Fall 1997

Testing statistical significance in sequence classification algorithms

Tom Tien-Hua Shih
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

TESTING STATISTICAL SIGNIFICANCE IN SEQUENCE CLASSIFICATION ALGORITHMS

by

Tom Tien-Hua Shih

Multiple sequence alignment has proven to be a successful method of representing and organizing of protein sequence data. It is crucial to medical researches on the structure and function of proteins.

There have been numerous tools published on how to abstract meaningful relationship from an unknown sequence and a set of known sequences. One study used a method for discovering active motifs in a set of related protein sequences. These are meaningful knowledge abstracted from the known protein database since most protein families are characterized by multiple local motifs. Another study abstracts knowledge regarding the input sequence using a preconstructed algorithm from a set of sequences.

Most of these studies of classification processes use statistically optimized heuristics to enhance their accompanying algorithms. Therefore, these algorithms can be analyzed for statistical significance using Baysian Theorems.

TESTING STATISTICAL SIGNIFICANCE IN SEQUENCE CLASSIFICATION ALGORITHMS

by
Tom Tien-Hua Shih

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

October 1997

APPROVAL PAGE

TESTING STATISTICAL SIGNIFICANCE IN SEQUENCE CLASSIFICATION ALGORITHMS

Tom Tien-Hua Shih

Dr. Jason T. L. Wang, Thesis Advisor — Date
Associate Professor of Computer and Information Science, NJIT

Dr. James McHugh, Committee Member — Date
Professor of Computer and Information Science, NJIT

Dr. Peter A. Ng, Committee Member — Date
Professor and Chairman of Computer and Information Science, NJIT

BIOGRAPHICAL SKETCH

Author: Tom Tien-Hua Shih

Degree: Master of Science

Date: October 1997

Undergraduate and Graduate Education:

- Master of Science in Computer Science
  New Jersey Institute of Technology, Newark, NJ, 1997

- Master of Science in Education
  Queens College of City University of New York, NY, 1995

- Bachelor of Science in Engineering
  The Cooper Union for the Advancement of Science and Art, NY, 1992

Major: Computer Science

To my beloved parents

ACKNOWLEDGMENT

I wound like to express my deepest appreciation to Dr. Jason Wang, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. James McHugh and Dr. Peter A. Ng for participating in my committee.

Many of my fellow graduate students in the Data and Knowledge Engineering Lab are deserving of recognition for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Multiple sequence alignment has proven to be a successful method of representing and organizing of protein sequence data. It is crucial to medical researches on the structure and function of proteins. It is becoming more probable that a search for similarity will succeed in detecting a relationship between any newly determined sequence and one or more known sequences. Medical researchers have been able to abstract important clues regarding gene or protein function. The question remains that when to determine the similarity between the testing sequence and the known sequence is too weak for a potentially meaningful relationship.

There have been numerous studies published on how to abstract meaningful relationship from an unknown sequence. Detection of distant relationships can be aided by the presence of multiple members of a single protein family. In fact, most studies have utilized this principle in constructing their sequence databases. Usually, a sequence is divided into numerous blocks, where each block is a local multiple alignment from a group of related proteins [3]. A query sequence is searched against an established database of blocks by calculating a position-specific score. Different studies use different algorithms in calculating the scores. For example, multiple sequence alignment has been instrumental in the study of molecular evolution [7].

These studies can often provide specific information on local relationships useful for identifying sequence motifs. These are meaningful knowledge abstracted from the

known protein database since most protein families are characterized by multiple local motifs.

There have been many tools for discovering motifs in sets of protein or DNA sequences. These tools try to abstract knowledge regarding the input sequence using a preconstructed algorithm from a set of sequences. A desirable tool would have the ability to find motifs in a totally unsupervised operation, and it also can allow itself to benefit when prior knowledge is available. In most of the studies mathematical theorems have been proven to be very helpful in protein modeling, structure prediction and bioengineering.

Another important problem in computational biology is DNA sequence classification. DNA sequence classification is the activity of determining whether or not an unbiased sequence belongs to an known class. Many algorithms have been developed in constructing classifiers from a library of labeled sequences. These techniques have been categorized into three classes:

(1) consensus search - this approach takes a collection of sequences of a class C and generates another sequence which is used to identify sequences in uncharacterized DNA.

(2) Inductive learning/neural networks - this approach takes a set of sequences of the class C and a set of sequences not in C. Using learning techniques, it derives a rule that determines whether the unlabeled sequence belongs to C or not.

(3) sequence alignment - this approach aligns the unlabeled sequence S with members of C using an existing tool and assigns S to C if the resulting alignment score is high.

There are many studies using probability models in detecting motifs. Many statistically based algorithms are used. It is the purpose of this thesis to analyze some of the studies on DNA classifications based on known statistical models.

CHAPTER 2

ACTIVE MOTIFS AND PROTEIN SEQUENCES

2.1 Overview

In the study by Wang et al. [14] a method for discovering active motifs in a set of related protein sequences is described. This method is a two-step process:

(1) find candidate motifs in a small sample of the sequences;

(2) test whether these motifs are approximately present in all the sequences.

This approach was used to discover commonly active motifs in a set of related protein sequences [7]. The structures of the motifs being discovered are regular expressions of the form *S1*S2*... The S1, S2, ... are segments of a sequence or subsequences made up of consecutive letters. Following is an example.

Consider the set of three sequences in Figure 1.

S_1: FEVHYDPMISEDENRLWYPEVPSVG
S_2: GWWRADVNHPTYDPAPFRMKENRAR
S_3: WRYDPMNSEDKTMTITLVGWNRLCD

Fig. 1. The set S of three sequences

Suppose only exactly coinciding segments occurring in at least two sequences and having lengths greater than 3 are considered active. Then S contains one active motif:

$$*S_1[5,8]* = *YDPM* \text{ and } S_3[3,6] = *YDPM*$$

where V[x,y] is a segment of a sequence V from the xth to the yth letter inclusively.

To discover such active motifs in a set of sequences, two optimization heuristics based on statistical estimation and pattern matching techniques were developed. By combining the discovered motifs with an existing fingerprint technique, a protein

classifier is developed. A number of techniques may be used to locate the active motifs. These techniques are based on either one of two approaches: (i) a multiple alignment of the sequence as a whole, and (ii) a search for local similar segments in the set. Both approaches have their own merits and deficiencies. The method used by Wang et al. [14] to find active motifs is implemented without prior knowledge of their structures, positions, or occurrence. This method is used to find active motifs composed of nonconsecutive segments separated by variable length without prior knowledge of their structures, positions, or occurrence frequency.

2.2 The DISCOVER and CLASSIFY Systems

DISCOVER takes a set of related proteins and produces a collection of active motifs in the set. CLASSIFY accepts a query protein and displays a PROSITE group to which the protein should belong. These systems can be executed either manually using user-specified parameters or using those determined by the systems.

There are other methods published for classifying proteins. One common way is via profile analysis [10, 12]. In Wang's study [14] the most frequently occurring segments and fingerprints are used as the selection criteria. In profile analysis profiles are obtained by global multiple segments. Each profile is a position-specific scoring table. Each table is created by aligning a group of related sequences. When determining the relevance of the query sequence to a group, one compares the sequence to all the profiles in the database.

Another technique for detecting distantly related sequence is via testing patterns [9]. Given patterns are generated by clustering the pairwise similarity scores among a set of related sequences to build a binary tree. The tree is reduced in a stepwise manner by progressively replacing a node. The node is replaced by connecting the two most similar termini by one common pattern until only a single common root pattern remains.

2.3 Discovery Algorithm

The algorithm consists of two phases: (1) find candidate segments among a small sample of the sequences; (2) combine the segments to form candidate motifs and evaluate the activity of the motifs in all samples to determine which motifs satisfy the specified requirements.

Phase one can be divided into two stages. In stage one, a generalized suffix tree for the sample of sequences is constructed. A suffix tree is a tree-like data structure that represents a string by collapsing a series of nodes having one child to a single node whose parent edge is associated with a string. A generalized suffix tree is an extension of the suffix tree, designed for representing a set of strings. Each suffix of a string is represented by a leaf in the generalized suffix tree. In stage two, the generalized suffix tree is traversed to fine all the segments that satisfy the minimum length. If the pattern specified by the user has the form $*X*$, then the minimum length is simply determined by the specified pattern. If the pattern specified by the user has the form $*X_1*X_2*$, all the segments V_1, V_2 are found, where at least one of the V_i is larger or equal to half of the specified length.

Phase two also has two stages. In stage one, the activities of the candidate motifs are evaluated and ranked from highest to lowest according to their occurrence numbers on the sample with respect to the mutation sequences. In stage two the most likely candidate motifs found in stage one are evaluated with the entire set. Simple sampling without replacement was used to select sample sequences from the set to achieve an optimization of selecting the comparison candidates.

2.4 Classification Algorithm

A training sample was selected from all known groups. 70% of the sequences in each group were selected at random to serve as a training sample. The training sample was processed as follows:

Find 50 characteristic motifs from the training sample of each group. The motifs are the length 4 segments having the highest occurrence numbers with zero mutations. When there are ties for occurrence numbers with respect to zero mutations, we break the ties by considering occurrence numbers with respect to one mutation. The training sequences are then hashed using the gapped fingerprint technique.

When classifying a query sequence T, T is compared with all the characteristic motifs. After comparison, each group obtains a raw score. The raw score equals the sum of the weights of the group's characteristic motifs occurring in T. The raw score for a group is normalized by dividing it by the total weight of all the characteristic motifs in that group and multiplying by 100. The highest-scoring group is then displayed as the

result of the classification provided that its score is greater than an experimentally determined threshold.

In the second phase, T is hashed by using the same hash function as the one used for the training sequences. The group containing sequences with the highest vote is displayed as the result of the classification.

CHAPTER 3

THE DISCOVERY ALGORITHM AND BAYSIAN THEOREMS

3.1 A Baysian Form of Classification Tree

The classification problem can be approached by building a binary decision tree according to some splitting rule based on some predicting variables. The partitioning is repeated until a node is reached for which no split is feasible. The splitting is stopped and this node becomes a terminal node. Prediction is determined by terminal nodes and takes the form of a class level in classification problems.

A tree T has a root node whose daughters can be divided into terminal nodes, and split nodes. The branch of T stems from node t includes t itself and all its daughters. The tree is grown as following the rules at each node.

(1) Examine every allowable split on each predicting variable.

(2) Select and execute the best splits.

(3) Stop splitting on a node when some stopping rule is satisfied.

Suppose the data consists of vector variables $X = (X_1, X_2, \cdots, X_m)$, with fixed dimensionality m. For ordered variables X_i the questions in step 1 are: is $X_i > c$ for all c in the range of X_i. Those cases in t satisfying the inequality go to the left descendant node and those not satisfying to the right descendant. One of popular criteria for rule two are Least Absolute Deviations. Let $\mu =$ ave. of $X_1, X_2, \cdots, X_m$. Then c is defined as the $\min(\mu - X_1, \mu - X_2, \cdots, \mu - X_m)$. Most applications use the minimum deviation as a selection criteria for sorting the nodes.

The tree tends to grow too big and to have too few data points in each terminal node because of rule three. To overcome this problem, trees are recursively pruned. The pruned subtree is constrained to have more than some minimum number of data points in each terminal node.

CART could be used to creat another way of classifying the protein sequence. Given a set of attributes, different subsets of values within those attributes can be grouped to form homogeneous populations. For example, if we have a group of input sequences and two groups of known sequences, A and B. We could devise an algorithm using attributes from A and B to classify the input sequences into group A or B. The unknown sequence may not belong to either group A or B. Given a set of protein structures, an classification can be set up to find their interior motifs using CART.

3.2 Algorithm for Classifier I

Given a set of base sequences, Classifier I first searches for active segments that approximately occurring the majority of sequences in the set of base sequences. A segment is defined as a substring made up of consecutive nucleotides of a sequence. Other factors considered are the length, occurrence number, and mutation of the segment. A mutation is defined as a mismatch, insertion or deletion of a nucleotide when matching the segment with a base sequence.

For example, consider the three base sequences in Fig. 2. Suppose the segment to be sought has length greater 6 (i.e., it contains more than 6 nucleotides), occurrence number 3 (i.e., it matches all three base sequences) and mutation 1 (i.e. one mismatch,

insertion or deletion of a nucleotide is allowed when matching the segment with a base sequence). Then GCCGGGC and GCCAGGC underlined in Figure 2 are two qualified segments.

GGAGAGGCCGGGCGTGTGCCGGTAC
GGCCAGGCGGCAGATCTTGACCAGG
TGTAATCAGAGCGCCAGGCAAACAT

Fig. 2. Three base sequences

The found segments may share a large common portion among them. A gluing procedure is used to combine two segments into a longer one. In gluing two segments together, the procedure aligns their common portion as much as possible.

To classify an unlabeled sequence, Classifier I preprocesses the base sequences in B by generating a set of representative patterns. The classifier first sorts the discovered active segments according to their occurrence numbers in B. This sorting indicates the order in which the segments are examined. For a segment at the i^{th} position in the sorted list, the classifier looks at elements below it in the list to see which element can be glued together with it. If it is found that the j^{th} segment qualifies where i < j, then the newly glued segment is placed at the i^{th} position and the original two segments are removed from the sorted list. The newly glued segment is used to represent the original two segments in the list. Figure 3 summarizes the algorithm.

Input: A sorted list L of active segments discovered from the set B of base sequences.
Output: A set R of representative patterns of the base sequences.
$i := 1$;
repeat
    /* Let the segment placed at the i^{th} position in L be S_1. */
    if there exists a segment which can be glued with S_1 and
    whose position in L is lower than i then
      begin
      /* Let S_2 be the first such segment and its position in L be j, $i < j$. */

```
        glue S1 and S2 and call the result S3;
        remove S1 and S2 from L;
        place S3 at the ith position in L;
      end
    else
      i := i + 1;
until L can not be shrunk further and i = |L| + 1;
```

Fig. 3. Algorithm Gluing

3.3 Summary of Statistical Theorems

This section will try to summarize general statistical theorems to reach a theoretical conclusion on how to calculate the statistical significance of Discovery algorithms.

(1) Definition of Conditional Probability

Let A and B be events with P(B) > 0. The conditional probability of the occurrence of A, given the occurrence of B, is defined to be the quantity :

$$P(A|B) = \frac{P(A)}{P(AB)}$$

(2) Bayes' Theorems

a. P(A'|B) = 1 - P(A|B)

where P(A') is P(event A not happen)

b. P(AB) = P(A|B)P(B)

(3) Hypothesis Testing

A statistical hypothesis is a hypothesis about a parameter of a distribution. One hypothesis is the null hypothesis and label it H_0. The other hypothesis is the alternative hypothesis and label it H_1. There are two types of errors involved:

Type I error: A type I error is the error of rejecting the null hypothesis when the null hypothesis is true.

Type II error: A type II error is the error of accepting the null hypothesis when the null hypothesis is false.

For example, we want to test a coin is a regular dime or rare dime. Suppose P_0, P_1, P are probabilities of heads using the rare dime, an ordinary dime and the testing dime. There are two possible hypotheses concerning P. Either $P = P_0 = 0.75$ or $P = P_1 = 0.5$. The first hypothesis is called the null hypothesis and the second hypothesis is called the alternative hypothesis. A type I error is the error of rejecting the testing dime as a rare dime while it is indeed a rare dime. A type II error is the error of accepting the testing dime as a rare dime while it is indeed a regular dime.

(4) Critical region

Suppose C is the subset of the sample space which, by a prescribed statistical test, leads to the rejection of the hypothesis under consideration. Then C is called the critical region of the test.

(5) Power function

The power function of a test of a statistical null hypothesis H_0 against an alternative hypothesis H_1 is the function, that defined for all distributions under consideration, which yields the probability that the sample point falls in the critical region C of the test. A power function is a function which yields the probability of rejecting the hypothesis under consideration. The value of the power function at a parameter point is called the power of the test at that point.

(6) Significance Level

Let H_0 denote a hypothesis which is to be tested against an alternative hypothesis H_1 in accordance with a prescribed test. The significance level of the test is the maximum value of the power function of the test when H_0 is true.

3.4 Statistical Significance on the Discover Systems

Let H be the hypothesis that a given protein sequence contains these data motifs or the unknown testing sequence is classified into family F.

Let D be the event that a candidate motif belongs to a protein sequence or the unknown testing sequence.

Now we can define Confidence Level of P(H|D) as

$$\frac{P(H|D)}{1-P(H|D)}.$$

The way to calculate P(H|D) is discussed as follows. Given is D (the unknown testing sequence T); the hypothesis H is that T is classified into family F. First, we observe that calculation of P(H|D) is algorithm dependent. Now we will divide the calculation into two parts.

(1)1-family classification algorithm

Given is a known family F. Call F training data. First discover b characteristic motifs from F. Now, suppose we are given an unknown testing sequence T.

If the testing sequence T contains $\geq$ c (out of b) characteristic motifs, then classify T into family F. Otherwise, we can classify T as non-F. We have the following constants:

b = number of characteristic motifs chosen for family F

d = length of characteristic motifs

Thus p = the probability for a randomly generated subsequence to be a characteristic motif. It is obvious that

$$p = \frac{b}{20^d}.$$

Now p^r = the probability for r characteristic motifs to appear in T (for example, T contains these r motifs). Let n = length of the testing sequence T.

Then the probability for the other portion of T not containing any characteristic motif is

$$(1\text{-}p)^{n-r}.$$

So the probability for a given unknown testing sequence T to be classified into family F is P(H|D):

$$\frac{\sum_{r=c}^{r=b} P^r(1\text{-}P)^{n-r}}{\sum_{r=0}^{r=b} P^r(1\text{-}P)^{n-r}}$$

Now we can use this formula to evaluate the theorectical confidence level of the Discovery algorithm.

(2) k-family classification algorithm

Suppose we are given k families. Each family has the same number, namely b, of motifs. The algorithm classifies T into the family with the most characteristic motifs appearing

in T. So, let h = max (j_1, ..., j_k) where j_i is the number of motifs of family i that appear in T. We can derive a similar formula for the probability for a given unknown testing sequence T to be classified into family F. Therefore, P(H|D) =

$$\frac{\sum_{r=h}^{r=b} P^r(1-P)^{n-r}}{\sum_{r=0}^{r=b} P^r(1-P)^{n-r}}$$

CHAPTER 4

THE FINGERPRINT ALGORITHM AND BAYSIAN THEOREMS

4.1 DNA Sequence Classification

There have been two new techniques for DNA sequence classification developed. The first technique used a computational tool to find active segments from a set of sequences. The second technique generated and matched gapped fingerprints of the sequences. The two proposed classifiers differ in their ways of processing the base sequences and calculating scores for the training sequences. The standard set includes positive sequences as well as negative sequences.

The approach works by first randomly selecting a set B of sequences of the class C, referred to as base data, Then another set of sequences of C is taken, referred to as positive training data. For each positive training sequence, a score is calculated with respect to the base sequences. The minimum score thus obtained is called the positive lower bound, denoted L_p. Next, a set of sequences not in C is taken, referred to as negative training set. For each negative training sequence, a score with respect to the base sequences is calculated. The maximum score obtained is called the negative upper bound, denoted U_n. Let $B_{high} = \max (L_p, U_n)$ and Blow = (L_p, U_n). When classifying the unlabeled sequence S, we calculate S's score with respect to the base sequences, denoted c. If $c \geq B_{high}$, then S is classified to be a member of C. If $c \leq B_{low}$, then S is classified not to be a member of C. If $B_{low} < c < B_{high}$, then S cannot be classified.

4.2 Algorithm for Classifier II

Classifier II uses a hash-based fingerprint technique to calculate the score of a sequence. Given a set of base sequences, their fingerprints are stored into a number of fingerprint files. Let S be a sequence in the set of base sequences. Every segment of length n from S is generated to form gapped fingerprints. The set of all generated segments from S is denoted Sseg. Each fingerprint is a substring of Sseg that begins with the segment's first nucleotide.

For each fingerprint f, a hash function h_k is used to hash f into a fingerprint file. In each fingerprint file, f is associated with a pair of integers (x,y). This pair serves as the position marker for f, where x indicates that f is generated from a segment of the xth sequence in the set of base sequences and y means that the first nucleotide of f occurs at the yth position in the sequence.

When calculating the score of a training sequence S, the sequence is segmented in the same way as the base sequences and fingerprints are generated from the resulting segments. The fingerprints then are hashed, using the same hash functions as for the base sequences. When a match between S's fingerprint and a base sequence's fingerprint occurs, one score is given to an appropriate position in the base sequence. Figure 4 summarizes the algorithm.

```
Input: A sequence S, a set B of base sequences and B's fingerprint files.
Output: A histogram of scores on the base sequences in B.
/* Let F contain all fingerprints generated from S. */
for each fingerprint f in F do
  begin
      /* Let the length of f be k. */
      hash f using h_k and probe into the fingerprint file F_k;
      for each match between f and a fingerprint f in F_k do
        begin
```

```
            /* Let the position marker associated with f be (I,q). */
            /* Suppose the first nucleotide of f occurs at the p^th position in S. */
            add one score to the position q - p + 1 in the ith base sequence in B;
        end;
end;
```

Fig. 4. Algorithm Scoring

4.3 Statistical Significance of the Fingerprint Algorithm

In general the systems regarding DNA sequence classification are very complicated to allow realistic models to be evaluated. This system can be studied by means of simulation. In regard to the Classifier II algorithm, the statistical significance of the algorithm is dependent on the way hash functions are devised.

The actual confidence level can be implemented through a simulation modeling of the standard set. However, the confidence level depends on how the standard sequences including positive as well as negative sets are related. Furthermore, unbiased input data needs to be generated to estimate the desired characteristics of the model.

The statistical reliability of the fingerprint algorithm could also be analyzed using the Baysian technique. The general rule is to use Confidence(O|I) = P(O|I)/(1- P(O|I)) where P(O|I) stands for the probability of the output sequence belonging to the set of standard sequences, given the input set.

Due to the complexity of the fingerprint algorithm, the threshold scores are used to approximate the confidence level of the algorithm. By adding the scored P(O|I) and P(O'|I), we can obtain an approximation to the confidence level. Several programs running the fingerprint algorithm have been designed. By running these programs, a

sample of ten scores have been obtained. As can be seen from the following table, the confidence level can reach 96%.

Table 1. Selected Threshold Scores of the Fingerprint Algorithm

	Positive U bound	Positive L bound	Negative U bound	Negative L bound	High Score	Low Score	P(O|I)	P(O'|I)	Confidence
	79.66	7.49	6.27	1.27	7.49	6.27	92.51%	6.27%	98.78%
	87.59	12.59	6.27	1.50	12.59	6.27	87.41%	6.27%	93.68%
	87.59	9.94	5.62	1.67	9.94	5.62	90.06%	5.62%	95.68%
	64.96	11.50	6.27	1.67	11.5	6.27	88.50%	6.27%	94.77%
	88.66	8.14	8.43	1.67	8.43	8.14	91.57%	8.14%	99.71%
	62.32	8.46	7.23	1.67	8.46	7.23	91.54%	7.23%	98.77%
	53.62	13.71	7.23	1.50	13.71	7.23	86.29%	7.23%	93.52%
	67.88	10.62	6.02	1.50	10.62	6.02	89.38%	6.02%	95.40%
	51.74	9.93	6.91	1.50	9.93	6.91	90.07%	6.91%	96.98%
	81.72	10.81	6.45	1.50	10.81	6.45	89.19%	6.45%	95.64%
Ave.					10.35	6.64	89.65%	6.64%	96.29%

4.4 Complementary Problems

We have analyzed both the Discovery algorithm and the Fingerprint algorithm. There are other tools developed for protein classifications using different approaches. How do we synchronize all these different approaches?

Suppose we have two approaches, A and B. A and B are complementary if using both A and B can produce more effective results than using A or B alone. When doing complementary analysis, one important factor that needs to be considered is the randomness factor. We should only consider the combinations of approaches which are more effective than the given approach combined with an random algorithm.

CHAPTER 5

CONCLUSION

There have been numerous tools published on how to abstract meaningful relationship from an unknown sequence and a set of known sequences. One method is to discover active motifs in a set of related protein sequences. Another method is to abstract knowledge regarding the input sequence using a preconstructed algorithm from a set of sequences. These are meaningful knowledge abstracted from the known protein database since most protein families are characterized by multiple local motifs.

Experimental results have been used to determine the performance of the algorithms. Most of these studies of classification processes use statistically optimized heuristics to enhance their accompanying algorithms. Therefore, these algorithms can be analyzed for statistical significance using Baysian Theorems.

APPENDIX A

SOURCE CODE FOR SORTING SCORES

```
#include <iostream.h>

const int MAX = 100; // maximum number of scores
const int TRUE = 1;
const int FALSE = 0;

int obtainNumData()
{
        int m;
        do { // obtain number of data points
                cout << "Enter number of data points [2 to "
                        << MAX << "] : ";
                cin >> m;
                cout << "\n";
                } while (m < 2|| m > MAX);
                return m;
}

void inputArray(float inputArr[], int n)
{
        //prompt user for data
        for (int i= 0; i < n; i++) {
                cout << "arr[" << i << "] : ";
                cout << inputArr[i] << " ";
        }
        cout << "\n";
}

void showArray(float inputArr[], int n)
{
        //prompt user for data
        for (int i= 0; i < n; i++) {
                cout.width(5);
                cout << inputArr[i] << " ";
        }
        cout << "\n";
}

void sortArray(float inputArr[], int n)
{
        int offset, inOrder;
```

```
        float temp;

        offset = n;
        do {
                offset = (8 * offset) /11;
                offset = (offset == 0) ? 1 : offset;
                inOrder = TRUE;
                for ( int i = 0, j = offset; i < (n - offset); i++, j++){
                        if (inputArr[i] > inputArr[j]){
                                inOrder = FALSE;
                                temp = inputArr[i];
                                inputArr[i] = inputArr[j];
                                inputArr[j] = temp;
                        }
                }
        } while (!(offset == 1 && inOrder == TRUE));
}

main()
{
        float arr[MAX];
        int n;

        n = obtainNumData();
        inputArray(arr, n);
        cout << "Unordered array is:\n";
        showArray(arr, n);
        sortArray(arr, n);
        cout << "\nSorted array is:\n";
        showArray(arr, n);
        return 0;
}
```

APPENDIX B

SOURCE CODE FOR CALCULATING MEANS OF SCORES

```
#include <iostream.h>

const int MAX_COL = 10; // maximum number of columns
const int MAX_ROW = 100; // maximum number of scores

int getRows()
{
        int n;
        // get number of rows
        do {
                cout << "Enter number of rows [2 to "
                        << MAX_ROW << "] : ";
                cin >> n;
        } while (n< 2 || n > MAX_ROW);
        return n;
}

int getColumns()
{
        int n;
        // get number of columns
        do {
                cout << "Enter number of columns [1 to "
                        << MAX_COL << "] : ";
                cin >> n;
                } while (n < 1 || n > MAX_COL);
                return n;
}

void inputMatrix( double mat[][MAX_COL],
                                int rows, int columns)
{
        // get the matrix elements
        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < columns; j++) {
                        cout << "X[" << i << "][" << j << "j : ";
                        cin >> mat[i][j];
                }
                cout << "\n";
        }
```

```
}

void showColumnAverage(double mat[][MAX_COL],
                                        int rows, int columns)
{
        double sum, sumx, mean;
        sum = rows;
        // obtain the sum of each column
        for (int j = 0; j < columns; j++) {
                // initialize summations
                sumx = 0.0;
                for (int i = 0; i < rows; i++)
                        sumx += mat[i][j];
                mean = sumx / sum;
                cout << "Mean for column " << j
                        << " = " << mean << "\n";
        }
}

main()
{
        double x[MAX_ROW][MAX_COL];
        int rows, columns;
        rows = getRows();
        columns = getColumns();
        // get scores
        inputMatrix(x, rows, columns);
        // show means of scores
        showColumnAverage(x, rows, columns);
        return 0;
}
```

REFERENCES

1. L. Brieman, J. H. Friedman, R. Olshen and C. J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.

2. W. G. Cochran, *Sampling Techniques*, Wiley, New York, 1977.

3. M. Gribskov, A. D. McLachlan, and D. Eisenberg, "Profile analysis: Detection of distantly related proteins," *Proc. Natl. Acad. Sci.* USA 84:4355-4358, 1987.

4. V. R. Hogg and T. A. Craig, *Introduction to Mathematical Statistics*, The Macmillan Company, London, 1970.

5. L. C. K. Hui, in A. Apostolico, M. Crochemore, Z. Gali, and U. Manber,(ed.), "Combinatorial Pattern Matching," *Lecture Notes in Computer Science*, Spring-Verlag, New York, pp. 230-243, 1992 .

6. M. A. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1982.

7. E. C. Lawrence and F. S. Altschul, " A Gibbs Sampler for the Detection of Subtle Motifs in Mutiple Sequences," *IEEE Knowledge and Data Engineering*, March 1984.

8. A. Papoulis, *Probability, Random Variables and Stochatic Processes*, McGraw-Hill, New York, 1986.

9. R. F. Smith and T. F. Smith, *Proc. Natl. Acad. Sci.* USA, 87, pp. 118-122, 1990.

10. R. Staden, *Mech. Enzymol.*, 183, pp. 193-211, 1990.

11. P. T. Strait, *Probability and Statistics with Applications*, 2nd Edition, Harcourt Brace Jovanovich, Inc. pp. 390-394, New York, 1989.

12. W. R. Taylor, *Journal of Molecular Biology*, 188, pp. 233-258, April 1986.

13. H. C. Tijms, *Stochastic Modeling and Analysis; A Computational Approach*, John Wiley, New York, 1986.

14. T. L. J. Wang, G. T. Marr, D. Shasha, A. B. Shapiro and G.W. Chirn, "Discovering active motifs in sets of related protein sequences and using them for classification," *Nucleic Acids Research*, Vol. 22, No. 14, 1994.

15. W. R. Wolff, *Stochastic Modeling and the Theory of Queues*, Prentice Hall, New York 1989.