276-283.

29. Devos D, Valencia A: Practical limits of function prediction. *Proteins* 2000, 41:98-107.

30. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel- Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G: Gene ontology: tool for the unification of biology. *Nat Genet* 2000, 25:25-29.

31. Gattiker A, Michoud K, Rivoire C, Auchincloss AH, Coudert E, Lima T, Kersey P, Pagni M, Sigrist CJ, Lachaize C, Veuthey AL, Gasteiger E, Bairoch A: Automated annotation of microbial proteomes in SWISS-PROT. *Comput Biol Chem* 2003, 27:49-58.

32. Wieser D, Kretschmann E, Apweiler R: Filtering erroneous protein annotation. *Bioinformatics* 2004, 20 Suppl 1:I342-I347.

33. Murzin AG, Brenner SE, Hubbard T, Chothia C: SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol* 1995, 247:536-540.

34. Holm L, Sander C: Mapping the protein universe. *Science* 1996, 273:595-603.

35. Jaakkola T, Diekhans M, Haussler D: A discriminative framework for detecting remote protein homologies. *J Comput Biol* 2000, 7:95-114.

36. Bairoch A: The ENZYME database in 2000. *Nucleic Acids Res* 2000, 28:304-305.

37. Boeckmann B, Bairoch A, Apweiler R, Blatter MC, Estreicher A, Gasteiger E, Martin MJ, Michoud K, O'Donovan C, Phan I, Pilbout S, Schneider M: The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res* 2003, 31:365-370.

38. Promponas VJ, Enright AJ, Tsoka S, Kreil DP, Leroy C, Hamodrakas S, Sander C, Ouzounis CA: CAST: an iterative algorithm for the complexity analysis of sequence tracts. Complexity analysis of sequence tracts. *Bioinformatics* 2000, 16:915-922.

Table 7. 85 'Pepstat(EMBOSS)' primary sequence descriptors used in the Study for class 1,2 and 3

Parameter	Class 1		Class 2		Class 3	
	Max	Min	Max	Min	Max	Min
Molecular Weight	0.207589	0.001823	0.176616	0.01511	0.207012	0.006066
Average Residue	0.118111	0.091159	0.11967	0.100633	0.116209	0.103872
Isoelectric Point	0.104656	0.0427	0.109987	0.045698	0.105404	0.040801
Extinction Coefficient	0.29032	0.0162	0.18136	0.00128	0.22801	0.00256
Extinction Coefficient (1 mg/ml)	0.275	0.0165	0.225	0.006	0.24	0.014
Improablity / Proability inclusion bodies	0.928	0.494	0.881	0.497	0.838	0.5
A_Mole %	0.18828	0.02881	0.24116	0.02712	0.18182	0.01887
A_DayhoffStat	0.2189	0.0335	0.2804	0.0315	0.2114	0.0219
B_Mole %	0.3125	0.0268	0.1642	0.0343	0.9195	0.0335
B_DayhoffStat	0.165	0.0095	1.1321	0.2642	0.2554	0.0268
C_Mole %	1	0.0921	0.3448	0.01254	0.3774	0.0095
C_DayhoffStat	0.3448	0.0215	0.1189	0.00145	0.1301	0.0921
D_Mole %	0.8147	0.0621	0.7353	0.197	0.894	0.1887
D_DayhoffStat	0.1481	0.0254	0.1337	0.0358	0.1626	0.0343
E_Mole %	1.018	0.01254	1.3095	0.3167	1.3208	0.2642
E_DayhoffStat	0.1697	0.00145	0.2183	0.0528	0.2201	0.044
F_Mole %	0.9195	0.1277	0.7251	0.098	0.7419	0.1887
F_DayhoffStat	0.2554	0.0355	0.2014	0.0272	0.2061	0.0524
G_Mole %	0.25	0.00769	0.11189	0.02387	0.11861	0.02273
G_DayhoffStat	0.2976	0.0092	0.1332	0.0284	0.1412	0.0271
H_Mole %	0.6513	0.012	0.4545	0.0489	0.5994	0.0505
H_DayhoffStat	0.3257	0.0654	0.2273	0.0169	0.2997	0.0253
I_Mole %	1	0.2077	1.215	0.1173	1.2453	0.16
I_DayhoffStat	0.2222	0.0462	0.27	0.0261	0.2767	0.0356
K_Mole %	1.018	0.0315	1.0894	0.1095	1.1321	0.04
K_DayhoffStat	0.1542	0.0654	0.1651	0.0166	0.1715	0.0061
L_Mole %	0.19444	0.03139	0.16742	0.03623	0.152	0.0566

L_DayhoffStat	0.2628	0.0424	0.2262	0.049	0.2054	0.0765
M_Mole %	0.5169	0.0154	0.4483	0.0457	0.5415	0.0658
M_DayhoffStat	0.3041	0.0805	0.2637	0.0269	0.3185	0.0387
N_Mole %	0.7186	0.1031	0.6107	0.045	0.6792	0.0649
N_DayhoffStat	0.1671	0.0606	0.142	0.495	0.158	0.0151
P_Mole %	0.9572	0.1401	0.8403	0.1639	1.0239	0.2632
P_DayhoffStat	0.1841	0.0326	0.1616	0.0315	0.1969	0.0506
Q_Mole %	0.585	0.1662	0.8173	0.0683	0.6213	0.1227
Q_DayhoffStat	0.15	0.00175	0.2096	0.1359	0.1593	0.0315
R_Mole %	1.0682	0.0235	1.5249	0.1934	1.0452	0.1707
R_DayhoffStat	0.218	0.0478	0.3112	0.0395	0.2133	0.0348
S_Mole %	0.9035	0.1796	1.25	0.2797	1.1024	0.1342
S_DayhoffStat	0.1291	0.0257	0.1786	0.04	0.1575	0.0192
T_Mole %	1.0497	0.3091	0.8511	0.1173	0.8136	0.2208
T_DayhoffStat	0.1721	0.0507	0.1395	0.0192	0.1334	0.0362
V_Mole %	0.15	0.04484	0.12941	0.03812	0.11321	0.03019
V_DayhoffStat	0.2273	0.0679	0.1961	0.0578	0.1715	0.0457
W_Mole %	0.4598	0.1276	0.3636	0.35088	0.4027	0.34121
W_DayhoffStat	0.3537	0.8673	0.2797	0.18644	0.3098	0.16848
X_Mole %	0.5123	0.1422	0.9572	0.09712	0.85	0.09021
X_DayhoffStat	0.2654	0.10526	0.1841	0.07627	0.54479	0.07065
Y_Mole %	0.6135	0.1595	0.5966	0.0322	0.7143	0.0613
Y_DayhoffStat	0.1804	0.2528	0.1755	0.0095	0.2101	0.018
Z_Mole %	0.1548	0.1945	0.54479	0.0654	0.1291	0.07107
Z_DayhoffStat	1.2306	0.0215	0.33533	0.2077	1.0497	0.51247
Tiny Mole %	0.6	0.15569	0.40836	0.17288	0.37014	0.20134
Small Mole %	0.75	0.4012	0.51896	0.3871	0.6019	0.39245
Aliphatic Mole %	0.31481	0.14808	0.33163	0.1653	0.27925	0.16327
Aromatic Mole %	0.24521	0.04918	0.17488	0.03939	0.17532	0.06089
Non-polar Mole %	0.85	0.45521	0.6881	0.46001	0.66	0.52101
Polar Mole %	0.54479	0.15	0.53999	0.3119	0.47899	0.34
Charged Mole %	0.33533	0.05	0.34409	0.16995	0.33962	0.20601
Basic Mole %	0.17365	0.05	0.20235	0.0836	0.18868	0.08233
Acidic Mole %	0.16168	0	0.17262	0.05665	0.16429	0.04906

Table 8. 85 'Pepstat(EMBOSS)' primary sequence descriptors used in the Study for class 4,5.

Parameter	Class 4		Class 5	
	Max	Min	Max	Min
Molecular Weight	0.067056	0.004217	0.044612	0.038203
Average Residue	0.117138	0.104423	0.115822	0.104421
Isoelectric Point	0.085963	0.045253	0.101241	0.04488
Extinction Coefficient	0.08506	0.00384	0.06657	0.02048
Extinction Coefficient (1 mg/ml)	0.172	0.031	0.172	0.048
Improablity / Proability inclusion bodies	0.871	0.497	0.848	0.503
A_Mole %	0.16102	0.02778	0.1601	0.02151
A_DayhoffStat	0.1872	0.0323	0.1862	0.025
B_Mole %	0.1332	0.098	0.9195	0.01254
B_DayhoffStat	0.4545	0.0272	0.2554	0.00145
C_Mole %	0.4	0.02387	0.2989	0.0489
C_DayhoffStat	0.1379	0.0284	0.1031	0.0169
D_Mole %	0.9032	0.1754	0.8184	0.2792
D_DayhoffStat	0.1642	0.0319	0.1488	0.0508
E_Mole %	1.1321	0.4159	0.9384	0.2989
E_DayhoffStat	0.1887	0.0693	0.1564	0.0498
F_Mole %	0.5556	0.01254	0.5914	0.1662
F_DayhoffStat	0.1543	0.00145	0.1643	0.0462
G_Mole %	0.0995	0.05263	0.11811	0.04986
G_DayhoffStat	0.1185	0.0627	0.1406	0.0594
H_Mole %	0.6349	0.1596	0.6094	0.1344
H_DayhoffStat	0.3175	0.0798	0.3047	0.0672
I_Mole %	1.1475	0.1724	1.3172	0.3073
I_DayhoffStat	0.255	0.0383	0.2927	0.0683
K_Mole %	0.8333	0.08	1.5323	0.1359
K_DayhoffStat	0.1263	0.0121	0.2322	0.0206
L_Mole %	0.13889	0.03448	0.13966	0.05959

L_DayhoffStat	0.1877	0.0466	0.1887	0.0805
M_Mole %	0.5556	0.2477	0.3652	0.1031
M_DayhoffStat	0.3268	0.0688	0.2148	0.0606
N_Mole %	0.8621	0.2581	0.8871	0.1401
N_DayhoffStat	0.2005	0.06	0.2063	0.0326
P_Mole %	0.8621	0.16	0.6793	0.1662
P_DayhoffStat	0.1658	0.0308	0.1306	0.032
Q_Mole %	0.8743	0.01254	0.8549	0.0587
Q_DayhoffStat	0.2242	0.00145	0.2192	0.015
R_Mole %	0.7742	0.2135	0.8683	0.1344
R_DayhoffStat	0.158	0.0436	0.1772	0.0274
S_Mole %	0.9346	0.2477	0.8929	0.3571
S_DayhoffStat	0.1335	0.0688	0.1276	0.051
T_Mole %	0.8743	0.1754	0.8673	0.2241
T_DayhoffStat	0.1433	0.0288	0.1422	0.0367
V_Mole %	0.152	0.04082	0.10526	0.04032
V_DayhoffStat	0.2303	0.0618	0.1595	0.0611
W_Mole %	0.2551	0.0702	0.2528	0.0326
W_DayhoffStat	0.1962	0.2184	0.1945	0.1662
X_Mole %	0.51896	0.0508	0.8531	0.00175
X_DayhoffStat	0.33163	0.1481	0.1984	0.0235
Y_Mole %	1.0345	0.2542	0.544	0.2073
Y_DayhoffStat	0.3043	0.0748	0.16	0.061
Z_Mole %	0.51896	0.3061	0.51896	0.006
Z_DayhoffStat	0.33163	0.0437	0.33163	0.497
Tiny Mole %	0.36441	0.13889	0.3866	0.17473
Small Mole %	0.59658	0.38889	0.59021	0.36828
Aliphatic Mole %	0.30556	0.17241	0.27566	0.20716
Aromatic Mole %	0.16667	0.07627	0.1662	0.07107
Non-polar Mole %	0.64912	0.52459	0.65879	0.51247
Polar Mole %	0.47541	0.35088	0.48753	0.34121
Charged Mole %	0.27778	0.18644	0.32258	0.16848
Basic Mole %	0.14286	0.09712	0.18817	0.09021
Acidic Mole %	0.15094	0.07627	0.14691	0.07065

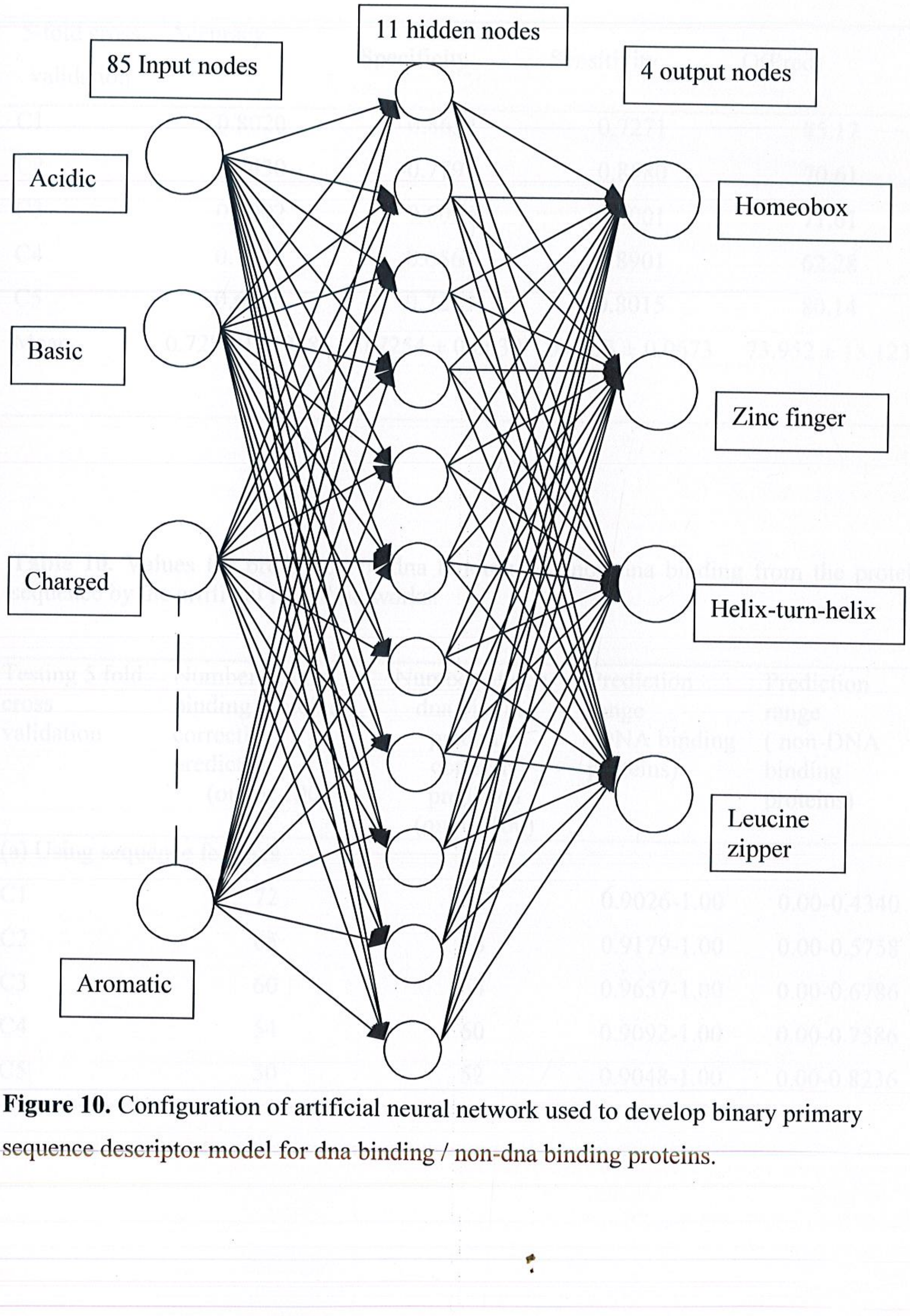


Figure 10. Configuration of artificial neural network used to develop binary primary sequence descriptor model for dna binding / non-dna binding proteins.

Table 9.. Results of dna binding / non-dna binding prediction methods, using five fold cross validation.

5-fold cross validation	Accuracy	Specificity	Sensitivity	Q(Pred)
C1	0.8020	0.8632	0.7271	85.12
C2	0.7430	0.7791	0.8580	70.61
C3	0.7002	0.6024	0.8001	71.61
C4	0.7140	0.6567	0.8901	62.28
C5	0.6906	0.7259	0.8015	80.14
Mean	0.7299 ± 0.0686	0.7254 ± 0.0639	0.8153 ± 0.0673	73.952 ± 13.123

Table 10. Values for prediction of dna binding and non-dna binding from the protein sequence by the artificial neural networks.

Testing 5 fold cross validation	Number of dna binding proteins correctly predicted (out of 100)	Number of non-dna binding preoteins correctly predicted (out of 100)	Prediction range (DNA binding proteins)	Prediction range (non-DNA binding proteins)
(a) Using sequence features				
C1	72	65	0.9026-1.00	0.00-0.4340
C2	65	58	0.9179-1.00	0.00-0.5758
C3	60	54	0.9657-1.00	0.00-0.6786
C4	54	60	0.9092-1.00	0.00-0.7586
C5	50	52	0.9048-1.00	0.00-0.8236

Table 11. Prediction values from the neural network for the fivefold cross validation set's of DNA binding classes.

5-Fold cross validation	Total cases taken	Correctly predicted dna binding proteins
C1	100	72
C2	100	68
C3	100	69
C4	100	65
C5	100	60
Total	500	334

Table 12.. Values for prediction of dna binding classes from the protein sequence by the artificial neural networks with the varying number of hidden nodes.

Number of hidden node	Number dna binding proteins taken	Correctly predicted proteins
1	500	265
3	500	220
5	500	241
7	500	298
11	500	343

Table 13.. Values for prediction of dna binding classes from the protein sequence by the artificial neural networks with the varying number of hidden nodes.

Number of Hidden nodes	Number of DNA binding proteins taken	Assigned as DNA binding(using sequence derived features)	Assigned as class 1 Homeo box	Assigned as class 2 Zinc finger	Assigned as class 3 Leucine zipper	Assigned as class 4 Helix-turn-helix
1	500	265	65	74	61	65
3	500	220	56	61	51	52
5	500	241	55	70	61	65
7	500	298	65	80	75	78
11	500	343	86	87	81	89

Table 13.. The results of DBPPred for the following organisms.

Organism	Annoted protein sequences	Assigned as DNA binding(using sequence derived features)	Assigned as class 1 Homeo box	Assigned as class 2 Zinc finger	Assigned as class 3 Leucine zipper	Assigned as class 4 Helix-turn-helix
E.coli	250	185	55	45	43	42
Herpes virus	250	192	45	60	43	44
Total	500	377	100	105	87	79

4. DISCUSSION

The ANN model developed in this study is based on sequence derived features Dna binding/non-dna binding proteins prediction accuracy has also been assessed and it has been found that preduiction of Dna binding/non-dna binding using sequence derived features is accurate for the cross-valdated sets used for the model.

The results demonstrate that the developed ANN-based binary prediction of dna binding/non-dna binding is adequate and can be considered an effective tool for 'in silico' screening. The results also demonstrate that the sequence derived parameters readily accessible from the protein sequences only, can produce a variety of useful information to be used 'in silico'; clearly demonstrates an adequacy and good predictive power of the developed ANN model. There is strong evidence, that the introduced sequence features do adequately reflect the structural properties of proteins. The structure of a protein is an important determinant for the detailed molecular function of proteins, and would consequently also be useful for prediction of dna binding and non-dna binding proteins. Based on the analysis of limited sequence features from protein sequences, differences in the parameters between dna binding and non-dna binding have previously been shown to exist and used for prediction of dna binding/non-dna binding in archaeal (Jensen et al., 2002). This agrees well with our result that sequence derived features can be used for predicting dna binding proteins. This observation is not surprising considering that the calculated parameters should cover a very broad range of proprieties of bound atoms and molecules related to their size, polarizability, electronegativity, compactness, mutual inductive and steric influence and distribution of electronic density, etc. As it can be seen that the average value for both the classes were clearly separated on the graph and, hence, the selected 85 parameters should allow building an effective ANN model for binary prediction (Fig. 8).

Considering that one of the most important implications for the "binary prediction" model is its potential use for identification of proteins from electronic databases, we have calculated the parameters of the Positive Predictive Values (PPV) for the networks while varying the number of hidden nodes. Taking into account the PPV values for the networks with the varying number of the hidden nodes along with the corresponding values of sensitivity, specificity and general accuracy we have selected neural network with seven hidden nodes as the most efficient among the studied (Table 3). The ANN with 85 input-, 7 hidden- and 2 output nodes has allowed the recognition of 72 % of dna binding and 70 % of non-dna binding proteins, on average (Table.4).

The tool DBPred has been developed in this study using two layered neural network based on sequence derived features. The results demonstrate that the developed ANN-based model for binary prediction of dna binding/non-dnabinding and classification of dna binding proteins into four major classes is adequate and can be considered an effective tool for 'in silico' screening. The results also demonstrate that the sequence derived parameters readily accessible from the protein sequences only, can produce a variety of useful information to be used 'in silico'; clearly demonstrates an adequacy and good predictive power of the developed ANN model. There is strong evidence, that the introduced sequence features do adequately reflect the structural properties of proteins. The structure of a protein is an important determinant for the detailed molecular function of proteins, and would consequently also be useful for prediction of dna binding/non-dna binding proteins and for their classification. Based on the analysis of limited sequence features from protein sequences, differences in the parameters between enzymes and non-enzymes have previously been shown to exist and used for prediction of enzymes/non-enzymes in archaeal (Jensen et al., 2002). This agrees well with our result that sequence derived features can be used for predicting dna binding proteins. This observation is not surprising considering that the calculated parameters should cover a very broad range of proprieties of bound atoms and molecules related to their size, polarizability, electronegativity, compactness, mutual inductive and steric influence and distribution of electronic density, etc. As it can be seen that the average value for both the classes were clearly separated on the graph and, hence, the selected 85 parameters should allow building an effective ANN model for binary prediction. The average value of all the 85 sequence derived parameters used in the study for all the four classes of dna binding proteins clearly separated and could be suitable for classification.

The ANN with 85 input-nodes, 7 hidden-nodes and 2 output nodes has allowed the recognition of 72 % of dna binding and 70 % of non-dna binding proteins, on average (Table 9) and has also demonstrated very good separation on positive (dna binding) and negative (non-dna binding) predictions. The second layer of neural network with 85input-

,11 hidden and 4 output nodes able to correctly classify 72 % of the dna binding (Table 9) This result revealed a good prediction with accuracy of > 69 %. Further, the reliability of developed ANN model for prediction of dna binding and non-dna binding and their further classification were tested on the complete annotated protein sequences of two organisms downloaded form GenBank (Benson et al., 2002). The predicted results was shown in Table 14

Presumably, accuracy of the approach operating by the sequence derived features can be improved even further by expanding the parameters or by applying more powerful classification techniques such as Support Vector Machines or Bayesian Neural Networks. Use of merely statistical techniques in conjunction with the sequence parameters would also be beneficial, as they will allow interpreting individual parameter contributions into "dna binding/non-dna binding-likeness".

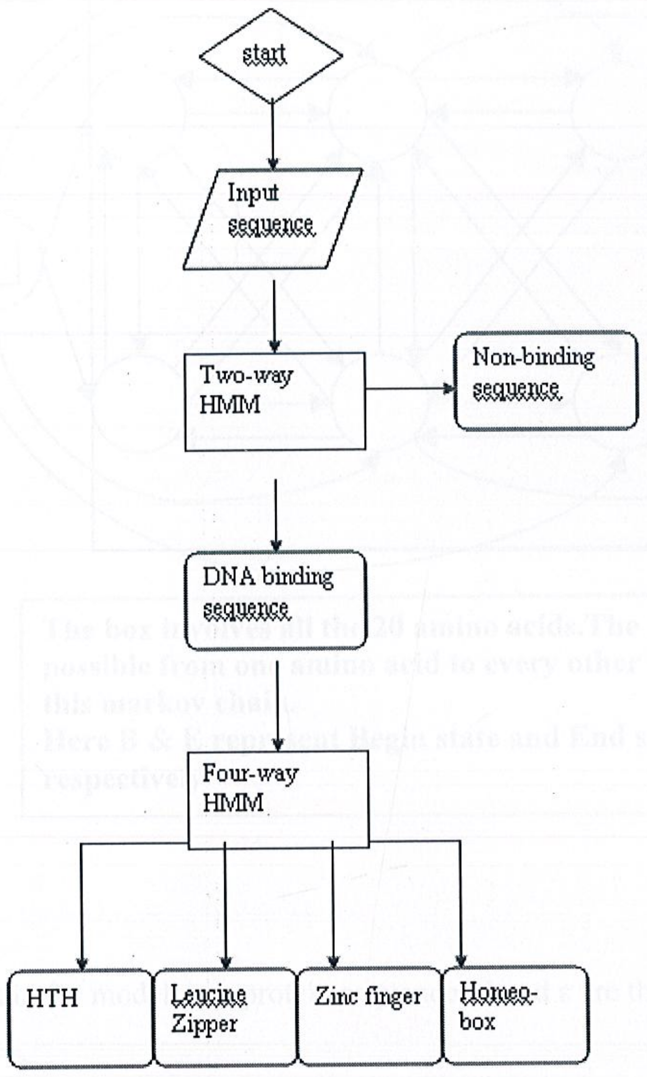
The results of the present work demonstrate that the sequence derived features with ANN appear to be a very fast protein classification mechanism providing good results, comparable to some of the current efforts in the literature. The developed ANN-based model for dna binding/non-dna binding proteins prediction can be used as a powerful tool for filtering through the collections of genome sequences to discover dna binding proteins.

Chapter 4

DBPPred: A tool for prediction and classification of DNA binding proteins into four major classes.

Methods

The steps we have followed for constructing the HMM model are mentioned below.



Markov chains

Definition: A Markov chain is a triplet (Q, {p(x1 = s)}, A), where:

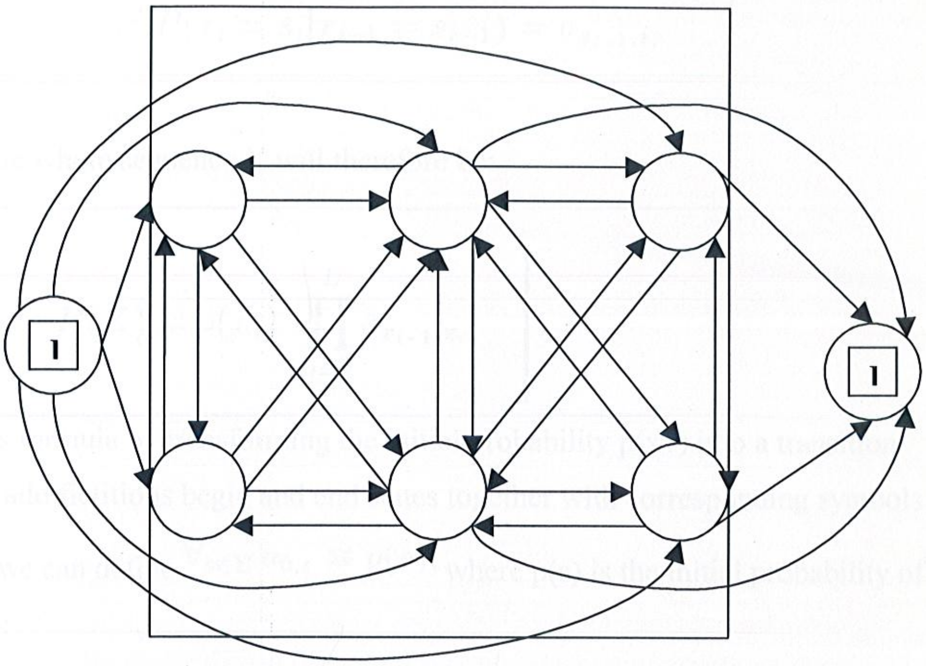
• Q is a finite set of states. Each state corresponds to a symbol in the alphabet Σ.

• p is the initial state probabilities.

• A is the state transition probabilities, denoted by ast for each s, t □ Q.

For each s, t ϵ Q the transition probability is:

$$ast \equiv P(xi = t|xi-1 = s)$$



The box involves all the 20 amino acids.The trasition is possible from one amino acid to every other amino acid in this markov chain.
Here B & E represent Begin state and End steate respectively

Figure 11: A Markov chain for modeling a protein sequence. B and ε are the begin and end states, respectively.

For a protein sequence: we have a Begin state(B), an end state(ε) and in between these, 20 amino acid states ie: ('A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'}

We assume that X = (x1, . . . , xL) is a random process with a memory of length 1, i.e., the value of the random variable xi depends only on its predecessor xi−1. Formally we can write:

$$\forall s_1, \ldots, s_i \in \Sigma \quad P(x_i = s_i | x_1 = s_1, \ldots, x_{i-1} = s_{i-1}) = \\ = P(x_i = s_i | x_{i-1} = s_{i-1}) = a_{s_{i-1}, s_i}$$

The probability of the whole sequence X will therefore be:

$$P(X) = p(x_1) \cdot \prod_{i=2}^{L} a_{x_{i-1}, x_i}$$

We can simplify this formula by transforming the initial probability p(x1) into a transition probability. We can add fictitious begin and end states together with corresponding symbols x0 and xL+1. Then we can define $\forall_{s \in \Sigma}\ a_{0,s} \equiv p(s)$, where p(s) is the initial probability of the symbol s. Hence:

$$P(X) = \prod_{i=1}^{L} a_{x_{i-1}, x_i}$$

Problem 1: Identifying a DNA binding sequence

INPUT: A short amino acid sequence X = (x1, . . . , xL) □ Σ∗
(where Σ = {'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'}).
QUESTION: Decide whether X is a DNA binding sequence.
We can use two Markov chain models to solve this problem : one for dealing with DNA binding sequences (the '+' model) and the other for dealing with non DNA binding sequences (the '-' model).
Let a+st denote the transition probability of s, t □ Σ inside DNA binding sequence and let a−st denote the transition probability outside a DNA binding sequence (see tables 5.1,5.2 for the values of these probabilities respectively).
Therefore we can compute a logarithmic likelihood score for a sequence X by:

$$Score(X) = \log \frac{P(X|\text{CpG island})}{P(X|\text{non CpG island})} = \sum_{i=1}^{L} \log \frac{a^{+}_{x_{i-1},x_i}}{a^{-}_{x_{i-1},x_i}}$$

The higher this score, the more likely it is that X is a CpG Island.

Problem 2: Locating DNA binding sub-sequences in a long Amino acid sequence.

INPUT: A long amino acid sequence X = (x1, . . . , xL) □ Σ∗.
(where Σ = {'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'}).
QUESTION: Locate the DNA binding sub-sequences along X.

A naive approach for solving this problem will be to extract a sliding window Xk = (xk+1, . . . , xk+l_) of a given length l (where l << L, usually several hundred bases long, and 1 ≤ k ≤ L- l) from the sequence and calculate Score(Xk) for each one of the resulting subsequences. Subsequences that receive positive scores are potential DNA binding sequences

The main disadvantage in this algorithm is that we have no information about the lengths of the islands, while the algorithm suggested above assumes that those islands are at least l nucleotides long. Should we use a value of l which is too large, the DNA binding sequences would be short substrings of our windows, and the score we give those windows may not be high enough. On the other hand, windows that are too small might not provide enough information to determine whether their bases are distributed like those of an island or not. A better solution to this problem is to combine the two Markov chains of the last section into a unified model, with a small probability of switching from one chain to the other at each transition point. However, this introduces the complication that we now have two states corresponding to each amino acid symbol. Therefore we need to use another model for this problem.

The model we have used to solve the above problem is :

Definition:A Hidden Markov Model (HMM) is a triplet M = (Σ,Q, Θ), where:

• Σ is an alphabet of symbols.

• Q is a finite set of states, capable of emitting symbols from the alphabet Σ.

• Θ is a set of probabilities, comprised of:

– State transition probabilities, denoted by akl for each k, l ϵ Q.

– Emission probabilities, denoted by ek(b) for each k ϵ Q and b ϵ Σ.

A path Π = (π1, . . . , πL) in the model M is a sequence of states. The path itself follows a simple Markov chain, so the probability of moving to a given state depends only on the previous state. As in the Markov chain model, we can define the state transition probabilities in terms of Π :

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k)$$

However, in a hidden Markov model there isn't a one-to-one correspondence between the states and the symbols. Therefore, in a HMM we introduce a new set of parameters, ek(b), called the emission probabilities.

Given a sequence X = (x1, . . . , xL) ϵ Σ∗ we define :

$$e_k(b) = P(x_i = b | \pi_i = k)$$

ek(b) is the probability that symbol b is seen when we are in state k.

The probability that the sequence X was generated by the model M given the path Π is therefore:

$$P(X, \Pi) = a_{\pi_0,\pi_1} \cdot \prod_{i=1}^{L} e_{\pi_i}(x_i) \cdot a_{\pi_i,\pi_{i+1}}$$

Where for our convenience we denote π0 = begin and πL+1 = end.

A model of transition probability matrix generated for two way classification HMM ie.,DNA binding/Non DNA binding:

	DNA binding (+) state 20 amino acid residues	Non-DNA binding (-) state 20 amino acid residues
DNA binding (+) state 20 amino acid residues	20 X 20 matrix of transition probabilities of DNA binding sequences against DNA bnding sequences	20 X 20 matrix of transition probabilities of DNA binding sequences against Non-DNA bnding sequences
Non-DNA binding (-) state 20 amino acid residues	20 X 20 matrix of transition probabilities of Non-DNA binding sequences against DNA bnding sequences	20 X 20 matrix of transition probabilities of Non-DNA binding sequences against Non-DNA bnding sequences

Example 5.1.3a An HMM for detecting DNA binding sub-sequences in a long Amino acid sequence

The model contains forty states corresponding to the twenty symbols of the alphabet

Σ = {'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'}:

State: A+,R+,N+,D+,C+,Q+,E+,K+,M+,F+,P+,S+,T+,W+,Y+,V+,G+,H+,I+,L+

Emitted Symbol:

A ,R ,N ,D ,C ,Q ,E ,K ,M ,F ,P ,S ,T ,W ,Y ,V ,G ,H ,I , L

State: A- ,R- ,N- ,D- ,C- ,Q- ,E- ,K- ,M- ,F- ,P- ,S- ,T- ,W- ,Y- ,V- ,G- ,H- ,I- ,L-

Emitted Symbol:

A ,R ,N ,D ,C ,Q ,E ,K ,M ,F ,P ,S ,T ,W ,Y ,V ,G ,H ,I , L

If the probability for staying in a DNA binding sequence is p and the probability of staying outside it is q, then the transition probabilities will be as described in following 4 tables:

(derived from the transition probabilities given in table 5.1 under the assumption that we lose memory when moving from/into a DNA binding sequence, and that we ignore background probabilities).

In this special case the emission probability of each state X+ or X− is exactly 1 for the symbol X and 0 for any other symbol.

Example :An HMM for modeling a dishonest casino dealer

Suppose a dealer in a casino rolls a die. The dealer use a fair die most of the time, but occasionally he switches to a loaded die. The loaded die has probability 0.5 of a six and probability 0.1 for the numbers 1 to 5. We also know that the dealer does not tend to change dies - he switches from a fair to a loaded die with probability 0.05, and the probability of switching back is 0.1. Given a sequence of die rolls we wish to determine when did the dealer use a fair die and when did he use a loaded die.

The corresponding HMM is:

• The states are Q = {F,L}, where F stands for ”fair” and L for ”loaded”.

• The alphabet is $\Sigma = \{1, 2, 3, 4, 5, 6\}$.

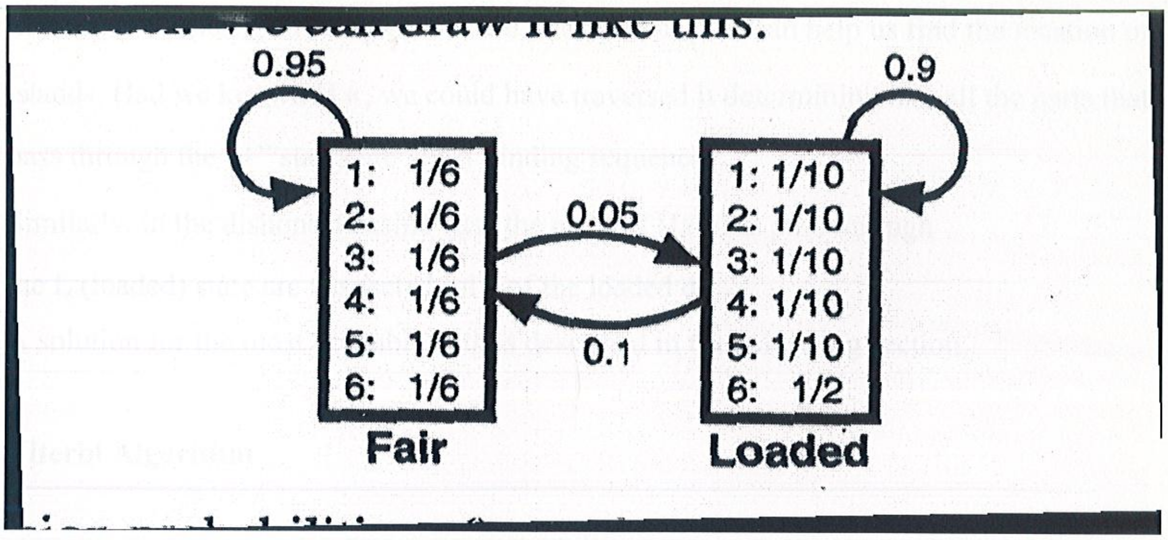


Figure 5.4: HMM for the dishonest casino problem.

The probabilities are:

$$a_{FF} = 0.95 \quad a_{FL} = 0.05$$
$$a_{LL} = 0.9 \quad a_{LF} = 0.1$$
$$\forall_{1 \leq i \leq 6}\, e_F(i) = \frac{1}{6}$$
$$\forall_{1 \leq i \leq 5}\, e_L(i) = 0.1 \qquad e_L(6) = 0.5$$

Figure 5.4 gives a full description of the model.

Returning to the general case, we have defined the probability P(X, Π) for a given sequence X and a given path Π. However, we do not know the actual sequence of states (π1, . . . , πL) that emitted (x1, . . . , xL). We therefore say that the generating path of X is hidden.

INPUT: A hidden Markov model M = (Σ,Q, Θ) and a sequence X □ Σ∗, for which the generating path Π = (π1, . . . , πL) is unknown.

QUESTION: Find the most probable generating path Π∗ for X, i.e., a path such that P(X,Π∗) is maximized. We denote this also by:

$$\Pi^* = \arg\max_{\Pi}\{P(X, \Pi)\}$$

In the DNA binding sub-sequences case, the optimal path can help us find the location of the islands. Had we known Π∗, we could have traversed it determining that all the parts that pass through the "+" states are DNA binding sequences.

Similarly, in the dishonest casino case the parts of Π∗ that pass through the L (loaded) state are suspected rolls of the loaded die.
A solution for the most probable path is described in the following section.

Viterbi Algorithm

We can calculate the most probable path in a hidden Markov model using a dynamic programming algorithm. This algorithm is widely known as Viterbi Algorithm. Viterbi devised this algorithm for the decoding problem, even though its more general description was originally given by Bellman [3].
Let X be a sequence of length L. For k є Q and 0 ≤ i ≤ L, we consider a path Π ending at k, and the probability of Π generating the prefix (x1, . . . , xi) of X. Denote by vk (i) the probability of the most probable path for the prefix (x1, . . . , xi) that ends in state k.

$$v_k(i) = \max_{\{\Pi|\Pi_i=k\}} P(x_1, \ldots, x_i, \Pi)$$

1. Initialize:

$$\begin{aligned} v_{begin}(0) &= 1 \\ \forall_{k \neq begin} \quad v_k(0) &= 0 \end{aligned}$$

2. For each i = 0, . . . , L − 1 and for each l □ Q recursively calculate:

$$v_l(i+1) = e_l(x_{i+1}) \cdot \max_{k \in Q} \{v_k(i) \cdot a_{kl}\}$$

3. Finally, the value of P(X,Π∗) is:

$$P(X, \Pi^*) = \max_{k \in Q} \{v_k(L) \cdot a_{k,end}\}$$

We can reconstruct the path $\Pi*$ itself by keeping back pointers during the recursive stage and tracing them.

Results:

The stated in the dishonest casino model the probability for staying in the fair dies is a_{ff}
And the probability of moving out of fair die is (1- a_{ff})
Similarly when we are in the Loeade state the probability for staying in the loaded state itesel is a_{LL}
And the probability of moving out of loaded die is (1- a_{LL})

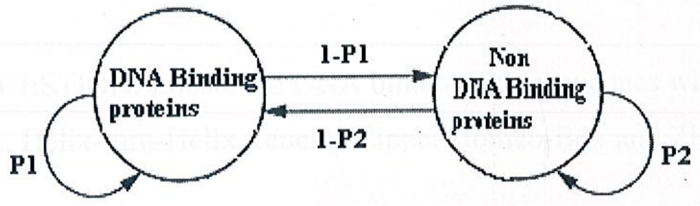
And these vaues are optimized at the values:

$$a_{FF} = 0.95 \quad a_{FL} = 0.05$$
$$a_{LL} = 0.9 \quad a_{LF} = 0.1$$

Now in case of DNA binding and Non DNA binding proteins these values have to be optimized.

So, starting at the highest possible transition from one state to the same state ie., the values of probability for staying in the DNA binding sequence(P1)=0.99 and
probability for staying in the Non-DNA binding sequence(P2)=0.99
(1 out of every 100 residues in the protein sequence the transition is possible from one state to the another)
By iterating the values of P1 and P2 all possible combinations of (P1,P2) are checked and the viterbi algorithm is implemented.

The optimal result of Hidden Marcov Model from the viterbi algorithm is predicting with 70% accuracy for both DNA binding and Non-DNA binding training data sequences at (P1,P2)=(0.9,0.6)



Where

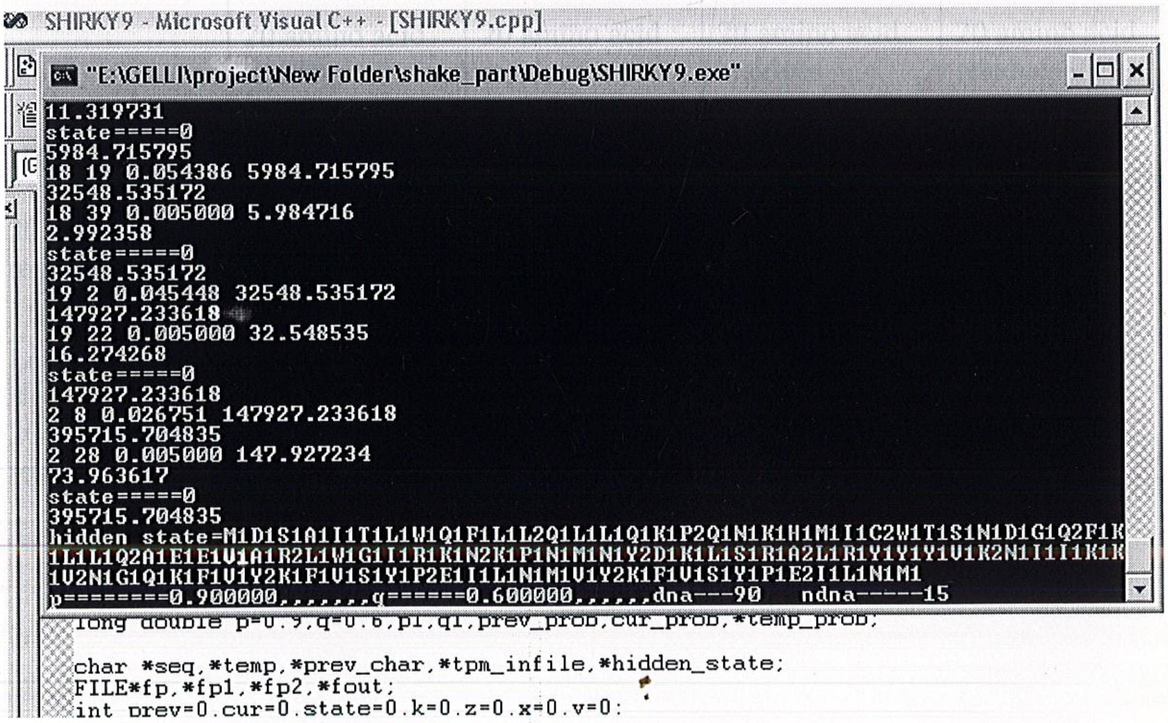
P1 and P2 are the transition probabilities from one state to the same state.

1-P1 & 1-P2 are transition probabilities from one state to the other

Values of P1 & P2 are 0.9 &0.6 respectively.

So by substituting the values of P1,P2 in the transition probability matrices generated earlier for the HMM they change into:

The screenshot of an output for a DNA binding protein is given below:


SHIRKY9 - Microsoft Visual C++ - [SHIRKY9.cpp]
"E:\GELLI\project\New Folder\shake_part\Debug\SHIRKY9.exe"

```
11.319731
state=====0
5984.715795
18 19 0.054386 5984.715795
32548.535172
18 39 0.005000 5.984716
2.992358
state=====0
32548.535172
19 2 0.045448 32548.535172
147927.233618
19 22 0.005000 32.548535
16.274268
state=====0
147927.233618
2 8 0.026751 147927.233618
395715.704835
2 28 0.005000 147.927234
73.963617
state=====0
395715.704835
hidden state=M1D1S1A1I1T1L1W1Q1F1L1L2Q1L1L1Q1K1P2Q1N1K1H1M1I1C2W1T1S1N1D1G1Q2F1K
1L1L1Q2A1E1E1V1A1R2L1W1G1I1R1K1N2K1P1N1M1N1Y2D1K1L1S1R1A2L1R1Y1Y1Y1V1K2N1I1I1K1K
1V2N1G1Q1K1F1V1Y2K1F1V1S1Y1P2E1I1L1N1M1V1Y2K1F1V1S1Y1P1E2I1L1N1M1
p========0.900000,,,,,,,,q======0.600000,,,,,,dna---90   ndna-----15
```

```
long double p=0.9,q=0.6,p1,q1,prev_prob,cur_prob,*temp_prob;

char *seq,*temp,*prev_char,*tpm_infile,*hidden_state;
FILE*fp,*fp1,*fp2,*fout;
int prev=0,cur=0,state=0,k=0,z=0,x=0,v=0;
```

Problem 3: Locating DNA binding classes in a long Amino acid sequence.

INPUT: A long amino acid sequence X = (x1, . . . , xL) □ Σ∗.

(where Σ = {'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'}).

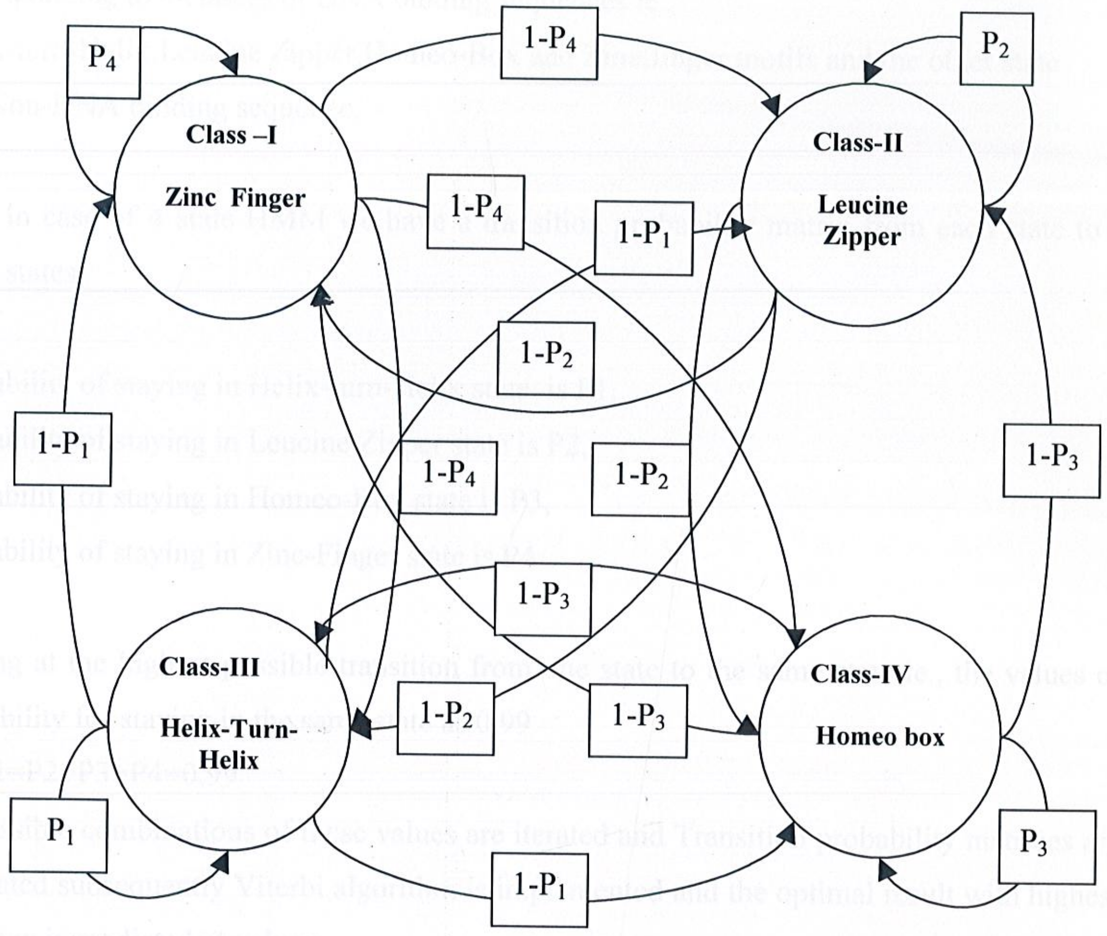
QUESTION: Locate the DNA binding sub-sequences which are devided into four classes ie., Helix-turn-Helix,Leucine Zipper,Homeo-Box and Zinc;finger motifs.

The HMM for modeling the same is:

	HTH 20 amino acid residues	Leucine zipper 20 amino acid residues	Zinc finger 20 amino acid residues	Homeo Box 20 amino acid residues
HTH 20 amino acid residues	HTH X HTH	HTH X Leucine zipper	HTH X Zinc finger	HTH X Homeo Box
Leucine zipper 20 amino acid residues	Leucine zipper X HTH	Leucine zipper X Leucine zipper	Leucine zipper X Zinc finger	Leucine zipper X Homeo Box

Zinc finger 20 amino acid residues	Zinc finger X HTH	Zinc finger X Leucine zippe	Zinc finger X Zinc finger	Zinc finger X Homeo Box
Homeo Box 20 amino acid residues	Homeo Box X HTH	Homeo Box X Leucine zippe	Homeo Box X Zinc finger	Homeo Box X Homeo Box

A model of the transition probability matrix of four way HMM.
The individual boxes are the values of the transition probability values where ith residue is the row state and i+1 th residue is the coloum state



P1, P2, P3 & P4 are the Transition probabilities from one state to the same state.

1-P1, 1-P2, 1-P3 & 1-P4 are the Transition probabilities from one state to the other.

The values of P1, P2, P3 & P4 are 0.4, 0.6, 0.5, 0.3 respectively.

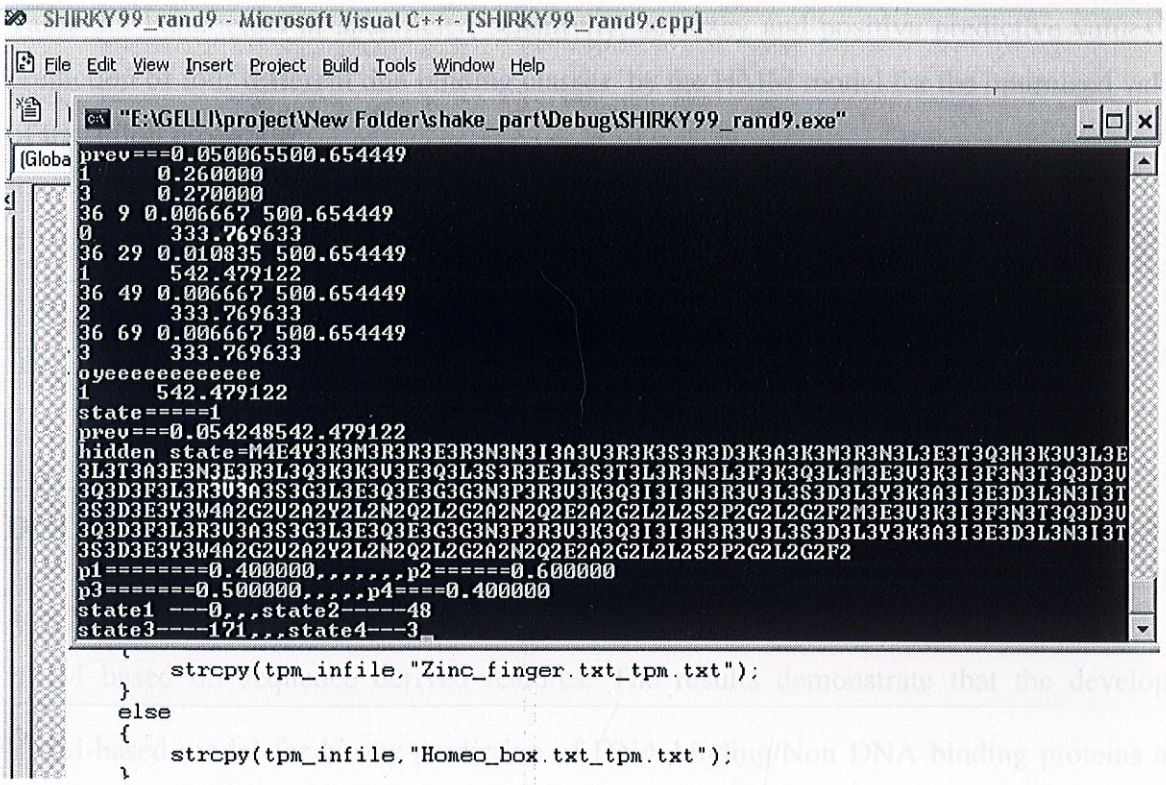
The 4 state HMM: In the previous case we only have two hidden states corresponding to DNA binding sequences and Non-DNA binding sequences. Now, we have 4 hidden states corresponding to 4 classes of DNA binding sequences ie.,
Helix-turn-Helix,Leucine Zipper,Homeo-Box and Zinc;finger motifs and the other state
For Non-DNA binding sequence.

Now in case of 4 state HMM we have a transition probability matrix from each state to 3 other states
If the
Probability of staying in Helix-turn-Helix state is P1,
Probability of staying in Leucine Zipper state is P2,
Probability of staying in Homeo-Box state is P3,
Probability of staying in Zinc-Finger state is P4

starting at the highest possible transition from one state to the same state ie., the values of probability for staying in the same state as 0.99
ie., P1=P2=P3=P4=0.99
all possible combinations of these values are iterated and Transition probability matrices are generated subsequently Viterbi algorithm is implemented and the optimal result with highest accuracy is predicted at values:
P1=0.4,p2=0.6,p3=0.5,p4=0.4

The subsequent transition probabilities are given below:

The screenshot of the output for a Zinc finger protein is given below:


SHIRKY99_rand9 - Microsoft Visual C++ - [SHIRKY99_rand9.cpp]
File Edit View Insert Project Build Tools Window Help
"E:\GELLI\project\New Folder\shake_part\Debug\SHIRKY99_rand9.exe"
(Globa

```
prev===0.050065500.654449
1      0.260000
3      0.270000
36 9 0.006667 500.654449
0       333.769633
36 29 0.010835 500.654449
1       542.479122
36 49 0.006667 500.654449
2       333.769633
36 69 0.006667 500.654449
3       333.769633
oyeeeeeeeeeeee
1      542.479122
state=====1
prev===0.054248542.479122
hidden state=M4E4Y3K3M3R3R3E3R3N3N3I3A3V3R3K3S3R3D3K3A3K3M3R3N3L3E3T3Q3H3K3V3L3E
3L3T3A3E3N3E3R3L3Q3K3K3V3E3Q3L3S3R3E3L3S3T3L3R3N3L3F3K3Q3L3M3E3V3K3I3F3N3T3Q3D3V
3Q3D3F3L3R3V3A3S3G3L3E3Q3E3G3G3N3P3R3V3K3Q3I3I3H3R3V3L3S3D3L3Y3K3A3I3E3D3L3N3I3T
3S3D3E3Y3W4A2G2V2A2Y2L2N2Q2L2G2A2N2Q2E2A2G2L2L2S2P2G2L2G2F2M3E3V3K3I3F3N3T3Q3D3V
3Q3D3F3L3R3V3A3S3G3L3E3Q3E3G3G3N3P3R3V3K3Q3I3I3H3R3V3L3S3D3L3Y3K3A3I3E3D3L3N3I3T
3S3D3E3Y3W4A2G2V2A2Y2L2N2Q2L2G2A2N2Q2E2A2G2L2L2S2P2G2L2G2F2
p1========0.400000,,,,,,,p2======0.600000
p3=======0.500000,,,,,p4====0.400000
state1 ---0,,,state2-----48
state3----171,,,state4---3_
```

```
        strcpy(tpm_infile,"Zinc_finger.txt_tpm.txt");
    }
    else
    {
        strcpy(tpm_infile,"Homeo_box.txt_tpm.txt");
    }
```

Table 2.2 Parameters of specificity, sensitivity, accuracy and positive predictive values for prediction of dna binding and non-dna binding proteins by the HMM model for the optimized values of transition probability.

Transition probability	Accuracy	Specificity	Sensitivity	Q(Pred)
0.9-0.6	0.73.18	0.78.19	0.8213	72.31

Table 2.3 Parameters of specificity, sensitivity, accuracy and positive predictive values for prediction of four different dna binding classes by the HMM model for the optimized values of transition probability.

Transition probability	Accuracy	Specificity	Sensitivity	Q(Pred)
0.4-0.6-0.5-0.4	0.7010	0.7273	0.8020	65.13

DISCUSSION

The tool DBPpred has been developed in this study using two layered model of HMM based on sequence derived features. The results demonstrate that the developed HMM-based model for binary prediction of DNA binding/Non DNA binding proteins and classification of DNA binding proteins into four major classes is adequate and can be considered an effective tool for 'in silico' screening. The results also demonstrate that the sequence information readily accessible from the protein sequences only, can produce a variety of useful information to be used '*in silico*'; clearly demonstrates an adequacy and good predictive power of the developed HMM model. There is strong evidence, that the sequence information do adequately reflect the structural properties of proteins. The structure of a protein is an important determinant for the detailed molecular function of proteins, and would consequently also be useful for prediction of DNA binding/Non DNA binding proteins and for their classification. Based on the analysis of limited sequence information from protein sequences, differences in the transitions between CpG islands and Non-CpG islands have previously been shown to exist and used for prediction of CpG islands in the long DNA chains however the same concept has been applied for the first time in protein sequences with a good accuracy. Thus our work here reflects the fact that sequence derived features can be used for predicting DNA binding protein sequences also. This observation is not surprising considering that the protein sequences have been translated from the DNA sequences itself.

The results of the present work demonstrate that the sequence information with HMM appear to be a very fast protein classification mechanism providing good results, comparable to some of the current efforts in the literature. The developed HMM-based model for DNA bnding and Non DNA binding proteins prediction and their classification into different classes can be used as a powerful tool for filtering through the collections of genome sequences to discover novel enzymes

References:

[1] N. M. Laird A. P. Dempster and D. B. Rubin. Maximum likelihood estimation from incomplete data. *Journal of the Royal Statistics Society*, 39:1–38, 1977.

[2] L. E. Baum. An inequality and associated technique occuring in the statistical analysis of probabilistic functions of markov chains. *Inequalities*, 3:1–8, 1972.

[3] R. Bellman. *Dynamic Programming*. Princeton University Press, Boston, 1957.

[4] R. Durbin, S. Eddy, A. Krough, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
[5] A. Krogh, M. Brown, S. Mian,M. Sj¨olander, and D. Haussler. Hidden markov models in computational biology. applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 4 February 1994.

[6] Anders Krogh, I. Saira Mian, and David Haussler. A hidden markov model that finds genes in E. coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[7] D. Kulp, D. Haussler, M.G. Reese, and F.H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. Smith, editors, *Proc. Conf. On Intelligent Systems in Molecular Biology '96*, pages 134–142. AAAI/MIT Press, 1996. St. Louis, Mo.

[8] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 8 October 1993.

[9] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

Conclusion

From a practical point of view the most important aspect of a prediction method is its ability to make correct predictions. Till date most of the available methods use the 3-d structure of the protein to predict and classify DNA binding proteins. This is a very tedious job and requires much costlier endeavors. The sequence of a protein is an important determinant for the detailed molecular function of proteins, and would consequently also be useful for prediction of DNA binding proteins and inturn intlo various classes. Therefore, we have developed a tool which predicts the DNA binding proteins and their subsequent classes based on both strategies.

ANN by using the sequence derived parameters which are readily accessible and HMM by using directly the sequence information itself.

This thesis contains detailed work on DNA binding proteins prediction and classification. We achieved an accuracy of ~ 70 % for the prediction of DNA binding and Non-DNA binding based on non-redundant dataset of over 500 proteins using the ANN technique. The neural network architecture used for the prediction was optimized for maximum accuracy. This was achieved by gradually testing networks with variable hidden nodes and retaining the one with highest true predictions. This is at par with best prediction tools available till date, but to the contrary, uses a much simpler and efficient prediction method based on sequence features only. This application not only gives optimum result with the dataset used, but also predicts DNA binding proteins from complex genomes to a very high satisfactory level. A much elaborate analysis has been done, which is evident from the extracted data, figures and tables compiled.

Similarly form Hidden marcov models we achieved an accuracy of ~ 69 % for the prediction of DNA binding and Non-DNA binding based on non-redundant dataset of over 400 proteins using the HMM technique. The HMM used for the prediction was optimized for maximum accuracy. This was achieved by gradually testing the model with variable state transition probabilities.

In addition to this tool, we further elaborated on classification of DNA binding sequences to their major classes. The first level of network imitates the binary model, and the second level uses the predicted result of the former to provide a much detailed and useful classification. We achieved an accuracy of ~ 68 % for classifying the protein sequences predicted as enzymes from the first network to their major classes. A similar methodology was used to optimize the network. The neural network architecture used for the classification was optimized for maximum accuracy. This was achieved by gradually testing networks with variable hidden nodes and retaining the one with highest true predictions. Similarly the four layer classification of HMM has achieved an accuracy of 70% prediction of DNA binding and Non-DNA binding based on non-redundant dataset of over 400 proteins using the HMM technique. The HMM used for the prediction was optimized for maximum accuracy. This was achieved by gradually testing the model with variable state transition probabilities.

APPENDIX

Artificial Neural Network:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>

void init(int,int,int);
void train(FILE*,FILE*,int,int,int,int);
void parse(struct node*,int,int,int,int);
void sig(struct node*,int);
void bprop(double*,int,int,int);
void clear(int,int,int);
void optimise(int,int,int);
int max(float*,int);

struct node
{
        long double in,out;
};
struct node *innode;
struct node *hnode;
struct node *onode;


struct weight
{
        long double* w;
}warr[2];

void clear_weights(struct weight*,int,int,int);

int f=0;

void main()
{
        int i,h,o,u,y,call=0,flag=1;
        FILE*fp,*fp1,*fp2,*fp3;
        char ch,*wihfile,*whofile;
        wihfile=(char*)malloc(30*sizeof(char));
        whofile=(char*)malloc(30*sizeof(char));
        srand(time(NULL));
        printf("ENTER THE NUMBER OF INPUT NODES:\n");
        scanf("%d",&i);
        fflush(stdin);
        printf("DO YOU WANT THE PROGRAM TO OPTIMISE THE NUMBER OF
HIDDEN NODES(Y/N)?\n");
        scanf("%c",&ch);
        printf("%c\n",ch);
        fflush(stdin);
        if(ch=='y'||ch=='Y')
        {
                printf("ENTER THE NUMBER TO WHICH YOU WANT THE
PROGRAM TO TEST FOR HIDDEN NODES\n");
                scanf("%d",&y);

        }
        else
        {
                printf("ENTER THE NUMBER OF HIDDEN NODES\n");
                scanf("%d",&y);
                flag=y;
        }
        printf("ENTER THE NUMBER OF OUTPUT NODES\n");
        scanf("%d",&o);

        printf("ENTER THE NUMBER OF ITERATIONS U WANT TO
CONTINUE:\n");
        scanf("%d",&u);
        printf("%d     %d\n",flag,y);
        getch();
        for(h=flag;h<=y;h+=2)
        {
                printf("%d\n",h);
        init(i,h,o);
        sprintf(wihfile,"weightih_%d.txt",h);
        sprintf(whofile,"weightho_%d.txt",h);
        fp2=fopen(wihfile,"w");
        fp3=fopen(whofile,"w");
        for(int g=0;g<u;g++)
        {

        fp=fopen("train_final_normalise.txt","r");
        fp1=fopen("train_output.txt","r");
train(fp,fp1,i,h,o,1);

}
fp=fopen("train_final_normalise.txt","r");
fp1=fopen("train_output.txt","r");
train(fp,fp1,i,h,o,0);
fclose(fp);
fclose(fp1);


printf("\n\nweights between input and hidden \n");

for(int l=0;l<i*h;l++)
{
        printf("%lf ",warr[0].w[l]);
        fprintf(fp2,"%lf ",warr[0].w[l]);
        if((l+1)%h==0)
        {
                printf("\n");
                fprintf(fp2,"\n");
        }
}



printf("\n\nweights between hidden and output \n");


for(l=0;l<h*o;l++)
{
        printf("%lf ",warr[1].w[l]);
        fprintf(fp3,"%lf ",warr[1].w[l]);
        if((l+1)%o==0)
        {
                printf("\n");
                fprintf(fp3,"\n");
        }
}

}
fclose(fp2);
fclose(fp3);
if(ch=='y'||ch=='Y')
{
        optimise(i,h,o);
        }
}


void init(int i,int h,int o)
{
        long double r;

        innode = (struct node*) calloc(i,sizeof(struct node));
        hnode = (struct node*) calloc(h,sizeof(struct node));
        onode = (struct node*) calloc(o,sizeof(struct node));
        warr[0].w=(long double*)calloc(i*h,sizeof(long double));
        warr[1].w=(long double*)calloc(h*o,sizeof(long double));
        printf("im allocating....\n");
        //getch();
        for(int m=0;m<2;m++)
        {
                if(m==0)
                {
                        for(int k=0;k<i*h;k++)
                        {
                                r=rand()%10;
                                r=(r-5)/10;
                                if(r)
                                {
                                        warr[m].w[k]=r;
                                }
                                else
                                {
                                        while(!r)
                                        {
                                                r=rand()%10;
                                                r=(r-5)/10;
                                        }
                                        warr[m].w[k]=r;
                                }

                        }
                }
                else
                {
                        for(int k=0;k<h*o;k++)
                        {
                                r=rand()%10;
                                r=(r-5)/10;
                                if(r)
```

```
                                        {
                                                warr[m].w[k]=r;
                                        }
                                        else
                                        {
                                                while(!r)
                                                {
                                                        r=rand()%10;
                                                        r=(r-5)/10;
                                                }
                                                warr[m].w[k]=r;
                                        }

                                }
                        }
                }
        }
```

// Training the neural net

```
void train(FILE*fp,FILE*fp1,int i,int h,int o,int call)
{
        int l=0,a=0;//count
        int f=1;
        static int g=0;
        double *arr;


        arr=(double*)calloc(o,sizeof(double));

        if(fp==NULL)
                printf("Empty file handle\n");
        while(!feof(fp) && !feof(fp1))
        {
                for(int p=0;p<i;p++)
                {
                        fscanf(fp,"%lf",&innode[p].in);
                        innode[p].out=innode[p].in;
                }

                for(int a=0;a<o;a++)
                {
                        fscanf(fp1,"%lf",&arr[a]);
                }

                        parse(innode,i,h,o,call);


                        if(call)
                        {
                                bprop(arr,i,h,o);
                        }

                        a=0;
                        l=0;
                        continue;

                l++;
                f++;
                a++;
        }
        free(arr);
        fclose(fp);
        fclose(fp1);
}
```

//Parse each set of values from input file

```
void parse(struct node *innode,int i,int h,int o,int call)

{

        int x=0;
        FILE* fp1;
        static int qw=0;
        char* file_name;
        file_name=(char*)malloc(40*sizeof(char));
        clear(i,h,o);

        for(int q=0;q<h;q++)
        {
                for(int g=0;g<i;g++)
                {
                        hnode[q].in+=innode[g].out*warr[0].w[x];
                        x++;
                }
        }
        sig(hnode,h);
        x=0;
        for(int g=0;g<o;g++)
        {
                for(int p=0;p<h;p++)
                {
                        onode[g].in+=hnode[p].out*warr[1].w[x];
                        x++;
                }
        }
        sig(onode,o);

        if(call==0)
        {


                qw++;
                f++;

                sprintf(file_name,"output_hidden_%d.txt",h);
                fp1=fopen(file_name,"a");
                if(fp1==NULL)
                {

                        printf("%d    file cant b opened\n",qw);
                        getch();

                }
                        for(int d=0;d<o;d++)
                        {
                                fprintf(fp1,"%lf\t",onode[d].out);

                        }

                                fprintf(fp1,"\n");

                                fclose(fp1);
        }
        free(file_name);


}
```

```
void sig(struct node*x,int size)
{
        for(int i=0;i<size;i++)
        {
        if((1/(1+exp(-x[i].in))))
                x[i].out=(1/(1+exp(-x[i].in)));
        }
}


void bprop(double* arr,int i,int h,int o)
{
        long double* error,*error2;
        long double x=0.0;
        int y=0;
        int k=0;
        error=(long double*)calloc(o,sizeof(long double));
        for(int l=0;l<o;l++)
        {
                error[l]=0.1*onode[l].out*(1-onode[l].out)*(arr[l]-onode[l].out);
        }
        for(int f=0;f<h*o;f++)
        {
                k=0;
                while(k<o)
                {
                        warr[1].w[f]=warr[1].w[f]+(error[k]*hnode[(int)f/o].out);
                        k++;
                        f++;
                }
                f--;
        }
        error2=(long double*)calloc(h,sizeof(long double));
        for(l=0;l<h;l++)
        {
                for(int m=0;m<o;m++)
                {
                        x+=error[m]*warr[0].w[y];
                        y++;
                }
                error2[l]=0.1*hnode[l].out*(1-hnode[l].out)*x;
        }
        for(f=0;f<(i*h);f++)
        {
                k=0;
                while(k<h)
                {
                                warr[0].w[f]=warr[0].w[f]+(error2[k]*innode[(int)f/h].out);
                                k++;
                                f++;
                }
                f--;
        }



free(error);
free(error2);
}


void clear(int i,int h,int o)
{
        for(int p=0;p<h;p++)
        {
                hnode[p].in=hnode[p].out=0.0;
        }
        for(int k=0;k<o;k++)
        {
                onode[k].in=onode[k].out=0.0;
        }
}


void optimise(int a,int b,int c)
{
        FILE*fp,*fp1,*results;
        int max1=0,max2=0,hidden_optimised=0;
        char*file_name,*file_results;
        float temp=0.0;
        float
*temp1,*temp2,*acc_count,**accuracy,**ppv,**sensitivity,**specificity,**npv,**mcc;
        float *tp,*falsep,*fn,*tn;
        acc_count=(float*)malloc(b*sizeof(float));
        file_name=(char*)malloc(30*sizeof(char));
        temp1=(float*)malloc(c*sizeof(float));
        temp2=(float*)malloc(c*sizeof(float));
        accuracy=(float**)malloc(b*sizeof(float*));
        ppv=(float**)malloc(b*sizeof(float*));
        specificity=(float**)malloc(b*sizeof(float*));
```

```
sensitivity=(float**)malloc(b*sizeof(float*));
npv=(float**)malloc(b*sizeof(float*));
mcc=(float**)malloc(b*sizeof(float*));
tp=(float*)malloc(c*sizeof(float));
falsep=(float*)malloc(c*sizeof(float));
fn=(float*)malloc(c*sizeof(float));
tn=(float*)malloc(c*sizeof(float));




for(int q=0;q<b;q++)
{
        accuracy[q]=(float*)malloc(c*sizeof(float));
        ppv[q]=(float*)malloc(c*sizeof(float));
        specificity[q]=(float*)malloc(c*sizeof(float));
        sensitivity[q]=(float*)malloc(c*sizeof(float));
        npv[q]=(float*)malloc(c*sizeof(float));
        mcc[q]=(float*)malloc(c*sizeof(float));

}

file_results=(char*)malloc(30*sizeof(char));

for(int t=0;t<b;t++)
{
        acc_count[t]=0.0;
}
results=fopen("results_2.txt","w");
for(int i=1,d=0;i<b;i+=2,d++)
{
        sprintf(file_name,"output_hidden_%d.txt",i);
        fp=fopen(file_name,"r");
        puts(file_name);
        fp1=fopen("train_output.txt","r");
        if(fp1==NULL)
        {
                printf("Unable to open train_output file\n");
                getch();
        }
        if(fp==NULL)
        {
                printf("Unable to open file %s\n",file_name);
                getch();
        }
        else
```

```
{

        while(!feof(fp1)&&!feof(fp))
        {
                for(int x=0;x<c;x++)
                {

                        fscanf(fp,"%f",&temp1[x]);
                        fscanf(fp1,"%f",&temp2[x]);
                }
                max1=max(temp1,c);
                max2=max(temp2,c);
                for(int g=0;g<c;g++)
                {
                        if(max1==max2)
                        {
                                printf("%d   %d\n",max1,max2);
                                acc_count[d]+=1.0;
                                if(max1==g)
                                {
                                        tp[g]+=1.0;
                                        printf("heree....%f",tp[g]);
                                        getch();
                                }
                                else
                                {
                                        tn[g]+=1.0;
                                }
                        }

                        else
                        {
                                if(max1==g)
                                {
                                        falsep[g]+=1.0;
                                }
                                else
                                {
                                        fn[g]+=1.0;
                                }
                        }

                }
        }
        for(int r=0;r<c;r++)
        {
```

```
                                        accuracy[d][r]=(tp[r]+tn[r])/(tp[r]+falsep[r]+fn[r]+tn[r]);
                                        ppv[d][r]=(tp[r])/(tp[r]+falsep[r]);
                                        sensitivity[d][r]=(tp[r])/(tp[r]+fn[r]);
                                        specificity[d][r]=(tn[r])/(falsep[r]+tn[r]);
                                        npv[d][r]=(tn[r])/(tn[r]+fn[r]);
                                        mcc[d][r]=((tp[r]*falsep[r])-
(tn[r]*fn[r]))/sqrt((tp[r]+tn[r])*(tp[r]+fn[r])*(fn[r]+tn[r])*(fn[r]+falsep[r]));
                                }
                        }
                        fclose(fp);
                }
                for(int u=0;u<d;u++)
                {
                        for(int cn=0;cn<c;cn++)
                        {
                                fprintf(results,"accuracy count for %d class when %d hidden
nodes%f\n",cn,(u*2+1),accuracy[u]);
                                fprintf(results,"ppv[%d][%d]====%f\n",(u*2+1),cn,ppv[u]);

                fprintf(results,"sensitivity[%d][%d]=====%f\n",(u*2+1),cn,sensitivity[u]);

                fprintf(results,"specificity[%d][%d]=====%f\n",(u*2+1),cn,specificity[u]);
                                fprintf(results,"npv[%d][%d]====%f\n",(u*2+1),cn,npv[u]);

                fprintf(results,"mcc[%d][%d]=====%f\n\n\n\n",(u*2+1),cn,mcc[u]);
                        }
                        fprintf(results,"\n\n\n\n\n");
                }
                        hidden_optimised=temp*2+1;
                printf("THE NETWORK IS FOUND TO BE OPTIMISED AT %d NUMBER OF
HIDDEN NODES\n",hidden_optimised);
                printf("SO PLEASE NOTE DOWN THE NUMBER OPTIMISED NUMBER OF
HIDDEN NODES FOR FURTHER USE\n");
}

int max(float *arr,int size)
{

        float max=arr[0];
        int ret_value=0;
        for(int a=0;a<size;a++)
        {
                if(arr[a]>max)
                {
                        max=arr[a];
                        ret_value=a;
                        }
                }
                return ret_value;
}

void clear_weights(struct weight* warr,int i,int h,int o)
{

        for(int m=0;m<2;m++)
        {
                if(m==0)
                {
                        for(int k=0;k<i*h;k++)
                        {
                                warr[m].w[k]=0.0;



                        }
                }
                else
                {
                        for(int k=0;k<h*o;k++)
                        {
                                warr[m].w[k]=0.0;



                        }
                }
        }
}
```

Appendix –II

C-program implementing 2 state HMM

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
long double max(long double*);
int to_int(char);
char* to_char(int);

char s[21]={'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'};

char
states[80][4]={"A1","R1","N1","D1","C1","Q1","E1","K1","M1","F1","P1","S1","T1","
W1","Y1","V1","G1","H1","I1","L1",

"A2","R2","N2","D2","C2","Q2","E2","K2","M2","F2","P2","S2","T2","W2","Y2","V2"
,"G2","H2","I2","L2",

        "A3","R3","N3","D3","C3","Q3","E3","K3","M3","F3","P3","S3","T3","W3","Y
3","V3","G3","H3","I3","L3",

        "A4","R4","N4","D4","C4","Q4","E4","K4","M4","F4","P4","S4","T4","W4","Y
4","V4","G4","H4","I4","L4"};
int no_class=4;

void main()
{
```

```
int hth=0,lz=0,zf=0,hb=0,count[4]={0,0,0,0},start_state=0;

long double start_prob[4],prev_prob,cur_prob,*temp_prob;

char *seq,*temp,*prev_char,*tpm_infile,*hidden_state;
FILE*fp,*fp1,*fp2;
int prev=0,cur=0,state=0,k=0,z=0,x=0,y=0;

tpm_infile=(char*)malloc(20*sizeof(char));
hidden_state=(char*)malloc(15000*sizeof(char));
temp_prob=(long double*)malloc(no_class*sizeof(long double));



printf("enter the probability for startin in HTH sequences: ");
scanf("%lf",&start_prob[0]);
printf("enter the probability for startin in LZ sequences: ");
scanf("%lf",&start_prob[1]);
printf("enter the probability for startin in HTH sequences: ");
scanf("%lf",&start_prob[2]);
printf("enter the probability for startin in HTH sequences: ");
scanf("%lf",&start_prob[3]);

fp2=fopen("seq.txt","r");

if(fp2==NULL)
{
        printf("unable to open file\n");
        return;
}

seq=(char*)calloc(15000,sizeof(char));
```

```
temp=(char*)malloc(150*sizeof(char));
prev_char=(char*)malloc(2*sizeof(char));
strcpy(seq,"");
strcpy(hidden_state,"");
while(!feof(fp2))
{
        fscanf(fp2,"%s",temp);
        strcat(seq,temp);
}


long double tpm[80][80],cla[4][20][20];


//tpm=(long double**)malloc(40*sizeof(long double));
//*tpm=(long double*)malloc(40*sizeof(long double));


for(int h=0;h<4;h++)
{
        if(h==0)
        {
                strcpy(tpm_infile,"Hth.txt_tpm.txt");
        }
        else if(h==1)
        {
                strcpy(tpm_infile,"Leucine_zipper.txt_tpm.txt");
        }
        else if(h==2)
        {
                strcpy(tpm_infile,"Zinc_finger.txt_tpm.txt");
        }
```

```
		else
		{
			strcpy(tpm_infile,"Homeo_box.txt_tpm.txt");
		}
		fp=fopen(tpm_infile,"r");



		//printf("hieeeeeeeeeeeeeeeee\n");
		if(!fp)
		{
			printf("file %s cant b opened\n",tpm_infile);
			return;
		}
		for(int l=0;l<20;l++)
		{
			for(int m=0;m<20;m++)
			{
				//printf("2222222222222222");
				//getch();
				fscanf(fp,"%lf",&cla[h][l][m]);
				//printf("%d,%lf\n",h,cla[h][l][m]);
				//getch();
			}
		}
		//printf("hieeeeeeeeeeeeeeeee\n");
		fclose(fp);
	}
	//printf("%lf\n",tpm[1][1]);
	//getch();
	for(int g=0;g<80;g++)
	{
```

```
          for(int h=0;h<80;h++)
          {
                    tpm[g][h]=0.0;
//                  printf("%f \n ",tpm[g][h]);
          }
//        printf("\n");
}
//getch();

///printf("heeeeeeeeeeceeeeeeeeeeeeeeeeeeeeeeee\n");
//getch();
//printf("\n\n\n\n\n%lf\n",tpm[0][0]);
//getch();
k=80/(no_class);

long double arr1[6]={0.9,0.8,0.7,0.6,0.5,0.4};
//float arr2[]={0.9,0.8,0.7,0.6,0.5,0.4};
long double p[4]={0.0,0.0,0.0,0.0};

//p=(long double*)malloc(no_class*sizeof(float));
for(int a=0;a<6;a++)
{
          p[0]=arr1[a];
          for(int b=0;b<6;b++)
          {
                    p[1]=arr1[b];
                    for(int c=0;c<6;c++)
                    {
                              p[2]=arr1[c];
                              for(int d=0;d<6;d++)
                              {
```

```
                    p[3]=arr1[d];


//      dna=0;
        for(int gb=0;gb<4;gb++)
        {
                count[gb]=0;
        }

for(int i=0;i<80;i++)
{
        z=i/k;
        //printf("%d,,,,,,hieeeeeeeeeeeeeeeeee\n",z);
        //getch();

        for(int j=0;j<80;j++)
        {
                if(i%20&&j%20)
                {
                        //getch();
                }
                if(i<20*(z+1)&&j<20*(z+1)&&i>=20*z&&j>=20*z)
                {

                        //printf("%d %d\n",(20-i),(20-j));

                        //if(((20-i)>=0)&&((20-j)>=0)&&((20-i)<20)&&((20-j)<20))
                        //{


                            tpm[i][j]=(cla[z][abs(i-(20*z))][abs(j-(20*z))]*p[z]);
                    //printf("ur here..\n");
                    printf("%d,%d %d,%lf          \n",z,(i),(j),tpm[i][j]);
                    //getch();
                //}
            }
            if((i<=20*(z+1)&&j>=20*(z+1))||(j<20*z&&i<=20*(z+1)))
            //{

            //else
            {
                tpm[i][j]=(1-p[z])/60;
                printf("%d,%d %d,%lf          \n",z,(i),(j),tpm[i][j]);
            //      getch();
            }
            //}

        }
}

        //getch();

for(i=0;i<strlen(seq);i++)
{
        //printf("%lf       %lf\n",(q1),p1);
        start_state=(int)max(start_prob);


if(i==0)
{

        prev_prob=start_prob[start_state];
        printf("%d",to_int(seq[i]));
        prev=(start_state*20)+(long double)(to_int(seq[i]));
        prev_char=to_char(prev);
        strcat(hidden_state,prev_char);
        printf("%d \n",prev);
        //getch();
        //puts(prev_char);

}
else
{
        //cur=(1*20)+to_int(seq[i]);
        //prev_char=to_char(prev);

        for(int d=0;d<no_class;d++)
        {
                cur=((d)*20+(long double)to_int(seq[i]));
                printf("%d %d %lf %lf\n",prev,cur,tpm[prev][cur],prev_prob);
                temp_prob[d]=((prev_prob*tpm[prev][cur])*100);
                if(temp_prob[d]>1000)
                {

                        temp_prob[d]=((prev_prob*tpm[prev][cur])/1000);
                }
                printf("%lf \n",temp_prob[d]);
                //getch();
        }
        printf("oyeeeeeeeeeeee\n");

        state=(int)max(temp_prob);
```

```
            printf("state=====%d\n",state);
            //getch();
            cur=((state)*20+(long double)to_int(seq[i]));
            prev=cur;
            count[state]++;

            //printf("%d\n",state);
            strcpy(prev_char,to_char(prev));
            strcat(hidden_state,prev_char);
            //printf("%d \n",prev);
            prev_prob=temp_prob[state];
            printf("%lf \n",prev_prob);
//          getch();
            //puts(prev_char);
}
}
printf("hidden state=%s\n",hidden_state);
printf("p1========%f,,,,,,p2======%f\np3=======%f,,,,,p4====%f\nstate1 ---
%d,,,state2-----%d\nstate3----%d,,,state4---
%d",p[0],p[1],p[2],p[3],count[0],count[1],count[2],count[3]);
getch();
//}
//}
}
}
}
}
}
```

```
long double max(long double* arr)
{
        long double temp=arr[0],ret_value=0;
        int store[4];
        int f=0,n=0;

        for(int i=1;i<no_class;i++)
        {
                if(arr[i]>temp)
                {
                        printf("%d     %f\n",i,arr[i]);
                        temp=arr[i];
                        ret_value=i;
                        f=i;
                }

        }
        for(int j=0;j<no_class;j++)
        {
                if(arr[j]==arr[f]&&f!=j)
                {
                        store[j]=j;
                        n++;
                }
        }
        if(n)
        {
                printf("areyy bababababbaa");
                getch();
        }
//      ret_value=store[rand()%n];
```

```
        printf("fck uuu %d   %f\n",n,ret_value);
        //getch();
//      return (long double)rand()%store[j];
        return ret_value;
}

char* to_char(int a)
{
        char *b;
        b=(char*)malloc(sizeof(char));
        strcpy(b,states[a]);
        //printf("fck uuuuu %d        %s\n",a,b);
        //getch();
        return b;
}

int to_int(char a)
{
        //printf("in module\n");
        int ret_value;
        for(int i=0;i<20;i++)
        {
                if(s[i]==a)
                {
                        ret_value=i;
                }
        }
        //printf("%d \n",ret_value);
        return ret_value;
}
```

Appendix-III

C-programme implementing 4 state HMM

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
long double max(long double*);
int to_int(char);
char* to_char(int);

char s[21]={'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'};

char
states[80][4]={"A1","R1","N1","D1","C1","Q1","E1","K1","M1","F1","P1","S1","T1","
W1","Y1","V1","G1","H1","I1","L1",

"A2","R2","N2","D2","C2","Q2","E2","K2","M2","F2","P2","S2","T2","W2","Y2","V2"
,"G2","H2","I2","L2",

        "A3","R3","N3","D3","C3","Q3","E3","K3","M3","F3","P3","S3","T3","W3","Y
3","V3","G3","H3","I3","L3",

        "A4","R4","N4","D4","C4","Q4","E4","K4","M4","F4","P4","S4","T4","W4","Y
4","V4","G4","H4","I4","L4"};
int no_class=4;

void main()
{
```

```
int hth=0,lz=0,zf=0,hb=0,count[4]={0,0,0,0},start_state=0,flag=1;

long double start_prob[4],prev_prob,cur_prob,*temp_prob;

char *seq,*temp,*prev_char,*tpm_infile,*hidden_state;
FILE*fp,*fp1,*fp2;
int prev=0,cur=0,state=0,k=0,z=0,x=0,y=0;

tpm_infile=(char*)malloc(20*sizeof(char));
hidden_state=(char*)malloc(15000*sizeof(char));
temp_prob=(long double*)malloc(no_class*sizeof(long double));


printf("enter the probability for startin in HTH sequences: ");
scanf("%lf",&start_prob[0]);
printf("enter the probability for startin in LZ sequences: ");
scanf("%lf",&start_prob[1]);
printf("enter the probability for startin in ZINC sequences: ");
scanf("%lf",&start_prob[2]);
printf("enter the probability for startin in Hbox sequences: ");
scanf("%lf",&start_prob[3]);

fp2=fopen("seq.txt","r");

if(fp2==NULL)
{
        printf("unable to open file\n");
        return;
}

seq=(char*)calloc(15000,sizeof(char));
```

```
temp=(char*)malloc(150*sizeof(char));
prev_char=(char*)malloc(2*sizeof(char));
strcpy(seq,"");
strcpy(hidden_state,"");
while(!feof(fp2))
{
        fscanf(fp2,"%s",temp);
        strcat(seq,temp);
}


long double tpm[80][80],cla[4][20][20];



//tpm=(long double**)malloc(40*sizeof(long double));
//*tpm=(long double*)malloc(40*sizeof(long double));

for(int h=0;h<4;h++)
{
        if(h==0)
        {
                strcpy(tpm_infile,"Hth.txt_tpm.txt");
        }
        else if(h==1)
        {
                strcpy(tpm_infile,"Leucine_zipper.txt_tpm.txt");
        }
        else if(h==2)
        {
                strcpy(tpm_infile,"Zinc_finger.txt_tpm.txt");
        }
        else
        {
                strcpy(tpm_infile,"Homeo_box.txt_tpm.txt");
        }
        fp=fopen(tpm_infile,"r");



        //printf("hieeeeeeeeeeeeeeeeee\n");
        if(!fp)
        {
                printf("file %s cant b opened\n",tpm_infile);
                return;
        }
        for(int l=0;l<20;l++)
        {
                for(int m=0;m<20;m++)
                {
                        //printf("2222222222222222");
                        //getch();
                        fscanf(fp,"%lf",&cla[h][l][m]);
                        //printf("%d,%lf\n",h,cla[h][l][m]);
                        //getch();
                }
        }
        //printf("hieeeeeeeeeeeeeeeeee\n");
        fclose(fp);
}
//printf("%lf\n",tpm[1][1]);
//getch();
for(int g=0;g<80;g++)
{
        for(int h=0;h<80;h++)
        {
                tpm[g][h]=0.0;
//              printf("%f \n ",tpm[g][h]);
        }
//      printf("\n");
}
//getch();

///printf("heeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee\n");
//getch();
//printf("\n\n\n\n\n%lf\n",tpm[0][0]);
//getch();
k=80/(no_class);

long double arr1[6]={0.9,0.8,0.7,0.6,0.5,0.4};
//float arr2[]={0.9,0.8,0.7,0.6,0.5,0.4};
long double p[4]={0.0,0.0,0.0,0.0};

//p=(long double*)malloc(no_class*sizeof(float));
for(int a=0;a<6;a++)
{
        p[0]=arr1[a];
        for(int b=0;b<6;b++)
        {
                p[1]=arr1[b];
                for(int c=0;c<6;c++)
                {
                        p[2]=arr1[c];
                        for(int d=0;d<6;d++)
                        {
```

```
                        p[3]=arr1[d];


//      dna=0;
        for(int gb=0;gb<4;gb++)
        {
                count[gb]=0;
        }

for(int i=0;i<80;i++)
{
        z=i/k;
        //printf("%d,,,,,,hieeeeeeeeeeeeeeeeee\n",z);
        //getch();

        for(int j=0;j<80;j++)
        {
                if(i%20&&j%20)
                {
                        //getch();
                }
                if(i<20*(z+1)&&j<20*(z+1)&&i>=20*z&&j>=20*z)
                {

                        //printf("%d %d\n",(20-i),(20-j));

                        //if(((20-i)>=0)&&((20-j)>=0)&&((20-i)<20)&&((20-j)<20))
                        //{


                            tpm[i][j]=(cla[z][abs(i-(20*z))][abs(j-(20*z))]*p[z]);
```

```
                                //printf("ur here..\n");
                                printf("%d,%d %d,%lf          \n",z,(i),(j),tpm[i][j]);
                                //getch();
                        //}
                }
                if((i<=20*(z+1)&&j>=20*(z+1))||(j<20*z&&i<=20*(z+1)))
                //{

                //else
                {
                        tpm[i][j]=(1-p[z])/60;
                        printf("%d,%d %d,%lf          \n",z,(i),(j),tpm[i][j]);
                //      getch();
                }
                //}

        }
}

        //getch();

for(i=0;i<strlen(seq);i++)
{
        //printf("%lf        %lf\n",(q1),p1);
        start_state=(int)max(start_prob);


if(i==0)
{

        prev_prob=start_prob[start_state];
        printf("%d",to_int(seq[i]));
        prev=(start_state*20)+(long double)(to_int(seq[i]));
        prev_char=to_char(prev);
        strcat(hidden_state,prev_char);
        printf("%d \n",prev);
        //getch();
        //puts(prev_char);

}
else
{
        //cur=(1*20)+to_int(seq[i]);
        //prev_char=to_char(prev);

        for(int d=0;d<no_class;d++)
        {
                cur=((d)*20+(long double)to_int(seq[i]));
                printf("%d %d %lf %lf\n",prev,cur,tpm[prev][cur],prev_prob);
                temp_prob[d]=((prev_prob*tpm[prev][cur])*100);
                if(temp_prob[d]>1000)
                {
                        flag=0;
                        prev_prob/=1000;
                        for(int h=0;h<=d;h++)
                        {
                                temp_prob[h]/=1000;
                                //printf("%lf    %lf\n",temp_prob[d],temp_prob[d]/1000);
                                //getch();
                        }
                //      getch();
                //      temp_prob[d]=((prev_prob*tpm[prev][cur])/1000);
```

```
            }
            printf("%d      %lf \n",d,temp_prob[d]);
            //getch();
        }
        printf("oyeeeeeeeeeeee\n");

        state=(int)max(temp_prob);



        printf("state=====%d\n",state);
        //getch();
        cur=((state)*20+(long double)to_int(seq[i]));
        prev=cur;
        count[state]++;

        //printf("%d\n",state);
        strcpy(prev_char,to_char(prev));
        strcat(hidden_state,prev_char);
        //printf("%d \n",prev);
        if(!flag)
        {
            prev_prob=temp_prob[state]/10000;
            printf("prev===%f",prev_prob);
            //getch();
        }
        prev_prob=temp_prob[state];
        printf("%lf \n",prev_prob);
//      getch();
        //puts(prev_char);
    }
    }
```

```
printf("hidden state=%s\n",hidden_state);
strcpy(hidden_state,"");
printf("p1========%f,,,,,,,p2======%f\np3=======%f,,,,,p4====%f\nstate1 ---
%d,,,state2-----%d\nstate3----%d,,,state4---
%d",p[0],p[1],p[2],p[3],count[0],count[1],count[2],count[3]);
getch();
//}
//}
}
}
}
}
}


long double max(long double* arr)
{
        long double temp=arr[0],ret_value=0;
        int store[4];
        int f=0,n=0;

        for(int i=1;i<no_class;i++)
        {
                if(arr[i]>temp)
                {
                        printf("%d     %f\n",i,arr[i]);
                        temp=arr[i];
                        ret_value=i;
                        f=i;
                }
```

```
}
for(int j=0,k=0;j<no_class;j++)
{

        if(arr[j]==arr[f]&&f!=j)
        {
                store[k]=j;
                k++;
        }
}
if(k)
{
        printf("areyy babababababbaa   %d",k);
        int a=rand()%(k);
        printf("uuuuuuuuuu........%d",a);
        //printf("%f\n",store[rand()%(n+1)]);
        return store[a];
/*      if(ret_value==0.0||ret_value==1.0||ret_value==2.0||ret_value==3.0)

                return ret_value;
        else
        {
                printf("errorrrrrrrrrr");
                printf("%f\n",ret_value);
                getch();
        }
        //printf("fck uuu %d   %f\n",n,ret_value);
        //getch();*/
}
```

```
        //getch();
//      return (long double)rand()%store[j];

        return ret_value;
}

char* to_char(int a)
{
        char *b;
        b=(char*)malloc(sizeof(char));
        strcpy(b,states[a]);
        //printf("fck uuuuu %d       %s\n",a,b);
        //getch();
        return b;
}

int to_int(char a)
{
        //printf("in module\n");
        int ret_value;
        for(int i=0;i<20;i++)
        {
                if(s[i]==a)
                {
                        ret_value=i;
                }
        }
        //printf("%d \n",ret_value);
        return ret_value;
}
```

Appendix-IV

C-program for calculating trasition probability matrix

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<io.h>
#include<fstream.h>
char seq[21]={'A','R','N','D','C','Q','E','K','M','F','P','S','T','W','Y','V','G','H','I','L'};
float frequency[20][20];

void calculate(char *);
int to_int(char);

int main()
{

  FILE *in,*fp1;
  char names[6][50]={"Non-
Dnabinding_proteins.txt","dna_binding.txt","Hth.txt","Leucine_zipper.txt",
           "Homeo_box.txt","Zinc_finger.txt"};
  char temp[50]={""},*temp1;
  temp1=(char*)malloc(50*sizeof(char));
  for(int i=0;i<6;i++)
  {
                 strcpy(temp,names[i]);
          in=fopen(names[i],"r");
          temp1=strcat(temp,"_tpm.txt");
                 fp1=fopen(temp1,"w");
```

```
    //in=fopen(in,"r");
    //fp1=fopen("dna_binding_tpm.txt","w");
            if(in==NULL||fp1==NULL)
    {
              printf("file cant b opened\n");
    }
            else
            {
                        printf("file opened\n");
                        //getch();
            }

  char *ptr,*temp;
  int i=0;

  ptr=(char*)malloc(14000*sizeof(char));
  temp=(char*)malloc(14000*sizeof(char));
  strcpy(ptr,"");

  while(!feof(in))
  {

            fscanf(in,"%s",temp);


            if(*(temp)=='>')
            {
                        if(i)
                        {
                        calculate(ptr);
                        }
```

```
                i++;
                printf("%s\n",ptr);
                //getch();
                strcpy(ptr,"");

        }
        else
        {
                strcat(ptr,temp);
        }
}
for(int u=0;u<20;u++)
{
        int a=0;
        float sum=0;
for(int l=0;l<20;l++)
{

        a=sum;
        for(int m=0;m<20;m++)
        {
                if(l==0)
                {
                        sum+=frequency[u][m];
                        //printf("%f",sum);
                }
        }
        frequency[u][l]/=sum;
        //fprintf(fp1,"%f ",frequency[u][l]);
}
//fprintf(fp1,"\n");
```

```
}
//getch();
for(int l=0;l<20;l++)
{
        for(int m=0;m<20;m++)
        {
                fprintf(fp1,"%f ",frequency[l][m]);
        }
        fprintf(fp1,"\n");
}

fclose(in);
fclose(fp1);
  }
}

void calculate(char *ptr)
{

//int *freq;
printf("next sequence\n");
for(int i=0;i<strlen(ptr);i++)
{
        frequency[to_int(ptr[i])][to_int(ptr[i+1])]++;
}
}
int to_int(char a)
{
        int ret_value=0;
        for(int i=0;i<20;i++)
        {
```

```
        if(a==seq[i])
        {
                ret_value=i;
        }
    }
    return ret_value;
}
```

Appendix-V

C-program for Normalising data for input into ANN

```
#include<stdio.h>
#include<string.h>
#include<conio.h>

void main()
{
    FILE*fp,*fp1;
    int n=0,count=0;
    float a,x,max=0.0,min=0.0;
    double b;
    char c;
    fp=fopen("train_in.txt","r");
    if(fp==NULL)
    {
        printf("unable to open file\n");
    }

    fp1=fopen("train_normalise.txt","w");
    if(fp1==NULL)
    {
        printf("unable to open write file\n");
```

```
}
while(!feof(fp))
{
        count++;
        fscanf(fp,"%f",&a);
        fscanf(fp,"%c",&c);
        if(c=='\n')
        {
                continue;
        }

        if(max<a)
        {
                max=a;
                printf("max\n");

        }
        if(min>a)
        {
                min=a;
                printf("min\n");
        }
        //getch();
        //printf("%d--------------------\n",count);
}
fclose(fp);
//printf("---------------------\n");
fp=fopen("train_in.txt","r");
while(!feof(fp))
{
        fscanf(fp,"%f",&a);
```

```
				x=(float)(a-min)/(max-min);

				fprintf(fp1,"%f ",x);
				fscanf(fp,"%c",&c);
				if(c=='\n')
					fprintf(fp1,"\n");
				/*else
				{
					//b=(double)a/255;
					//fprintf(fp1,"%lf ",b);

				}
				if((n+1)%3==0)
				{
					fprintf(fp1,"\n");
				}
				n++;*/
			}
			fclose(fp);
			fclose(fp1);
		}
```