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

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

ABSTRACT

PATTERN DISCOVERY IN SEQUENCE DATABASES: ALGORITHMS AND APPLICATIONS TO DNA/PROTEIN CLASSIFICATION

by
Gung-Wei Chirn

Sequence databases comprise sequence data, which are linear structural descriptions of many natural entities. Approximate pattern discovery in a sequence database can lead to important conclusions or prediction of new phenomena. Traditional database technology is not suitable for accomplishing the task, and new techniques need to be developed.

In this dissertation, we propose several new techniques for discovering patterns in sequence databases. Our techniques incorporate pattern matching algorithms and novel heuristics for discovery and optimization. Experimental results of applying the techniques to both generated data and DNA/proteins show the effectiveness of the proposed techniques.

We then develop several classifiers using our pattern discovery algorithms and a previously published fingerprint technique. When we apply the classifiers to classify DNA and protein sequences, they give information that is complementary to the best classifiers available today.

PATTERN DISCOVERY IN SEQUENCE DATABASES: ALGORITHMS AND APPLICATIONS TO DNA/PROTEIN CLASSIFICATION

by
Gung-Wei Chirn

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer and Information Science

May 1996

APPROVAL PAGE

PATTERN DISCOVERY IN SEQUENCE DATABASES: ALGORITHMS AND APPLICATIONS TO DNA/PROTEIN CLASSIFICATION

Gung-Wei Chirn

Dr. Jason T. L. Wang, Dissertation Advisor — Date
Assistant Professor of Computer and Information Science Department, NJIT

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

Dr. David Nassimi, Committee Member — Date
Associate Professor of Computer and Information Science Department, NJIT

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

Dr. Wen-Syan Li, Committee Member — Date
Faculty of Management
Center for Information Management, Integration, and Connectivity, Rutger University

BIOGRAPHICAL SKETCH

Author: Gung-Wei Chirn

Degree: Doctor of Philosophy

Date: May 1996

Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1996
- Master of Science in Computer Engineering,
  National Chiao Tung University, Hsinchu, Taiwan R.O.C, 1986
- Bachelor of Science in Computer Engineering,
  National Chiao Tung University, Hsinchu, Taiwan R.O.C, 1984

Major: Computer and Information Science

Publications:

J. T. L. Wang, T. G. Marr, D. Shasha, B. A. Shapiro, G. W. Chirn and T. Y. Lee, "Complementary Classification Approaches for Protein Sequences," to appear in *Protein Engineering*, 1996.

J. T. L. Wang, G. W. Chirn and K. Zhang, "Algorithms for Approximate Graph Matching," *Information Sciences*, vol. 82, pp. 45–74, 1995.

J. T. L. Wang, T. G. Marr, D. Shasha, B. A. 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, pp. 2769–2775, 1994.

J. T. L. Wang, G. W. Chirn, T. G. Marr, B. A. Shapiro, D. Shasha and K. Zhang, "Combinatorial Pattern Discovery for Scientific Data: Some Preliminary Results," *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, pp. 115–125, May 1994.

G. J. S. Chang, J. T. L. Wang, G. W. Chirn and C. Y. Chang, "A Visualization Tool for Pattern Matching and Discovery in Scientific Databases," *Proceedings of the Eighth International Conference on Software Engineering and Knowledge Engineering*, Lake Tahoe, Nevada, June 1996.

J. T. L. Wang, G. W. Chirn, C. Y. Chang, G. J. S. Chang, A. Noriega and K. Pysniak, "An Integrated Toolkit for Pattern Matching and Pattern Discovery in Scientific, Program and Document Databases," *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering*, Rockville, Maryland, pp. 497, June 1995.

G. W. Chirn, "A Tool for Discovering Motifs in Protein Databases," *The Second Annual Graduated Study Association Student Research Conference, University of Medicine and Dentistry of New Jersey*, Newark, New Jersey, June 1995.

J. T. L. Wang, K. Zhang and G. W. Chirn, "The Approximate Graph Matching Problem," *Proceedings of the 12th International Conference on Pattern Recognition*, Jerusalem, Israel, Vol. 2, pp. 284–288, October 1994.

J. T. L. Wang, K. Zhang and G. W. Chirn, "Approximate Graph Matching Using Probabilistic Hill Climbing Algorithms," *Proceedings of the 6th IEEE International Conference on Tools with Artificial Intelligence*, New Orleans, Louisiana, pp. 390–396, November 1994.

This work is dedicated to
my family

ACKNOWLEDGMENT

I would like to thank Mr. Chia-Yo Chang and George Chang for their help on the toolkit presented in the thesis.

I also want to express my appreciation to my father, Mingliang Chin, my mother, Chihsiou Kang, my brothers, Kungcho Chin and Kungyu Chin, my wife, Wanyi Tung, and my daughter, Alice, my son, Brian. Only they continuously gave me support during my most difficult days. Without their support and encouragement, my accomplishment would not be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Sequence databases are databases comprising one dimensional data structures such as text, digital signals, proteins and DNA (deoxyribonucleic acid). Such objects are often represented as sequences in the databases. For example, a protein is represented as a sequence made from 20 amino acids, each represented as a letter (as shown in Table 1.1). A digital signal is represented by a series of 0's or 1's digits. A DNA is represented as a sequence of four nucleotides: A (adenosine), T (thymidine), C (cytidine) and G (guanine).

Table 1.1 20 amino acids.

abbreviations	amino acids
A	alanine
C	cystine
D	aspartate
E	glutamate
F	phenylalanine
G	glycine
H	histidine
I	isoleucine
K	lysine
L	leucine
M	methionine
N	asparagine
P	proline
Q	glutamine
R	arginine
S	serine
T	threonine
V	valine
W	tryptophan
Y	tyrosine

With the significant growth of sequence database sizes in recent years, it becomes increasingly important to develop new techniques for data organization

and query processing in the sequence databases. *Pattern Discovery* is a fundamental operation in the sequence databases. It attempts to discover useful patterns which can help scientists to find new properties of the databases or predict the function of a new entity. Figure 1.1 illustrates the pattern discovery operation. Given data representing real-world entities, a scientist chooses a pattern metric that seems reasonable. The pattern discovery algorithm identifies (approximately) common substructures based on that metric. The substructures are then tested against the data to see if they characterize the data in the sense of being good classifiers. The results may show the pattern metric to be irrelevant or may suggest that the patterns merit further study.

Pattern discovery in sequences is computationally expensive - sometimes requiring more than a day to complete an analysis of even a moderately sized database. The main theme of the dissertation is the study of pattern discovery in sequence-based scientific databases and the implementation of several classifiers for protein and DNA sequences. We start by investigating new techniques to discovery active patterns in a sequence database. By combining these techniques with a previously published fingerprint technique, we develop several tools for DNA and protein classification. Some experimental results are then presented, which indicate that our work concerning pattern discovery in sequences is significant.

The rest of the dissertation is organized as follows. Chapter 2 presents the algorithms and theorems for discovering active patterns in sequence databases. Chapter 3 describes the performance of the algorithms. Chapter 4 presents two DNA classification algorithms. Chapter 5 shows the experimental results. Chapter 6 describes the algorithms for discovering blocks in a protein family. Chapter 7 uses the blocks to classify proteins. Chapter 8 shows classification results. Chapter 9 describes several software packages developed from this dissertation. Chapter 10 concludes and discusses future works.

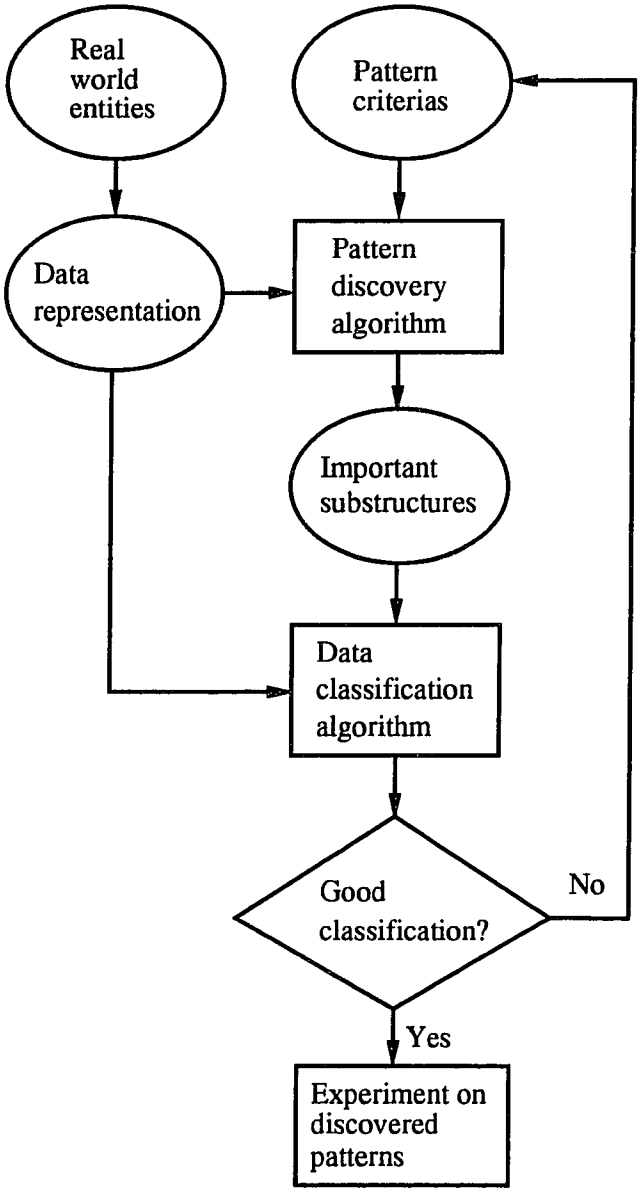

Figure 1.1 Illustrations of the pattern discovery operation.

CHAPTER 2

ALGORITHMS FOR DISCOVERING ACTIVE PATTERNS IN SEQUENCES

Combinatorial pattern discovery or *combinatorial data mining* is the activity of finding structural or topological patterns in data that can lead to important conclusions or prediction of new phenomena. The patterns may approximately characterize a set of structures in the database given a pattern metric.

In this chapter, we focus on discovering active patterns in sequences such as protein or DNA. It has been known that DNA acts like a biological computer program with some 3 billion bits long that spells out the instructions for making proteins, the basic building blocks of life. A protein is a 3D molecular structure constructed by hundreds or thousands of amino acids [20]. A simple protein example is shown in Figure 2.1. Each amino acid is represented by a dot in the example.

The most popular representation model for biologists to describe a protein is to use the sequence [20]. A protein is represented as a sequence made from 20 amino acids, each represented as a letter. Figure 2.2 shows a protein sequence with 922 amino acids.

A DNA (deoxyribonucleic acid) is a twisted double helix structure [45]. An example is shown in Figure 2.3. Each strand of the DNA double helix is a polymer consisting of four elements called *nucleotides*: A, T, C and G (the abbreviations for adenine, thymine, cytosine and guanine). The two strands of DNA are prefectly complementary: whenever there is a T on one strand, there is an A on the corresponding position on the other strand; Whenever there is a G on one strand, there is a C on the corresponding position on the other. That is, T pairs with A and G pairs with C.

From a computer scientist's viewpoint, the DNA double helix is a clever and robust information storage and transmission system. Computer scientists

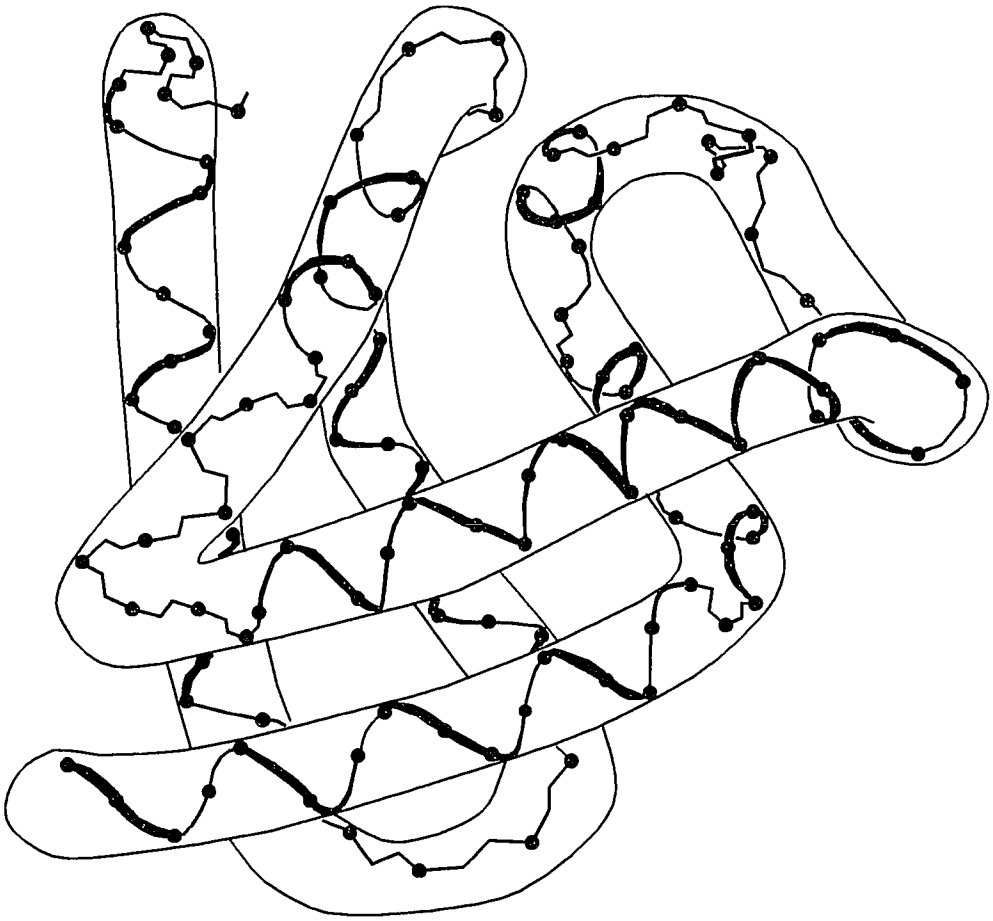

Figure 2.1 A protein structure.

MEAPGQDTEDALRRSLDPEGYEDTKGSRTSLGTMSNPLVSSVDLEAAGSR
QPSAHRDTYEGYVELHELVLDSRKDPCWMEAGRWLHLEESHEPGGAWGSH
LPSLTYHSLLELHRAFAKGVVLLDVAANSLAAVAHVLLDQAIYEGQLKPQ
HRHDVLRALLLRHKHPSEAESVWTLPAAQLQCSDGEQKDAMERALLRDQR
AVOMRELHGAGQSPSRAQLGPQLHQQLPEDTEATLVLVACEAFLEQPLLA
LVLLGAPCPDAVLAVPLPBRFVLTVLGPDSPRLSYHEIRRLAATVMADRV
FREDAYLCGGRAELLGGLIGFLEASIVLPPQEVPSEQHLHELIPLQRHAV
RRJYQHPDTVRSPGGPTATKDTGDKGQAPQDDDPLLRTRRSFGGLVRDIR
RRAPKYLSDIRHALNPQCCAAVIFIYFAALSPAITFGGLLSEKTRGMMGV
SESLLSTSVQCILFSLLSHQPLLVVGFSGPLLVFEEAFFRPCEDHGLEYI
VGOVWIGFWLITLVLLVVLCEGTVLVRYLSRYTQEIFSFLESLIFIYETF
AKNVTIFEAHPNQQSYDTIVSTEPSVPKPNTALLSLVLMATTFFLALFLR
QFWNSVFLPGKJRRLIGDUGVPISIFVMALADFFIKDTYTEKLKVPRFLE
VTAGTARGWFIIPMGSATPFPIWMMFASPVPALLVFILIFRETQITTUIV
SKNERKLVKGSTFHLDLLLIVAMGGLAALFGMPWLSATTVNTITHANCLT
VVGKSAVPGERAHIVEVKEQRLSGLLVAVLIGVSILMEPIGKYIPLAKLF
GIFLYMGVTSLFGIQLFDRILLLLMPPKYHPKEPYVTRVKTWRITSSYLT
QILVVALLWGVKVSPASLRCPFVLVLTVPLRRLLLPRIFSEIELKCLOTD
MEGPGQDTEDALRRSLDPEGYEDTKGSRTSLGTMSNPLVSDVDLEAAUSR
DAVVTFEEAEGQDVYNEVQMPS

Figure 2.2 A protein sequence.

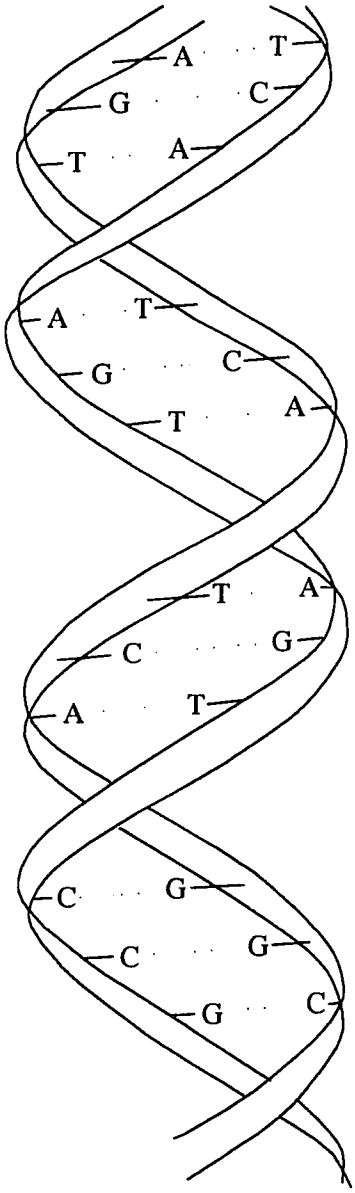

Figure 2.3 A DNA structure.

```
GGCTGGGTGCGGTGACTCATCCTGGAATCCAGCACTTTGGGAGGCGAGGC
AGGTGGATCACTGAGGTCAGCGAGTTTCGATGACCACCCTGGCCAACATA
GTAAAACCCTGTCTCTACTAAAAATTACAAAATTAGCTCAGTGTGGTGGT
AGGCGCCTGTAGTCCCAGCTACTCTGGAGGCTGAGGCAAGAGAATCACTT
GAACCTGGGAGGCAGAAGTTTCAGCAAGCTGAGACTGCACCACTGCACTT
CAGCCTGGGAGGCAGAAGTTTCAGCAAGCTGAGACTGCACCACTGCACTT
CAGCCTGGGAGACAGAGCAAGACTCCATCTCAAAACAAAAAACAAAACAA
AAAAAAGAAAAGAAATAGATGTAGTCAGA
```

Figure 2.4 A DNA sequence.

accustomed to dealing with a binary alphabet will immediately recognize that the four-letter alphabet of DNA is sufficient for encoding messages of arbitrary complexity. A DNA can be represented by a sequence of A, C, G and T letters. Figure 2.4 shows a DNA sequence composed of 379 nucleotides.

The patterns we wish to discover within a sequence database are regular expressions of the form $*X_1 * X_2 * \ldots$. The $X_1, X_2, \ldots$ are *segments* of a sequence, i.e., subsequences made up of consecutive letters, and $*$ represents a variable length don't care (VLDC). In matching the expression $*X_1 * X_2 * \ldots$ with a sequence S, the VLDCs may substitute for zero or more letters in S at zero cost.

The dissimilarity measure used in comparing two sequences is the *edit distance*, i.e., the minimum weighted number of insertions, deletions and substitutions used to transform one sequence to the other after an optimal substitution for the VLDCs [86, 95]. The edit distance is a useful measure of evolutionary distance [11, 84]. For the purpose of this work, we assume that all the edit operations have unit cost, though the techniques we propose do not depend on that cost assumption or essentially on the edit distance metric.

Example 2.1 (Matching sequences with VLDCs)

Consider the expression *TQI* and the sequence MYALTIHKR. In matching the expression with the sequence, the first asterisk would substitute for MYAL and the

S_1: YDPMIEDKEYSRLVG
S_2: RMKQLGRTYDPAVWG
S_3: YDPMNWNFEKETLVG

Figure 2.5 The set $\mathcal{S}$ of three sequences.

second asterisk would substitute for HKR. The distance is 1 (representing the cost of deleting Q). The length of the pattern *TQI* is three. □

Example 2.2 (Finding active patterns in sequences)

Consider the set $\mathcal{S}$ of three sequences in Figure 2.5.

Suppose only exactly coinciding segments occurring in at least two sequences and having lengths greater than 3 are considered as 'active.' Then $\mathcal{S}$ contains one active pattern:

$$*S_1[1,4]*=*\text{YDPM}* \iff *S_3[1,4]*=*\text{YDPM}*$$

where $V[x,y]$ is a segment of a sequence V from the xth to the yth letter inclusively. If patterns occurring in all the three sequences within one mutation are considered as active, i.e., one mismatch, insertion or deletion is allowed in matching a pattern with a sequence, then $\mathcal{S}$ contains three active patterns:

$$\begin{aligned} & *S_1[1,4]* = *\text{YDPM}* \\ \iff\ & *S_2[8,11]* = *\text{TYDP}* \\ \iff\ & *S_2[9,12]* = *\text{YDPA}* \\ \iff\ & *S_3[1,4]* = *\text{YDPM}* \end{aligned}$$

If patterns having the form $*X*Y*$ are sought with lengths greater than 7 and one mutation allowed, then S_1 and S_2 share the following four active patterns:

$$
\begin{aligned}
& *S_1[1,4] * S_1[12,15]* = *\text{YDPM} * \text{RLVG} * \\
\Longleftrightarrow \quad & *S_1[1,5] * S_1[13,15]* = *\text{YDPMI} * \text{LVG} * \\
\Longleftrightarrow \quad & *S_3[1,4] * S_3[12,15]* = *\text{YDPM} * \text{TLVG} * \\
\Longleftrightarrow \quad & *S_3[1,5] * S_3[13,15]* = *\text{YDPMN} * \text{LVG}*
\end{aligned}
$$

□

To discover such active patterns in a database of sequences, our overall strategy is first to find candidate segments among a small sample and then to combine the segments into candidate patterns and check which pattern satisfy the specified requirements.

Many techniques have been published in the literature to solve similar problems.[1] A commonly used one is based on multiple sequence alignment (see [92] for review). The technique is useful when entire sequences in the database are similar. However, when the sequences have only short regions of local similarities, this approach makes no sense. There are also techniques based on local similarity search. The techniques work effectively when similarities meet some constraints, such as they occur in a predetermined number of sequences in the database [69], they differ by mismatches, but not by insertion/deletions [5], or they are situated at almost the same distance from the start of the sequences [85]. In contrast to the above techniques, our approach can find similarities composed of nonconsecutive segments separated by variable length don't cares without prior knowledge of their structures, positions, or occurrence frequency.

[1]These problems are mostly concerned with discovering patterns made up of single segments, or multiple segments separated by fixed length don't cares.

2.1 Basic Discovery Queries

Given a database $\mathcal{D}$ of sequences, there exist various requirements on the lengths and forms of similarities to be sought. The following parameters appear to be most significant (all the parameter values are specified by the user):

- the form of patterns, in our case regular expressions of the form $*X_1 * X_2 * \ldots$.
- the minimum length of a pattern of interest *Length*, in our case the number of the non-VLDC letters.
- the distance metric, in our case edit distance with unit cost having free substitution for VLDCs (the asterisks).
- the allowed distance *Dist*.
- the minimum occurrence number *Occur* with respect to the distance and length of a chosen pattern. The occurrence number or activity of a pattern is the number of sequence in $\mathcal{D}$ matching the pattern within the distance. We say the occurrence number of a pattern P with respect to distance i and set $\mathcal{S}$, denoted $occurrence\_no^i_{\mathcal{S}}(P)$, is k if $*P*$ matched k sequences in $\mathcal{S}$ within distance at most i, i.e., the k sequence contain P within distance i.

The basic query is to find the pattern P where P is within the allowed distance *Dist* of at most *Occur* sequences in $\mathcal{D}$ and $|P| \geq Length$.[2]

2.2 Query Processing Algorithms

Our approach to query processing is a two phase process:

1. Find candidate segments among a small sample $\mathcal{A}$ of the sequences.

[2]Many related queries are also possible, e.g., a query to identify all patterns having at most a certain length with at least a certain activity.

2. Combine the segments to form candidate patterns and evaluate the activity of the patterns in all sequences of $\mathcal{D}$ to determine which patterns are solutions of the query.

Phase 1 consists of two subphases. In subphase A, we construct an index structure for the sequences in the sample. In subphase B, we traverse the structure to locate the candidate segments.

2.2.1 Subphase A of Phase 1

We construct a *generalized suffix tree* [39] (GST) for the sample of sequences. A suffix tree is a trie-like data structure that compactly represents a string by collapsing a series of nodes having one child to a single node whose parent edge is associated with a string. Suffix trees are used extensively in string matching [46, 54]. A GST 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 GST. Each leaf is associated with an index i. The edges are labeled with character strings such that the concatenation of the edge labels on the path from the root to the leaf with index i is a suffix of the ith string in the set. See Figure 2.6 for an example (the node labeled with a 1 above the leaf MTRM is an example of the result of a collapsing).[3] The GST can be constructed asymptotically in $O(n)$ time and space where n is the total length of all sequences in the sample $\mathcal{A}$.

For each non-leaf node v in the GST, let $subtree(v)$ be the subtree rooted at v. Let $string(v)$ be the string on the edge labels from the root to v. Let $count(v)$ represent the number of different indexes associated with the leaves in $subtree(v)$. We observe the following facts.

[3]The algorithm for constructing the GST works as follows. We append a unique symbol to each sequence in the sample and concatenate the sequences into a single one. We insert the suffixes of the sequences as into a trie except that if a node has only one child, we collapse the child with the parent and label the edge going down from the parent with a substring instead of a single character.

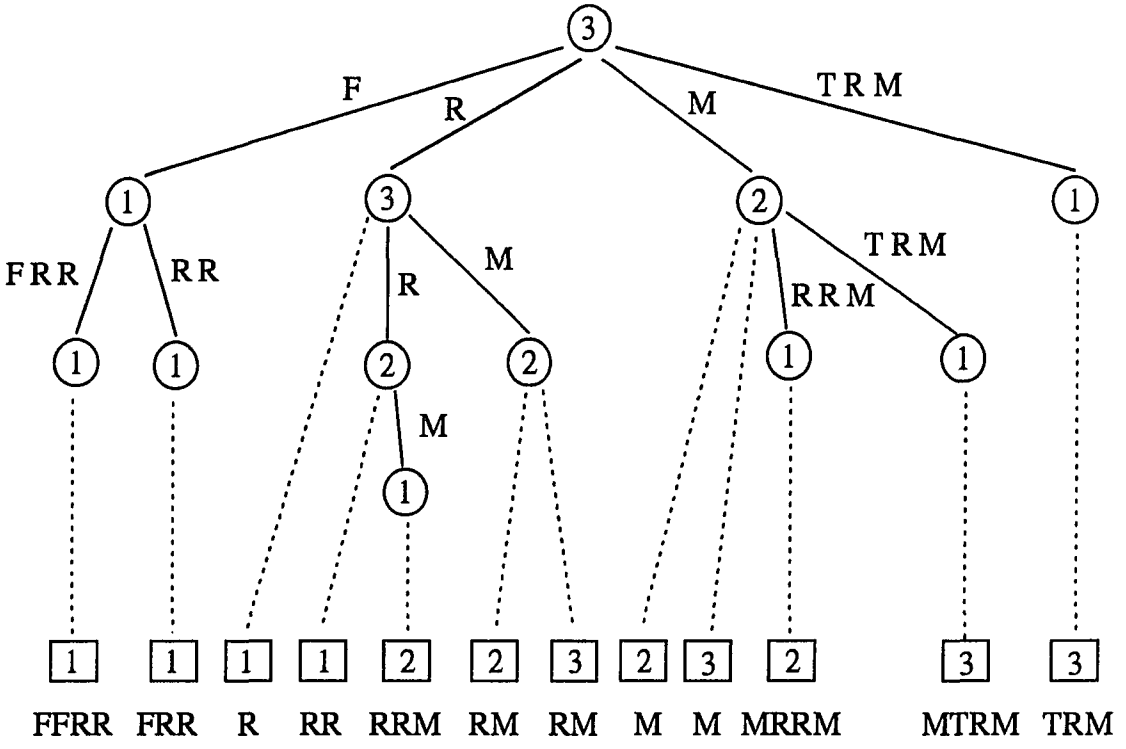

Figure 2.6 The GST for a sample $\mathcal{A}$ = {FFRR, MRRM, MTRM}. Leaves are represented as rectangles, labeled with the indexes. Non-leaf nodes are represented as circles, labeled with the *count* values. The suffix corresponding to a leaf is shown below the leaf. Note that the suffixes RM and M appear in two strings and hence appear twice in the leaves.

Fact 2.1 $\forall u \in subtree(v)$, $count(u) \leq count(v)$ and $|string(u)| < |string(v)|$.

Fact 2.2 If $count(v) = b$, then $occurrence\_no^0_{\mathcal{A}}(string(v)) = b$

The reason is that if $count(v) = b$, $string(v)$ is a prefix of the suffixes from b sequences in $\mathcal{A}$.

Fact 2.3 [39, 54] The time and space needed to construct the GST is $O(n)$ where n is the total length of all sequences in the sample.

2.2.2 Subphase B of Phase 1

In this phase, we traverse the GST constructed in subphase A to find all segments (i.e., all prefixes of strings labeled on root-to-leaf paths) that satisfy the length minimum. If the pattern specified by the user has the form $*X*$, then the length minimum is simply the specified minimum length of the pattern. If the pattern specified by the user has the form $*X_1*X_2*$, we find all the segments V_1, V_2 where at least one of the V_i, $1 \leq i \leq 2$, is (larger than or equal to) half of the specified length and the sum of their lengths satisfies the length requirement. If the user-specified pattern has the form $*X_1*X_2*\ldots*X_k*$, we find the segments $V_1, V_2, \ldots, V_k$ where at least one of the V_i, $1 \leq i \leq k$, is (larger than or equal to) $1/k$th of the specified length and the sum of the lengths of all these segments satisfies the length requirement.

2.2.3 Phase 2

This phase also has two subphases. In subphase A, we evaluate the activity of the candidate patterns and rank them from highest to lowest according to their occurrence numbers on the sample with respect to distance *Dist*. If the interesting patterns are of the form $*X_1*X_2*\ldots$, we consider all possible combinations $V_1, V_2, \ldots$ of the segments obtained in phase (1) that meet the length requirement and match $*V_1*V_2*\ldots$ with the sequences in the sample. Subphase B evaluates the most likely candidate patterns found in subphase A with respect to the entire database.

The motivation for having two subphases is that comparing a regular expression pattern P with a sequence S requires a dynamic programming approach that can take, in the worst case, $O(|P| \times |S|)$ time [94]. Screening out those unlikely candidate patterns in the first subphase may save significant time in the overall computation.

2.3 Generalizable Optimization Techniques

Whereas the discussion so far is specialized to the problem of finding patterns in sequences, certain heuristics can improve the efficiency of combinatorial discovery in general.

2.3.1 Pruning Unlikely Candidates

We would like to compare only the most likely candidate patterns with the entire set. The main question from an optimization point of view is which candidates to compare. Our strategy is as follows.

We use *simple random sampling without replacement* [28, 38, 51, 64] to select sample sequences from the set. Consider a candidate pattern P. Let D (a, respectively) denote the number of sequences in the entire set $\mathcal{D}$ (the sample $\mathcal{A}$, respectively) that contain P within the allowed number of distance. Let N be the database size and n the sample size; $F = D/N$ and $f = a/n$.

Fact 2.4 [19] With probability = 99%, F is in the interval $(\hat{F}_L, \hat{F}_U)$ where

$$\hat{F}_L = f - (t\sqrt{\frac{N-n}{N-1}}\sqrt{\frac{f(1-f)}{n}} + \frac{1}{2n}),$$

$$\hat{F}_U = f + (t\sqrt{\frac{N-n}{N-1}}\sqrt{\frac{f(1-f)}{n}} + \frac{1}{2n}).$$

The symbol t is the value of the normal deviate corresponding to the desired confidence probability. When the probability = 99%, $t = 2.58$ [19]. The values of N, n are given; f, a can be obtained by checking with the sample (cf. subphase A of phase 2). Thus, if the estimator $(\hat{F}_U \times N) < Occur$ for the candidate pattern P,

then with probability $\geq 99\%$, P won't be an active pattern satisfying the specified requirements. We therefore discard it. This pruning will be referred to as *candidate pattern optimization.* Since the optimization has to do only with sampling, it can be applied to not only sequences, but objects having other topological structures.

2.3.2 Eliminating Redundant Calculation of Occurrence Numbers

Observe that the most expensive operation in our algorithms is to find the occurrence number of a pattern with respect to the database, since that entails matching the pattern against all sequences. We develop two heuristics to avoid such computation when possible.

Definition 2.1 (Subpatterns for sequences) *Let* $P = *U_1 * U_2 * \ldots * U_m*$ *and* $P' = *V_1 * V_2 * \ldots * V_n*$ *where* $m \leq n$. *An* embedded mapping M *from* P *to* P' *is a set of* m *ordered pairs of integers* (i, j) *satisfying:*

1. $1 \leq i \leq m$, $1 \leq j \leq n$ *and* $i \leq j$;
2. *for any two distinct pairs* (i_1, j_1) *and* (i_2, j_2) *in* M, *(a)* $i_1 \neq i_2$ *and* $j_1 \neq j_2$, *(b)* $i_1 < i_2$ *iff* $j_1 < j_2$;
3. *if* $(i, j) \in M$, *then* $V_j = X \bullet U_i \bullet Y$ *where* X *and* Y *are two (possibly empty) segments and* $\bullet$ *represents the concatenation of segments. (Thus,* U_i *is a segment of* V_j.*)*

P is a subpattern *of P' if there exists an embedded mapping from P to P'.*

Proposition 2.1 *If P is a subpattern of P', then for any distance parameter k,* $occurrence\_no^k_{\mathcal{D}}(P) \geq occurrence\_no^k_{\mathcal{D}}(P')$

Proof. Let $dist(P, S)$ represent the distance between a pattern P and sequence S. The result follows by observing that for any sequence $S \in \mathcal{D}$, if $dist(P', S) = j$ for an integer j, we must have $dist(P, S) \leq j$. □

Thus, if P' is in the final output set, then we need not bother matching P, since it will be too. If P is not in the final output set, then P' won't be either, since its occurrence number will be even lower.

Let $occurrence\_set^k_{\mathcal{D}}(P)$ denote the set of all sequences in D that contain P within distance k, i.e., $|occurrence\_set^k_{\mathcal{D}}(P)| = occurrence\_no^k_{\mathcal{D}}(P)$

Proposition 2.2 *If P and P' are subpatterns of P'', then for any distance parameter k,*

$$occurrence\_set^k_{\mathcal{D}}(P'') \subseteq (occurrence\_set^k_{\mathcal{D}}(P) \cap occurrence\_no^k_{\mathcal{D}}(P')$$

Proof. For any sequence $S \in \mathcal{D}$, if $S \in occurrence\_no^k_{\mathcal{D}}(P'')$, by definition, we must have $dist(P'', S) = j$ for some integer $j \leq k$. It follows that $dist(P, S) \leq j$ and $dist(P', S) \leq j$. Hence, $S \in occurrence\_no^k_{\mathcal{D}}(P)$ and $S \in occurrence\_no^k_{\mathcal{D}}(P')$. □

Thus if $|(occurrence\_set^k_{\mathcal{D}}(P) \cap occurrence\_no^k_{\mathcal{D}}(P'))| < Occur$, we can eliminate P'' from consideration, since its occurrence number will be even lower. We refer the pruning strategies derived from the above propositions as *evaluation minimization.*

Example 2.3 (Illustration of the two-phase approach)
Consider the database $\mathcal{D}$ = {TFUR, MRRM, FFRR, MTRM, DPKY, VRWM, AVLG, KMRR}. Consider the query "Find the pattern P of the form $*X*$ where P is within distance 1 of at least 5 sequences in $\mathcal{D}$ and $|P| \geq 3$."

Suppose the chosen sample $\mathcal{A}$ = {MRRM, FFRR, MTRM, DPKY, AVLG}. At the end of phase 1, we obtain the following candidate patterns:

MRR *RRM* *MRRM*

FFR *FRR* *FFRR*

MTR *TRM* *MTRM*

DPK *PKY* *DPKY*

AVL *VLG* *AVLG*

By the statistical estimator, the most likely candidate pattern must occur (within distance 1) in at least 2 sequences in the sample. If a candidate is unlikely to be an answer, then any pattern containing it as a subpattern is unlikely either and should be discarded. Thus, at the end of subphase A of phase 2, we are left with

MRR *RRM* *MRRM*

FRR

MTR *TRM* *MTRM*

In subphase B of phase 2, since $occurrence\_no^1_{\mathcal{D}}$(*MRR*) = 4, we discard *MRRM*. Similarly, since $occurrence\_no^1_{\mathcal{D}}$(*MTR*) = 3, we discard *MTRM*. We compute $occurrence\_no^1_{\mathcal{D}}$(*TRM*) = 2, and $occurrence\_no^1_{\mathcal{D}}$(*FRR*) = 4. Neither of the two patterns is an answer. The only answer to the query is *RRM* where $occurrence\_no^1_{\mathcal{D}}$(*RRM*) = 5. □

2.4 Generalizing to Other Combinatorial Structures

The evaluation minimization techniques presented in the previous subsection have to do with relationships among patterns of the following form: if $dist(P, O) = d$ for some data object O and integer d, then $dist(P', O) \geq d$. This allows two conclusions. First, whenever P matches too few objects in the database within distance d, then P' will surely match no more. Second, if P' matches enough objects in the database within distance d, then P will surely match enough as well. The goal here is to

characterize the relationships between P and P' of that form in a more general setting.

Definition 2.2 (Subpatterns for general objects) *Let P and P' be two patterns containing VLDCs. P is a* subpattern *of P' if any object (which may or may not be in the database) C that matches P' within distance 0 will match P within distance 0. This conforms to the intuition that P is the less constraining of the two patterns.*

Call the set of objects within distance 0 of P, $Obj(P)$. The above definition of subpattern implies that $Obj(P) \supseteq Obj(P')$.

Definition 2.3 *A distance metric* dist *is said to be* VLDC-sensitive *if for any pattern P containing VLDCs and object Q, $dist(P,Q) = \min_{C \in Obj(P)}\{dist(C,Q)\}$.*

Proposition 2.3 *If P is a subpattern of P' and dist is VLDC-sensitive, then for any object Q in the database, $dist(P,Q) \leq dist(P',Q)$.*

Proof. Since P is a subpattern of P', $Obj(P) \supseteq Obj(P')$. So, $dist(P,Q) = \min_{C \in Obj(P)}\{dist(C,Q)\} \leq \min_{C \in Obj(P')}\{dist(C,Q)\} = dist(P',Q)$, which gives the result. □

Under this generalization of the notion of subpattern and this characterization of distance, Propositions 2.1 and 2.2 remain true, and therefore evaluation minimization still applies.

The general optimization strategy is summarized in Figure 2.7. Given a set of structures, the candidate enumeration subroutine first generates all candidate patterns from the set. The pruning subroutine eliminates impossible candidates from further consideration based on the sampling technique and pruning super-patterns away when their subpatterns are known to be unimportant. The verification subroutine compares those promising candidates against the data set to find qualified discovered patterns.

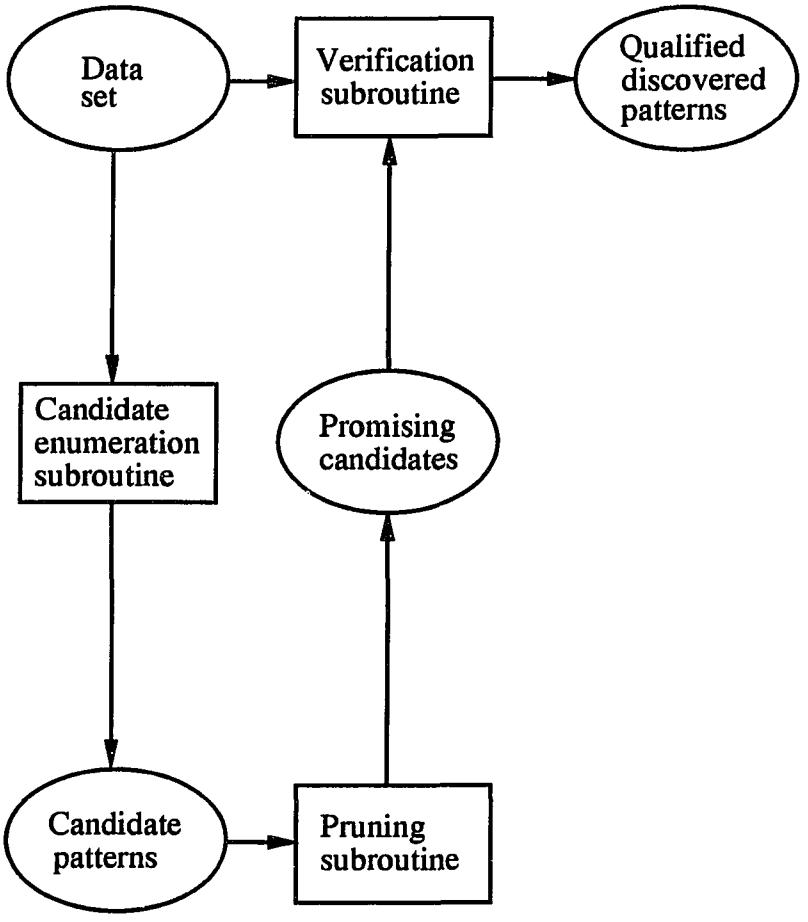

Figure 2.7 Overview of the general optimization strategy.

CHAPTER 3

PERFORMANCE ANALYSIS OF THE PATTERN DISCOVERY ALGORITHMS

The algorithm described in Chapter 2 has been implemented into a tool: DISCOVER. DISCOVER is written in C and run under the UNIX operation system as well as DOS system (cf. Chapter 9). In this chapter, we present two sets of experimental results. The first set of experimental results show the effectiveness and speed of DISCOVER. The second set of experimental results show the results of applying DISCOVER to proteins obtained from the Cold Spring Harbor Laboratory.[1]

3.1 Effectiveness and Speed of DISCOVER

3.1.1 Data and Parameters

We carried out a series of experiments to evaluate the effectiveness and speed (measured by elapsed CPU time) of our approach. The data was a set of randomly generated 150 sequences, each having length 100. Every letter of the generated sequence was drawn randomly from the protein alphabet. To gain a better understanding of the performance of our algorithms, we also tested the algorithms on real protein sequences. 150 proteins were selected randomly from the functionally related kinase family obtained from the Cold Spring Harbor Laboratory. The lengths of the kinase sequences ranged from 10 to 2938.

Table 3.1 shows the parameters and base values used in the experiments. The sequences in the sample were chosen randomly from the database. The parameter *NumSample* indicates the number of samples chosen for each database. In all the experiments presented here, only one sample was used in running a database. The

[1] Dr. Thomas Marr provided the data.

sample size was obtained by multiplying *DBSize* by *SizeRatio*. The patterns of interest had the form $*X*Y*$.

Table 3.1 Experimental parameters and base values.

Parameter	Value	Description
DBSize	150	# of sequences in a database
NumSample	1	# of samples tested for a database
SizeRatio	20%	Ratio between sample size and database size
Length	5	Minimum length of an interesting pattern
Dist	1	Allowed distance between a pattern and a sequence

The metric used to evaluate the effectiveness of our algorithms is $HitRatio = NumDiscovered/TotalNum \times 100\%$

where *NumDiscovered* is the number of interesting patterns discovered by our techniques. *HitRatio* stands for the percentage of the interesting patterns obtained from the exhaustive search method. The method works by considering all combinations of the segment pairs V_1, V_2 appearing in the database.[2] One would like this percentage to be as high as possible.

3.1.2 Results

Figure 3.1 shows the effectiveness of our approach for varying sample sizes. In this experiment, we had turned on both candidate pattern optimization and evaluation minimization when running our algorithms. The minimum occurrence number required $Occur = 60$ for the artificial data and $Occur = 8$ for kinase. (The different parameter values were chosen to illustrate different results using different data.) Examining the graphs, we see that when $Dist = 0$ and $SizeRatio \geq 0.2$, our

[2]We have rejected approximately occurring patterns that never appear in the database yet satisfy the *Dist* and *Occur* constraints in favor of those that obey the constraints and do appear in the database. This is a theoretical limitation of our work that we have introduced to save computation time, though this also seems pragmatically to be a reasonable approach.

approach behaves almost like exhaustive search. When $Dist = 1$, the hit ratio reaches 80% provided the $SizeRatio \geq 0.4$. We were somewhat disappointed that smaller sample didn't give a better hit ratio, but research is like that sometimes.

We next compared the running times of the algorithms for the $Dist = 1$ case. Figure 3.2 shows the results. It can be seen that our algorithms run significantly faster than the brute force method. Even with $SizeRatio = 0.8$, in which case the algorithms achieve nearly 100% hit ratio, they are 10 times faster than exhaustive search. When the sample is this large, both segments V_1, V_2 in a solution pattern appear in the sample. Our algorithms work by enumerating all promising segment pairs in the sample, and therefore can find all the interesting patterns.

We also examined the effectiveness of the proposed optimization heuristics. To isolate the effect of the heuristics, we started by turning off the optimizations, and then turned on only one of them, and finally turned on both. To make the experiment manageable, we considered only patterns of the form $*X*$. The minimum occurrence number required $Occur = 55$. The other parameters had the values shown in Table 3.1. Figures 3.3 and 3.4 show the results obtained from the kinase sequences. (The results for the generated sequences are omitted since they lead to similar conclusions.)

Examining the graphs, we see that very few solutions were missed by the candidate pattern optimization. Pruning based on subpattern information works more effectively than that based on statical estimation. Both optimizations together sped up the algorithms by a factor of nearly 100.

We repeated the experiments by varying the compositions of samples and parameter values *Length*, *Dist*, *Occur*. The results obtained are mostly consistent with those given above.

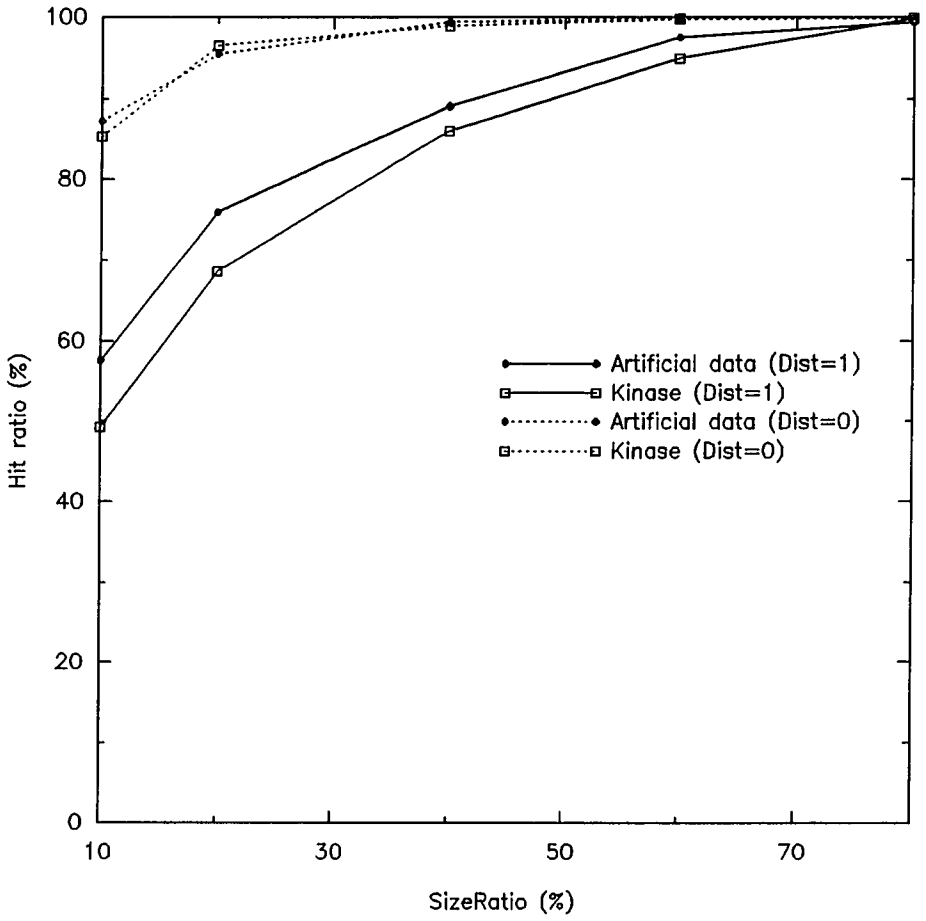

Figure 3.1 Effect of sample size.

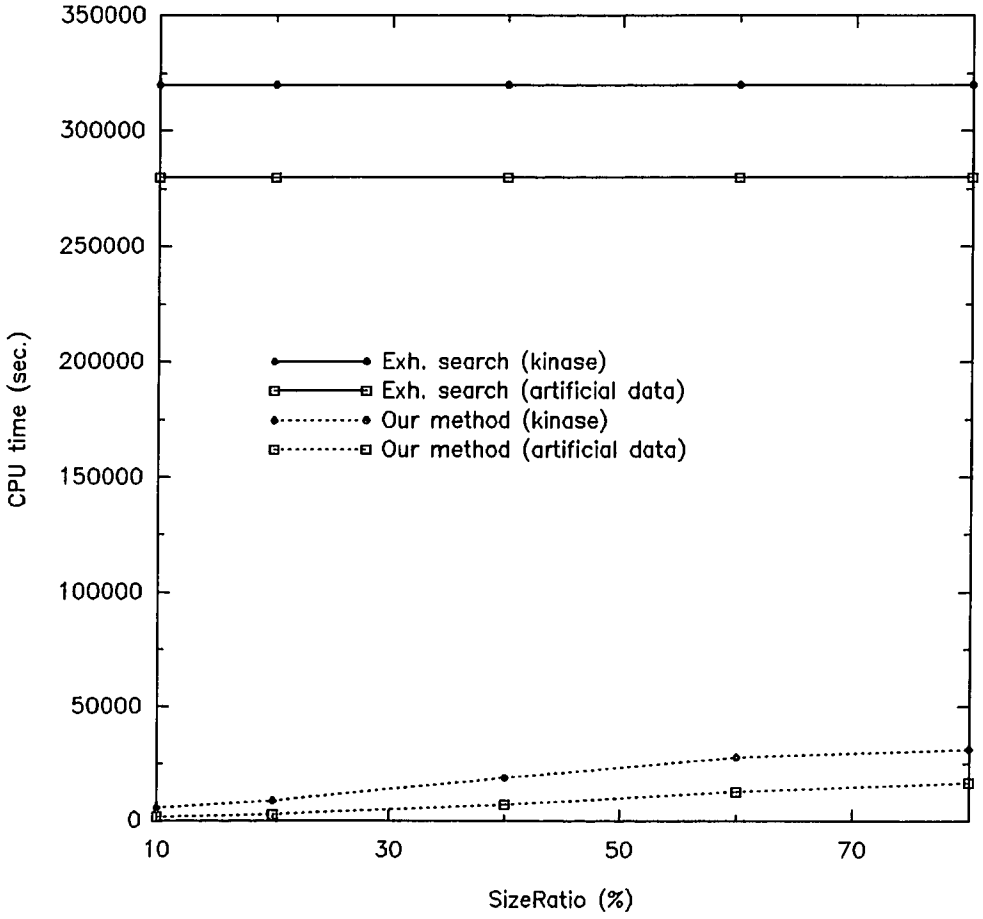

Figure 3.2 Comparison of running time.

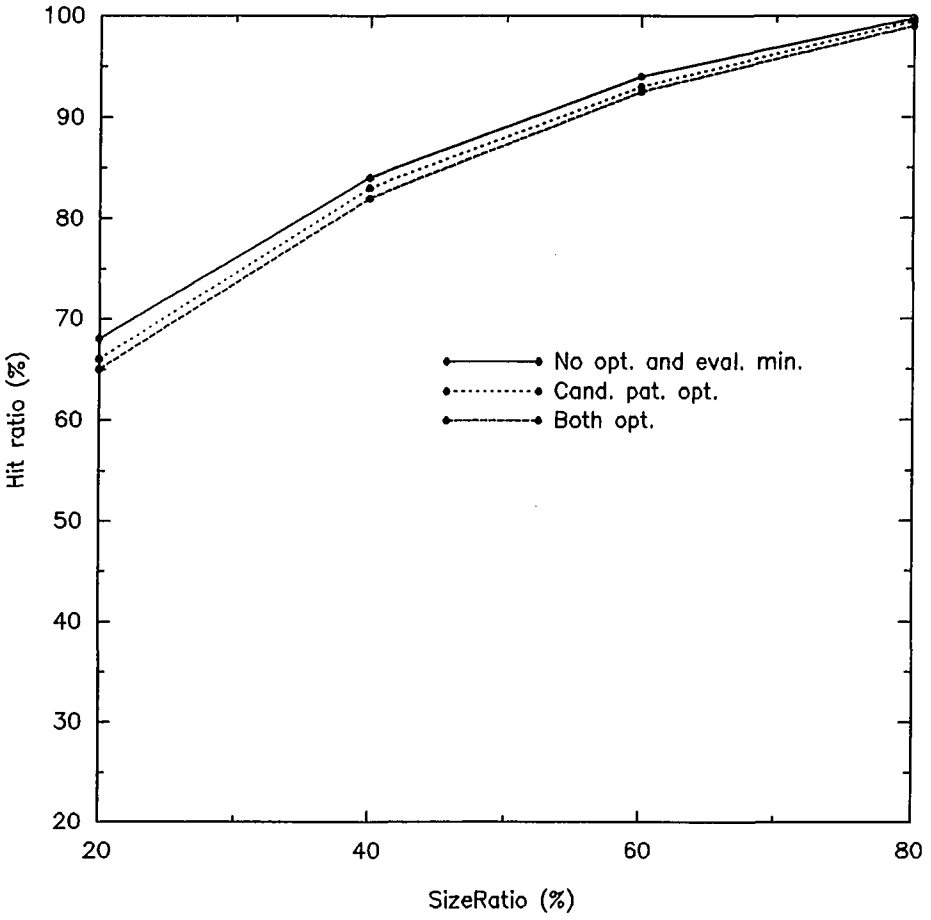

Figure 3.3 Performance of the pruning techniques.

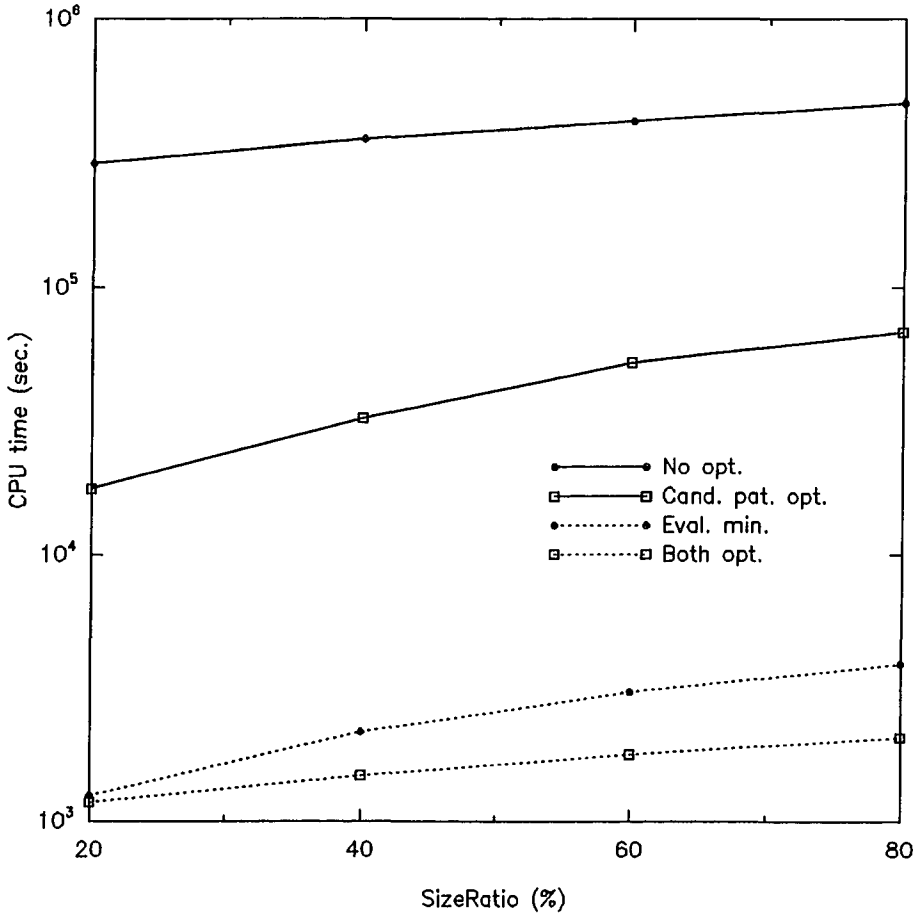

Figure 3.4 Efficiency of the pruning techniques.

3.2 Discovery of Active Patterns from Proteins

In this set of experiments we examined three protein families (cyclin, ras and kinase) obtained from the Cold Spring Harbor Laboratory to see whether the patterns discoveried correspond to those shown in previous studies which used other methods. The cyclin family contained 47 protein sequences, with the lengths ranging from 190 residues to 780 residues. The ras family contained 149 protein sequences, with the lengths ranging from 35 residues to 3079 residues. The kinase family contained 1077 protein sequences, with the lengths ranging from 10 residues to 2938 residues. Figure 3.5 shows 3 sequences in the FASTA format [66] obtained respectively from cyclin, ras and kinase family. In the FASTA format, each sequence has a title line starting with the symbol ">", followed by unlimited lines of data. Spaces, tabs and carriage returns are ignored in the data line.

Tables 3.2 3.3, and 3.4 show the active patterns (also known as motifs) of the form $*X*$ found by DISCOVER and their occurrence numbers with respect to mutation i, $0 \leq i \leq 4$, for the three protein families, respectively. The tables show, for each length of active patterns, the top one (or two) most active patterns discovered in each family. The activity of a pattern in a family is ranked in terms of its occurrence number with respect to mutation 0.

From these tables, it can be seen that shorter patterns tend to have higher occurrence numbers. The occurrence frequency of patterns is family dependent. In the ras family, for example, there is a very active segment DTAGQE, which appears in more than 60% proteins in the family. On the other hand, in the kinase family, the most active segment of the same length, DFGLAR, appears in less than 10% proteins in the family.

There are also patterns composed of segments appearing nonconsecutively in the protein sequences. Table 3.5 shows several active patterns composed of 2 nonconsecutive segments in the ras family.

```
>G1/S-SPECIFIC CYCLIN CLN1.
MTSLQQQQQQ QRVKYGPPHH IKRRPYHPIL ESLEFQTNQH LIQEYSLDIV NTLSQLESLT
LVNPAMIDLQ PEIQWFMRPF LLDFLIELHS SFKLQPTTLF LCLNIIDRYC AKRIVFKRHY
QLVGCTALWI ASKYEDKKLR VPTLKELTIM CRNAYDEEMF VQMEMHILST LDWSIGHPTL
EDCLQLAIDL NNLSNNTTND IENKSVRPNR KSSISSAVTA VARFLCELSL YDKYFLSVPP
SLIAITANLL SCSMLQIPHA SITLKNLIEQ EIINPQQKKQ KKAFSSNSSR TTTASYTHQN
QLDVRHSSFD EDIDLDSGDE GDDDEDYIDE FYETNNYDDT NATTFDESIN KSTTINDENQ
PPQIHTPFLS GLDEDSILSI KKICLMLIIQ LSKVTEVLSK KYENLGVIQV INNFHSNYKF
IIQSIYENQE LLLNTINDST NNNEIDYKLI QSSEILLQFP KFDEYLTEDE DENVSTDDEA
NSQPQGYDGS GSDGNNQLFT PKSPNAFSSN SSLTLNNHPQ SMVPVTPPSA TSQYSLFSNK
NNRTHESTSG LNSTCNTPTH ISISSFAPPQ PPPGSILKPK LTSINSTNSL KIKKLTSNSN
SSNINIHHGH HNTKQEKRYS HISIGSNSSS KYDGFSPIKS ISTNGS
```

(a)

```
>RAS PROTEINS GERANYLGERANYLTRANSFERASE (EC 2.5.1.-)
MCQATNGPSR VVTKKHRKFF ERHLQLLPSS HQGHDVNRMA IIFYSISGLS IFDVNVSAKY
GDHLGWMRKH YIKTVLDDTE NTVISGFVGS LVMNIPHATT INLPNTLFAL LSMIMLRDYE
YFETILDKRS LARFVSKCQR PDRGSFVSCL DYKTNCGSSV DSDDLRFCYI AVAILYICGC
RSKEDFDEYI DTEKLLGYIM SQQCYNGAFG AHNEPHSGYT SCALSTLALL SSLEKLSDKF
KEDTITWLLH RQVSSHGCMK FESELNASYD QSDDGGFQGR ENKFADTCYA FWCLNSLHLL
TKDWKMLCQT ELVTNYLLDR TQKTLTGGFS KNDEEDADLY HSCLGSAALA LIEGKFNGEL
CIPQEIFNDF SKRCCF
```

(b)

```
>ABCISIC ACID-INDUCIBLE PROTEIN KINASE (EC 2.7.1.-)
GSGNFGVAKL VRDVRTKEHF AVKFIERGHK IDEHVQREIM NHRSLKHPNI IRFKEVVLTP
THLAIVMEYA SGGELFQRIC NAGRFSEDEG RFFFQQLISG VSYCHSMQVC HRDLKLENTL
LDGSVAPRLK ICDFGYSKSS VLHSQPKSTV GTPAYIAPEV LSRREYDGKV ADVWSCGVTL
YVMLVGAYPF EDPDEPRNFR KTITRILSVQ YSVPDYVRVS MDCIHLLSRI FVGNPQQRIT
IPEIKNHPWF LKRLPVEMTD EYQRSMQLAD MNTPSQSLEE AMAIIQEAQK PGDNALGVAG
QVACLGSMDL DDIDFDIDDI DVESSGDFVC PL
```

(c)

Figure 3.5 Examples of test proteins. (a) a cyclin sequence; (b) a ras sequence; (c) a kinase sequence.

Table 3.2 Patterns discovered for the cyclin family ($SizeRatio$ = 20%). For each length of patterns, only the top one (or two) most active ones discovered are shown in the table.

Patterns found by DISCOVER	Occurrence number w.r.t. distance				
	0	1	2	3	4
LQL	27	47	47	47	47
QLV	26	47	47	47	47
QLVG	24	35	47	47	47
KYEE	20	42	47	47	47
LQLVG	19	29	43	47	47
ASKYEE	13	25	35	46	47
KLQLVG	13	22	33	47	47
IASKYEE	9	20	29	36	47

Table 3.3 Patterns discovered for the ras family ($SizeRatio$ = 20%). For each length of patterns, only the top one (or two) most active ones discovered are shown in the table.

Patterns found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
TAG	106	147	149	149	149
DTAG	99	125	148	149	149
DTAGQ	92	104	134	148	149
LVGNK	62	103	138	149	149
DTAGQE	90	101	110	146	149
WDTAGQE	50	90	101	111	147
GVGKSALT	41	45	61	88	130
YDPTIEDSY	38	41	42	47	76

Table 3.4 Patterns discovered for the kinase family ($SizeRatio$ = 5%). For each length of patterns, only the top one (or two) most active ones discovered are shown in the table.

Patterns found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
DFG	338	1018	1076	1077	1077
ELL	331	1048	1075	1077	1077
HRDL	174	484	1034	1077	1077
DFGL	166	567	1062	1076	1077
DFGLA	127	257	844	1070	1076
FGLAR	97	183	817	1070	1077
DFGLAR	97	146	367	967	1072
RDLAARN	67	79	124	515	1039

Table 3.5 Occurrence numbers for the active patterns composed of 2 nonconsecutive segments in the ras family.

Patterns composed of 2 nonconsecutive segments found by DISCOVER	Occurrence number w.r.t. mutation				
	0	1	2	3	4
*DTAGQE*LVGNK*	52	93	97	104	112
*GGVGKSALT*LVGNK*	29	40	43	55	63
*YDPTIEDSY*LVGNK*	29	40	40	44	58
*YDPTIEDSY*DTAGQE*	31	36	41	46	57
*GGVGKSALT*DTAGQE*	28	40	44	61	82

It is worth pointing out that the patterns discovered in the cyclin and ras sequences are a superset of those found manually by O'Farrell and Leopold [62]. The kinase sequence patterns that we were able to detect overlap with the sequence patterns described in [29, 76].

CHAPTER 4

ALGORITHMS FOR DNA SEQUENCE CLASSIFICATION

After describing the sequence pattern discovery algorithm and the experimental results concerning the algorithm, we now turn to the algorithms for DNA sequence classification.

DNA sequence classification is an important problem in computational biology [12, 25, 42, 43, 59, 67, 93]. Given an unlabeled sequence S, a classifier makes predictions as to whether or not the sequence belongs to a particular class $\mathcal{C}$. Many computer-assisted techniques have been proposed for constructing classifiers from a library of labeled sequences. In general, these techniques can be categorized into the following three classes:

- consensus search – this approach takes a collection of sequences of the class $\mathcal{C}$ and generates a "consensus" sequence which is then used to identify sequences in uncharacterized DNA [9, 10, 24, 25, 42, 56, 57, 58, 59, 60, 61, 65, 77, 80];
- inductive learning/neural networks – this approach takes a set of sequences of the class $\mathcal{C}$ and a set of sequences not in $\mathcal{C}$ and then, based on these sequences and using learning techniques, derives a rule that determines whether the unlabeled sequence S belongs to $\mathcal{C}$ or not [25, 35, 36, 47, 52, 68, 70, 75];
- sequence alignment – this approach aligns the unlabeled sequence S with members of $\mathcal{C}$ using an existing tool such as FASTA [50, 66] and assigns S to $\mathcal{C}$ if the resulting alignment score is sufficiently high.

In this chapter, we propose two new techniques, as an alternative of alignment, for calculating scores to classify DNA sequences. Our approach works by first randomly selecting a set $\mathcal{B}$ of sequences of the class $\mathcal{C}$, referred to as "base data." Then we take another set of sequences of $\mathcal{C}$, referred to as "positive training data,"

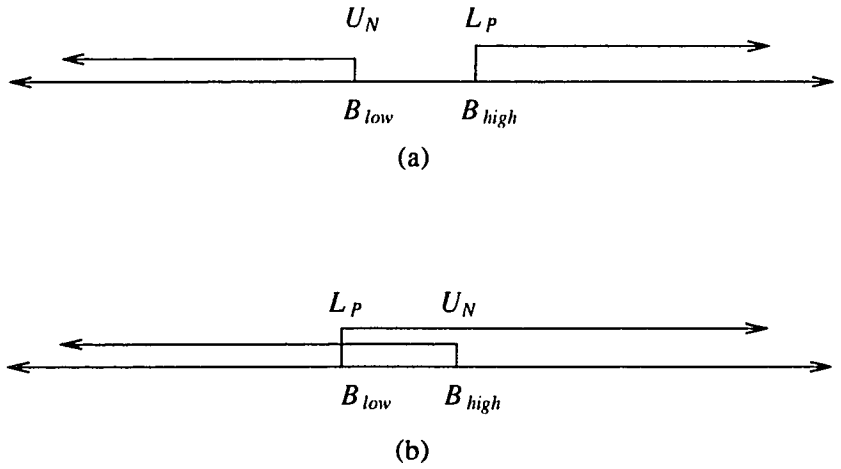

Figure 4.1 Illustration of the parameters B_{low} and B_{high} for cases (a) $U_N \leq L_P$ and (b) $U_N > L_P$.

and calculate, for each positive training sequence, a score with respect the base sequences. The minimum score thus obtained is called the positive lower bound, denoted L_p. Next, we take a set of sequences not in $\mathcal{C}$, referred to as "negative training data," and again calculate, for each negative training sequence, a score with respect to the base sequences. The maximum score thus obtained is called the negative upper bound, denoted U_n. Let $B_{high} = \max\{L_p, U_n\}$ and $B_{low} = \min\{L_p, U_n\}$ (see Figure 4.1. 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 $\mathcal{C}$. If $c \leq B_{low}$, then S is classified not to be a member of $\mathcal{C}$. If $B_{low} < c < B_{high}$, then the "no opinion" verdict is given.

The two proposed classifiers differ in their ways of processing the base sequences and calculating scores for the training and unlabeled sequences. We describe their algorithms in detail in the following sections. To demonstrate the utility of our approach, we compare it with FASTA and evaluate the precision rates obtained by using these tools to classify Alu sequences [41]. Our results show that the proposed classifiers work as well as FASTA in terms of the number of correct classifications, but misclassify different sequences. Thus, using these tools together either gives high

GGAGAGGCCGGGCGTGTGCCGGTAC
GGCCAGGCGGCAGATCTTGACCAGG
TGTAATCAGAGCGCCAGGCAAACAT

Figure 4.2 Three base sequences.

confidence to the classification (if the tools agree) or suggests further study on the given unlabeled sequence (if the tools disagree).

4.1 Algorithm for DNA Classifier I (DC-1)

Given the set $\mathcal{B}$ of base sequences, DNA Classifier I (referred to as DC-1) first searches for *active patterns* that approximately match all, or the majority of, sequences of $\mathcal{B}$ using our tool DISCOVER [88, 89] (cf. Chapter 2 and 3). A pattern here is a substring made up of consecutive nucleotides of a sequence. For example, consider the three base sequences in Figure 4.2. Suppose the pattern to be sought has length greater than 6 (i.e., it contains more than 6 nucleotides), occurrence number 3 (i.e., it matches all the three base sequences), and mutation 1 (i.e., one mismatch, insertion or deletion of a nucleotide is allowed in matching the pattern with a base sequence). Then GCCGGGC and GCCAGGC underlined in Figure 4.2 are two qualified patterns.

The patterns thus found may share a large common portion among them. Central to the classification algorithm is a gluing procedure, which combines two patterns into a longer one. In gluing two patterns, the procedure aligns their common portion as much as possible. The aligned portion can differ by at most k nucleotides (i.e., at most k mismatches are allowed in the common portion), and the length of the substring of each pattern that are not aligned must be less than a threshold α.

Example 4.1 (Gluing two patterns)

Consider the two patterns S_1 = GCAGCG and S_2 = ACCGC in Figure 4.3(a) and six possible alignments between them. Suppose k is 1 and α is 3. Then in the 5^{th} alignment, the two patterns can be glued into a new one. The reason is that the

```
S1:    GCAGCG   GCAGCG  GCAGCG  GCAGCG  GCAGCG   GCAGCG
         |||      ||||    |||||   |||||    ||||     |||
S2:  ACCGC      ACCGC    ACCGC   ACCGC    ACCGC    ACCGC
```

(a)

GCAXCGC

(b)

Figure 4.3 (a) Six possible alignments between S_1 and S_2. (b) The glued result.

aligned portion differs by only one nucleotide and the length of the substring of S_1 (S_2, respectively) that is not aligned is 2 (1, respectively), which is less than the threshold 3. Figure 4.3(b) shows the glued result. After gluing the two patterns, the procedure replaces the mismatched G and C by an introduced letter X. Intuitively, we consider it to be a match when aligning X with either G or C. (We used 15 introduced letters to represent 15 different combinations of the four nucleotides A, C, G and T.) Notice that the newly glued pattern may not be longer than the original ones if one of the original patterns is a substring of the other.

□

To classify an unlabeled sequence, DNA Classifier I pre-processes the base sequences in $\mathcal{B}$ by generating a set of "representative patterns" from them as follows. The classifier first sorts (in descending order) the discovered active patterns according to their occurrence numbers in $\mathcal{B}$. (The occurrence number of a pattern is the number of sequences in $\mathcal{B}$ which the pattern can match within the allowed mutation.) The classifier then looks at two patterns at a time, in a top down fashion, in the sorted list. If it is found that the i^{th} pattern can be glued with the j^{th} pattern in the sorted list where $i < j$ (i.e., the i^{th} pattern has a larger occurrence number than the j^{th} pattern), then the newly glued pattern is placed in the i^{th} position and the original two patterns are removed from the sorted list. Intuitively we use the newly glued pattern to represent the original two patterns in the list. Thus the sorted list shrinks

Input: A sorted list $\mathcal{L}$ of active patterns discovered from the set $\mathcal{B}$ of base sequences.
Output: A set $\mathcal{R}$ of representative patterns of the base sequences.
$i := 1$;
repeat
  /* Let the pattern placed at the i^{th} position in $\mathcal{L}$ be S_1. */
  if there exists a pattern which can be glued with S_1 and
  whose position in $\mathcal{L}$ is lower than i **then**
    begin
    /* Let S_2 be the first such pattern and its position in $\mathcal{L}$ be j. */
      glue S_1 and S_2 and call the result S_3;
      remove S_1 and S_2 from $\mathcal{L}$;
      place S_3 at the i^{th} position in $\mathcal{L}$;
    end
  else
    $i := i + 1$;
until $\mathcal{L}$ can not be shrunk further and $i = |\mathcal{L}|$;

Figure 4.4 Algorithm Gluing.

after applying each gluing operation. This continues until the sorted list can not be shrunk any further. The result is a set $\mathcal{R}$ of representative patterns. Figure 4.4 summarizes the algorithm.

Intuitively, the gluing algorithm makes up representative patterns of the base sequences by combining active patterns into longer ones. The longer and the more active a representative pattern is (i.e., the more base sequences the pattern matches), the better it characterizes the base sequences.

Given a sequence S (which could be a training or an unlabeled one), the score between S and a representative $P \in \mathcal{R}$, denoted $score(S, P)$, is defined as $|L|$ (i.e., the number of nucleotides in L), where L is the longest common substring of S and P. (The time complexity for finding the score is bounded by $O(|L|)$, and at worst $O(|S| \times |P|)$ [17, 18].) The score of S with respect to the base sequences is defined as

$$score(S) = \max\{score(S, P) | P \in \mathcal{R}\}.$$

4.2 Algorithm for DNA Classifier II (DC-2)

DNA Classifier II (referred to as DC-2) uses a hash-based fingerprint technique to calculate the score of a sequence. This technique is an extension of Califano and Rigoutsos's table look-up scheme [14] which uses fingerprints to find the best alignment between two sequences.

Given the set $\mathcal{B}$ of base sequences, we store their fingerprints into a number of *fingerprint files* as follows. Let S be a sequence in $\mathcal{B}$. We take every contiguous substring (or *segment*), denoted by Seg, of length n from S and generate gaped fingerprints from Seg. Each fingerprint is a substring of Seg that always begins with the segment's first nucleotide. The lengths of the fingerprints range from 2 to $n-1$. The number of gaps in a fingerprint is bounded by a parameter *gap*.

Next, for each fingerprint f of length k, $2 \leq k \leq n-1$, we use a hash function h_k to hash f into a fingerprint file $\mathcal{F}_k$. In $\mathcal{F}_k$, 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 x^{th} sequence in $\mathcal{B}$ and y means that the first character of f occurs at the y^{th} position in that sequence.

Example 4.2 (Generating fingerprints)

Consider the following three base sequences: S_1 = `ACGTTGCA`, S_2 = `ACCAGTG`, S_3 = `CGGACTA`. Suppose the length of segments is 6. Then, for example, we obtain the following segments from S_1: `ACGTTG`, `CGTTGC` and `GTTGCA`.

Now consider the segment Seg = `ACGTTG` taken from S_1. Suppose $gap = 2$. Then, we can generate the following 3-nucleotide gaped fingerprints from Seg: `ACG` (0 gap); `AGT` (1 gap at position 2), `ACT` (1 gap at position 3); `ATT` (1 gap at position 2 and 1 gap at position 3), `AGT` (1 gap at position 2 and 1 gap at position 4) and `ACT` (1 gap at position 3 and 1 gap at position 4). Table 4.1 summarizes all gaped fingerprints generated from the segments of S_1.

Table 4.1 Gaped fingerprints (of lengths 2, 3, 4, 5, respectively) generated from the segment ACGTTG (the segment length is 6 and $gap = 2$).

	2-nucleotide fingerprints	3-nucleotide fingerprints	4-nucleotide fingerprints	5-nucleotide fingerprints
0 gap	AC	ACG	ACGT	ACGTT
1 gap	AG	ACT	ACGT	ACGTG
		AGT	ACTT	ACTTG
			AGTT	AGTTG
2 gaps	AT	ACT	ACGG	
		AGT	ACTG	
		ATT	AGTG	
			ATTG	

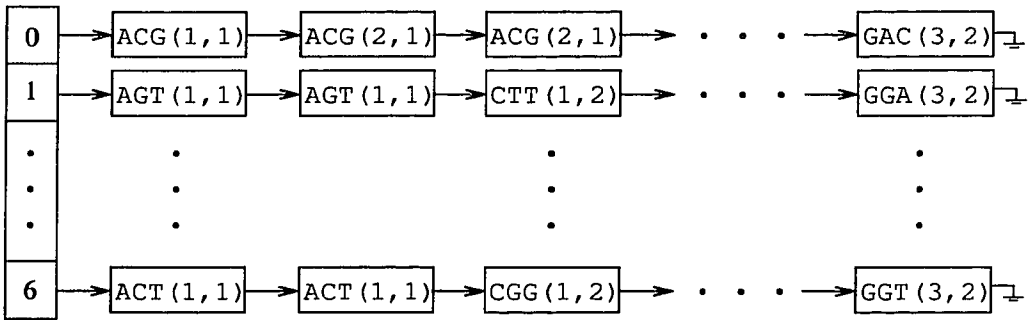

Figure 4.5 The fingerprint file $\mathcal{F}_3$ for the three base sequences in Example 4.2.

Let f = XYZ be a fingerprint of length 3. Suppose the hash function h_3 is $h_3(f)$ = $(num(\mathtt{X}) \times 4^2 + num(\mathtt{Y}) \times 4^1 + num(\mathtt{Z}))$ mod 7, where $num(\mathtt{X})$ is X's ASCII value minus 64. Figure 4.5 shows the fingerprint file $\mathcal{F}_3$ for the base sequences S_1, S_2 and S_3. Thus, for example, in bucket 1 in $\mathcal{F}_3$, GGA(3,2) means that the fingerprint GGA is generated from S_3 and it starts from the 2^{nd} position in S_3.

□

When calculating the score of a sequence S (whether it is a training or an unlabeled sequence), we segment S in the same way as for the base sequences and generate fingerprints from the resulting segments. We then hash the fingerprints, using the same hash functions as for the base sequences. When a match between

Input: A sequence S, a set $\mathcal{B}$ of base sequences and $\mathcal{B}$'s fingerprint files.
Output: A histogram of votes on the base sequences in $\mathcal{B}$.
/* Let $\mathcal{F}$ contain all fingerprints generated from S. */
for each fingerprint f in $\mathcal{F}$ **do**
  begin
    /* Let the length of f be k. */
    hash f using h_k and probe into the fingerprint file $\mathcal{F}_k$;
    for each match between f and a fingerprint $\hat{f}$ in $\mathcal{F}_k$ **do**
      begin
        /* Let the position marker associated with $\hat{f}$ 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 i^{th} base sequence in $\mathcal{B}$;
      end;
  end;

Figure 4.6 Algorithm Scoring.

S's fingerprint and a base sequence's fingerprint occurs, we give one score to an appropriate position on the base sequence. The result is a histogram of votes on the base sequences. Figure 4.6 summarizes the algorithm.

Example 4.3 (Voting on the fingerprints)

Suppose we are given a sequence S = CGATGCAT. Figure 4.7 shows the histogram obtained after matching S's fingerprints with the fingerprints of the base sequences in Example 4.2. □

Let B be a base sequence in $\mathcal{B}$ and let p be a position in B, $1 \leq p \leq |B|$. Let $score(B[p])$ represent the total scores added to the position p after applying the algorithm Scoring to the given sequence S and the base sequences in $\mathcal{B}$. The score of B, denoted $score(B)$, is defined to be

$$score(B) = \max\{vote(B[p]) | 1 \leq p \leq |B|\}.$$

The score of S with respect to the base sequences, denoted $score(S)$, is defined to be

$$score(S) = \frac{\max\{score(B) | B \in \mathcal{B}\}}{|S|} \times 100.$$

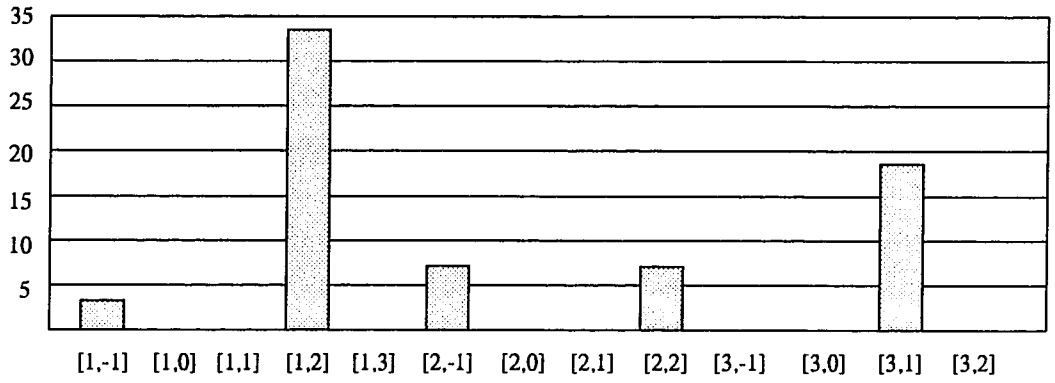

Figure 4.7 The histogram obtained after processing the sequence S in Example 3. The y-axis shows votes. Each $[i, q]$ on the x-axis represents the q^{th} position in the i^{th} base sequence in Example 4.2.

CHAPTER 5

EXPERIMENTAL RESULTS OF DNA CLASSIFICATION

The algorithms for the proposed classifiers DC-1 and DC-2 were implemented in C on a Sun SPARCstation 20 running the operating system Solaris version 2.4. We compared the relative performance of the classifiers by running them on the Alu sequences [41]. 300 Alu sequences were used in the experiments, among which 100 were used as the base sequences, 100 were used as positive training sequences, and the other 100 were treated as unlabeled test sequences. In addition, 1,253 non-Alu sequences were selected from the Eukaryotic Promoter Database, among which 100 were used as the negative training sequences and the other 1,153 were also treated as unlabeled test sequences. All these sequences were obtained from the BLASTN database [1];[1] their lengths ranged from 58 bp to 600 bp. The Alu sequences were obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/pub/jmc/alu/". The Eukaryotic Promoter Database was obtained from the ftp site "ncbi.nlm.nih.gov" under the directory " /repository/EPD/epd/".

5.1 Data and Parameters

DC-1 found active segments (patterns of the form *X*) from the 100 base sequences using our tool DISCOVER [88, 89] (cf. Chapter 2 and 3). The active segments had length greater than 10, occurrence numbers 20 and mutation 0 (i.e., these segments matched at least 20 base sequences without mutation). There were 2,046 active segments. After gluing the segments using the algorithm in Figure 4.4, we obtained 432 representatives patterns for the base sequences, with lengths ranging from 11 bp to 70 bp. DC-2 fixed the segment length at 5, and generated fingerprints with

[1]This server can be accessed by sending an email to "blast@ncbi.nlm.nih.gov" with the word HELP in the body of the message.

Table 5.1 Experimental parameters and their default values used in performance analysis.

Parameter	Value	Description
$\|\mathcal{B}\|$	100	Number of base sequences (Alu)
$\|\mathcal{T}_P\|$	100	Number of positive training sequences (Alu)
$\|\mathcal{T}_N\|$	100	Number of negative training sequences (non-Alu)
NumTest	1,253	Number of unlabeled test sequences (Alu & non-Alu)
Length	11	Minimum length of active patterns used in DC-1
Occur	20	Minimum occurrence number of active patterns used in DC-1
Mutation	1	Allowed mutation between an active pattern and a base sequence used in DC-1
k	0	Allowed number of mismatches in the gluing algorithm of DC-1
α	3	Threshold used in the gluing algorithm of DC-1
n	5	Segment length used in DC-2
gap	0	Number of gaps allowed in DC-2

lengths ranging from 2 to 5 and *gap* being 0. Table 5.1 summarizes the parameters and base values used in the experiments.

The metrics used to evaluate the effectiveness of our classification algorithms are precision rates (PR) and no-opinion rates (NR), where

$$PR = \frac{NumCorrect}{NumTest} \times 100\%$$

and

$$NR = \frac{NumNoOpinion}{NumTest} \times 100\%$$

NumCorrect is the number of test sequences classified correctly, *NumNoOpinion* is the number of test sequences obtaining the "no opinion" verdict, and *NumTest* is the total number of test sequences, 1,253 in our case. (A test sequence S in a class $\mathcal{C}$ is said to be classified correctly by an algorithm if S is determined by the algorithm to belong to $\mathcal{C}$.)

Table 5.2 Classification results for the three studied classifiers.

	DC-1	DC-2	FASTA
PR	96.0%	99.4%	99.2%
NR	2.8%	0.2%	—
B_{high}	15	45	—
B_{low}	13	35	—

5.2 Experimental Results

Table 5.2 shows the classification results and the B_{high} and B_{low} values obtained in the proposed classifiers. For comparison purposes, we also list the results obtained from the FASTA classifier [50, 66] currently used in the Whitehead Institute.[2] It can be seen that the two proposed classifiers are comparable to the FASTA classifier. Also, very few sequences obtained the "no opinion" verdict from the proposed classifiers.

Table 5.3 shows the complementarity results between the three studied classifiers. The three classifiers are said to *agree* on a test sequence, if all of the classifiers determined that the test sequence was an Alu, or all of them determined that the sequence was a non-Alu. Otherwise, the three classifiers are said to *disagree* on the test sequence. We see from the table that when the three classifiers agree, the classification has a high likelihood of being correct. Specifically, the correct agreed-upon classification divided by the total agreed-upon classification is 95.37% / (95.37% + 0.24%) = 99.75 %.

It is interesting to note that every sequence of the 0.24% of test data (3 sequences) in Table 5.3 were promoter, but were misclassified as Alu by all the three tools. Figure 5.1 shows these sequences. This result suggests that membership of the sequences merit further study.

[2]This tool classifies a given unlabeled DNA sequence into either Alu or non-Alu; it does not provide the "no opinion" option. Dr. Steve Rozen of the MIT Whitehead Institute used the tool to generate data used in the experiments.

>EPD26007 (+) Hs PBGD H E1; range -499 to 100.
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNCAGCTGCTTGGGAGGCTGAGGTGGGGGGATCACTGAAGCCGGGAGGTCAAGGCTGCAG
TGACCCGTGGTCATGCCGCTGCACTCTAGTCTGGGGACACAGTGAGACCCCGTATCAAAA
AGAAAAATGCTGCCTATTTCAAGGTTGTAGCAAAGCTAAGTTTGAACAGAGCAAAGGAAG
CGCCATAGAAGCTGCACTACTTGCTCATGTCACAGCTGGGGAATGGGGAGGTCGAATGGG
GAGGTCCACTGTCGCAATGTTCCAATTCCCGCCCAGAGGGAGGGACCTCCCCTTCGAGGG
AGGGCGCCGGAAGTGACGCGAGGCTCTGCGGAGACCAGGAGTCAGACTGTAGGACGACCT
CGGGTCCCACGTGTCCCCGGTACTCGCCGGCCGGAGCCTCCGGCTTCCCGGGGCCGGGGG

>EPD41008 (+) Hs snRNP E; range -499 to 100.
TTGGGAGGCCGAGGCGGGCGGATCATGAGGTCAGGAGATCGAGACCATCCTGGCCAACAT
GGTGAAACCCCGTCTCTACTAAAAACACAAAAAATTAGTCGGACGTGGTGGCAGGCGCCT
GTAGTCCCAGCTACTCGGGAGGCTGAGGCAGGAGAATCGCTTGAACCCTGGAGGCTGAGG
TTGCAGTGAGCTGAAATCGCGCCACTGCACTCCAGCCTGGGCGACAGAGCGAGACTTCGC
CTCAAAAAAAAAAAAAAGAAAAGAAATTGGAGACAGGACGGCACTCCTCCTACCACGGCCA
CTATACAACCGAAGGCACCTCCTCCCGCCTCCCTCAGGTGGCGCTGTAGGACCGGCCCAC
CTTCCACAGCGCAGACGCCTCAGTTTGGCCTCCCTCGGAGGCCATGCAGCGAAGAAACGT
GACTTCATGGCTAGAGGGGCATTTCCCAGGAGCCGCCCATGCCGCCGCGTGACCTTCACA
CTTCCGCTTCCGGTTCTTTATTCCGGAAGTTGCTCTCAGAGGCAGCGTGCGGGTGTGCTC
TTTGTGAAATTCCACCATGGCGTACCGTGGCCAGGGTCAGAAAGTGCAGAAGGTTATGGT

>EPD39002 (+) Hs renin-b. p. RnBP; range -499 to 100.
AGGAGTTCGAGACCAGCCTGGGCAACATTGCAAAACCCTGTCTCCACTAAAAATACAAAA
ATTAGCCAGGCATGGTGGCCCACACCTGTGGTCCCACCTACTCAGGAGGCTGAGGTGGGA
GGATCGCTTGAGCCTGGCAGGTCAAGGCTGCAGTGAGCTGCCTGGGCGACAGGACACTGT
CTCAAAAAACAAAGAAACAAACAACAACAACAACAACAAAAACAGAACTTGCCTCCTGTC
TACCCTGTGTGGTGACAGGTCAGAGGGACCTCTGCAATTCAGGCCCACTAGCTGTTCCCT
GTCCTGCTGCACCCCTGCCCCACAGGCTTGGTAGGCCAAGGCCCTCAGCCGCCTGCCGGG
GCCCCTGGTGTAGGGCGGGGTGGGTGGCAGCCCCGTTCCCCCTCATCCCAGACCCTCCGG
CCATCCCAGCTTTTCCCACCAAGCAGGATCTTGAGACCAGAGCTCCCAGCAGGGGCCCTG
GAGGGACTGGCCCATTCGAGTGGAGGGTCGCGGCAGGAACAAGTGCGGGGGCGGGGCCTC
AAAGCCCCGCCCCCAGGCCCAGCGGAACCCTGAAGAGGAGGGGGAGGGTGAGCCGGGCGC

Figure 5.1 The three misclassified sequences.

Table 5.3 Complementarity between the three studied classifiers. (Note: The percentages in the table add up to 100%.)

Classification results	Percentage of the test sequences
All classifiers agreed and all were correct	95.37%
All classifiers agreed and all were wrong	0.24%
The classifiers disagreed and at least one of them was correct	4.39%
The classifiers disagreed and none of them was correct	0.00%

We also conducted a series of experiments to examine the impact of the parameter values on the performance of the two proposed classifiers. To avoid the mutual influence of parameters, in each experiment we only varied one parameter's values, with the other parameters being fixed and having the values as shown in Table 5.1. Figure 5.2 and 5.3 show the results for DC-1.

We see from the figures that the performance of DC-1 changed substantially when varying *Length*, *Occur*, *Mutation* and k. In 5.3(a) and 5.3(c), NR is high when $Length \leq 9$ or $Mutation \geq 2$. Short active segments (e.g. with $Length = 7$) or segments with high mutation (e.g. $Mutation = 3$) may appear, by chance, in both Alu and non-Alu sequences, and therefore the representative patterns constructed from these segments can not characterize the sequences. In 5.2(b), the performance of DC-1 degrades as *Occur* becomes large. It was observed that the discovered segments in Alu generally have low occurrence numbers (e.g. with $Occur \leq 30$). When $Occur \geq 50$, very few segments were discovered and thus the glued representatives patterns can not characterize the sequences as well.

In 5.2(d), PR drops sharply as k (i.e., the allowed number of mismatches in the gluing algorithm) increases. When k is large, the representative patterns contain many long segments solely composed of the introduced letter `Y`. (We consider it to be a match when aligning `Y` with all of the four nucleotides `A`, `C`, `G` and `T`.) As a result,

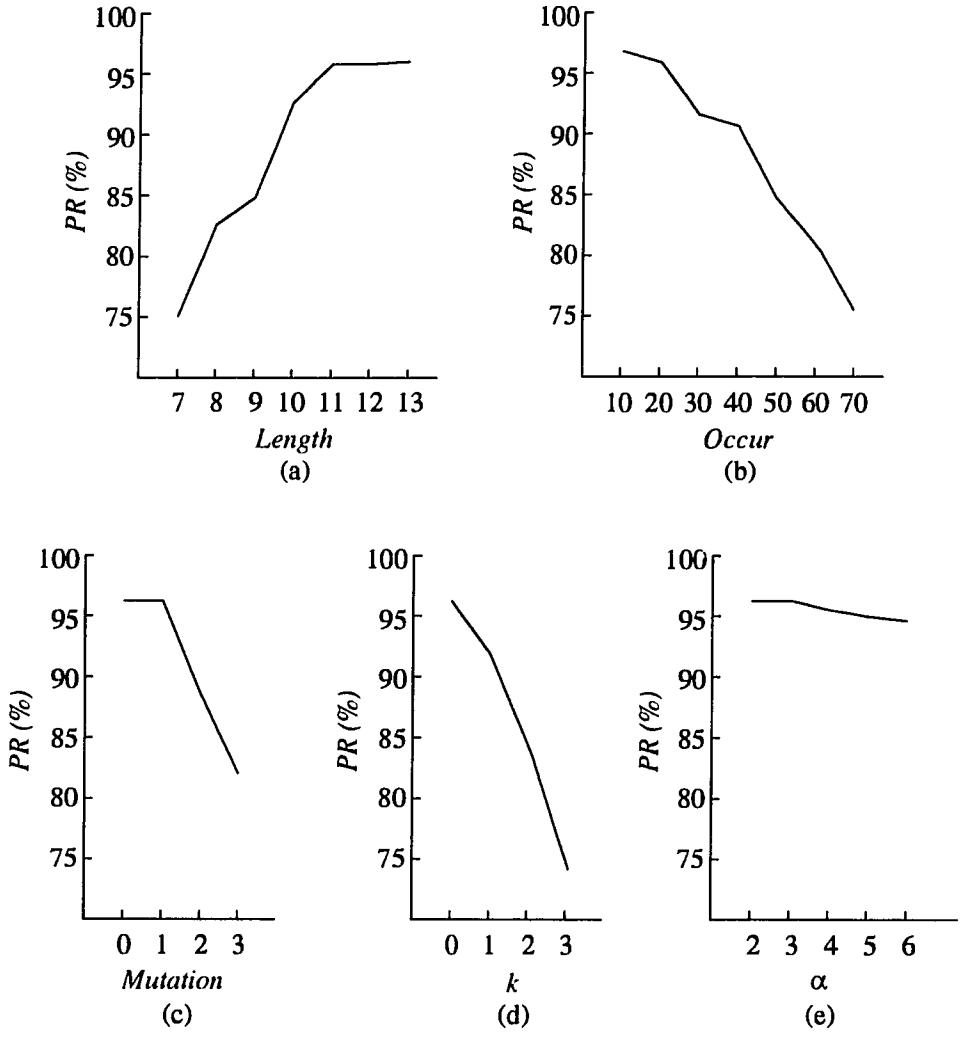

Figure 5.2 PR for DC-1.

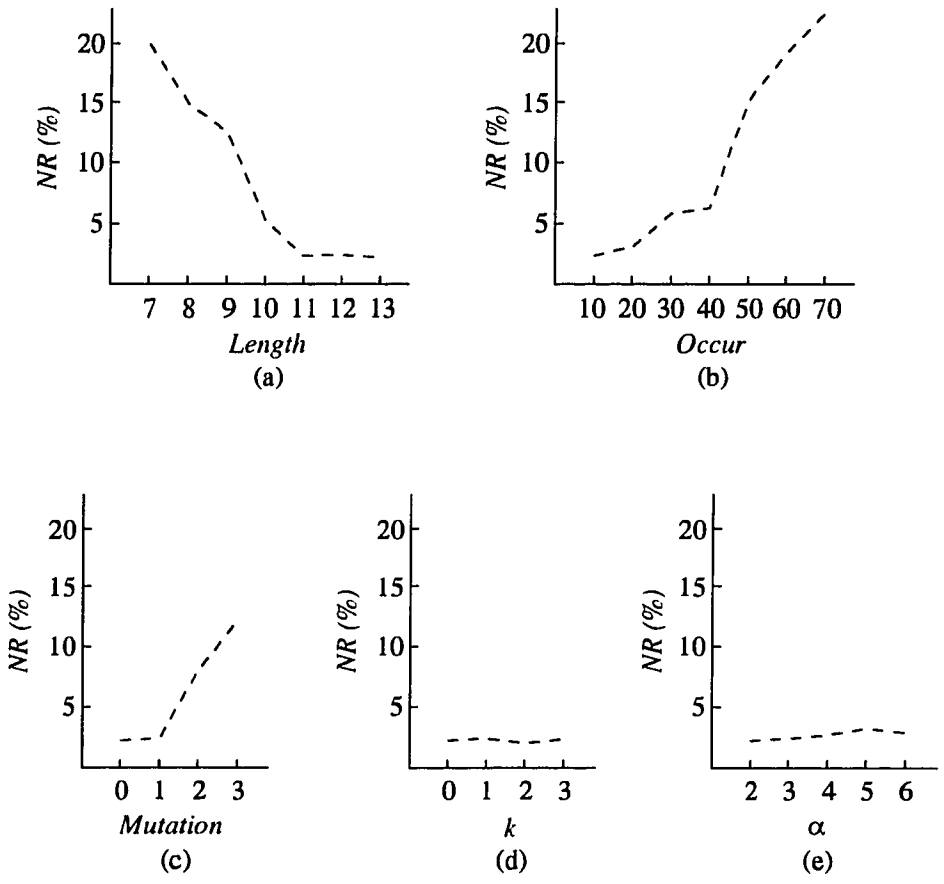

Figure 5.3 NR of the various parameters used in DC-1.

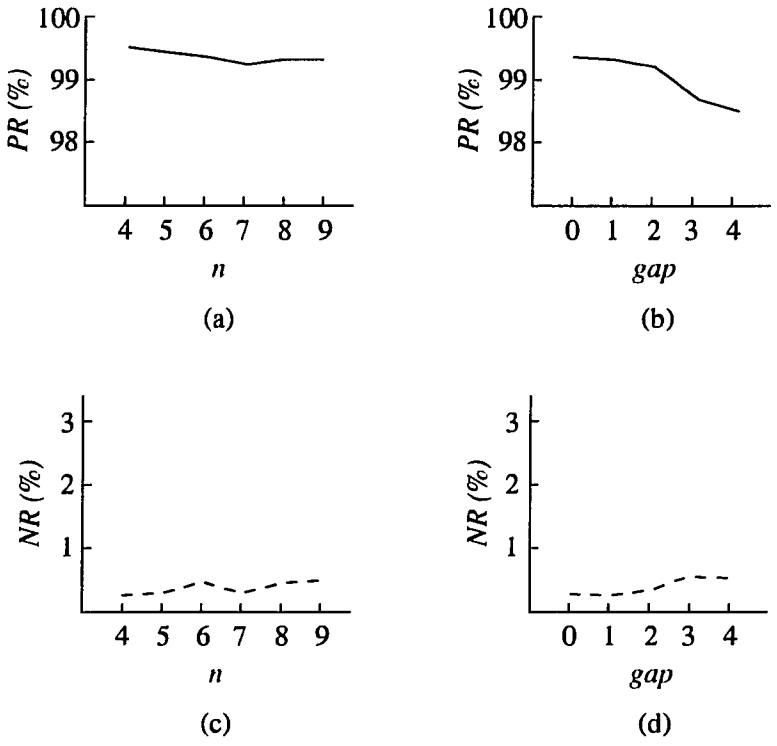

Figure 5.4 Impact of the segment length and *gap* on the performance of DC-2.

both Alu and non-Alu sequences can get the same high score when matching with the representative patterns, making it difficult to distinguish between them.

Figure 5.4 shows the impact of varying the segment length and *gap* on the performance of DC-2. No trend is evident with regard to the two parameters. However, programs using a short segment (e.g., $n = 5$) and a small *gap* (e.g., $gap = 0$) run much faster than programs using a long segment (e.g., $n = 9$) and a large *gap* (e.g., $gap = 4$).

5.3 Discussion

Traditionally, the "consensus sequence" approach has been used to classify DNA sequences, in particular promoters [10, 61, 65, 77, 80]. This approach searchs for consensus sequence (which often appear in a particular position) in DNA. Our

techniques differ from the consensus sequence approach in that we do not look for those positions.

On the other hand, FASTA is basically a alignment-based method. Our classifiers used fingerprints and active patterns instead of aligning two sequences. Our experimental results show that the proposed classifiers are complementary to each other. Using the classifiers together increases the confidence level if they agree on their classification results.

CHAPTER 6

ALGORITHMS FOR DISCOVERING BLOCKS FOR A FAMILY OF PROTEINS

After describing the algorithms for constructing DNA classifiers and the experimental results concerning those algorithms, we now turn to discovery algorithms for proteins.

The most highly conserved regions of a family of proteins can be represented as "blocks" of locally aligned sequence segments. Each block can be considered as a special type of pattern for the protein family. If a query sequence belongs to a family with multiple blocks, then at least a subset of these blocks should score highly when matching the query with the blocks [30, 33].

In this chapter, we present several algorithms to discover blocks for protein families. We focus on the 768 groups of related proteins documented in the PROSITE catalog v. 12.0 [7] (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/prosite") keyed to the SWISS-PROT protein sequence databank version 29 [8] (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/swiss-prot"). Currently, each protein family in the PROSITE catalog is associated with a set of blocks; each block is obtained from ungapped aligned regions extracted from the sequences in the group. A best set of blocks is then selected using a program called PROTOMAT developed by Henikoff and Henikoff of the Howard Hughes Medical Institute, Seattle, Washington [30, 33] (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/blocks/unix/protomat"). All the selected blocks are then calibrated and concatenated into the BLOCKS database [30, 33] (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/blocks/unix"). The overall strategy of the BLOCKS system for constructing a database of blocks is shown in Figure 6.1.

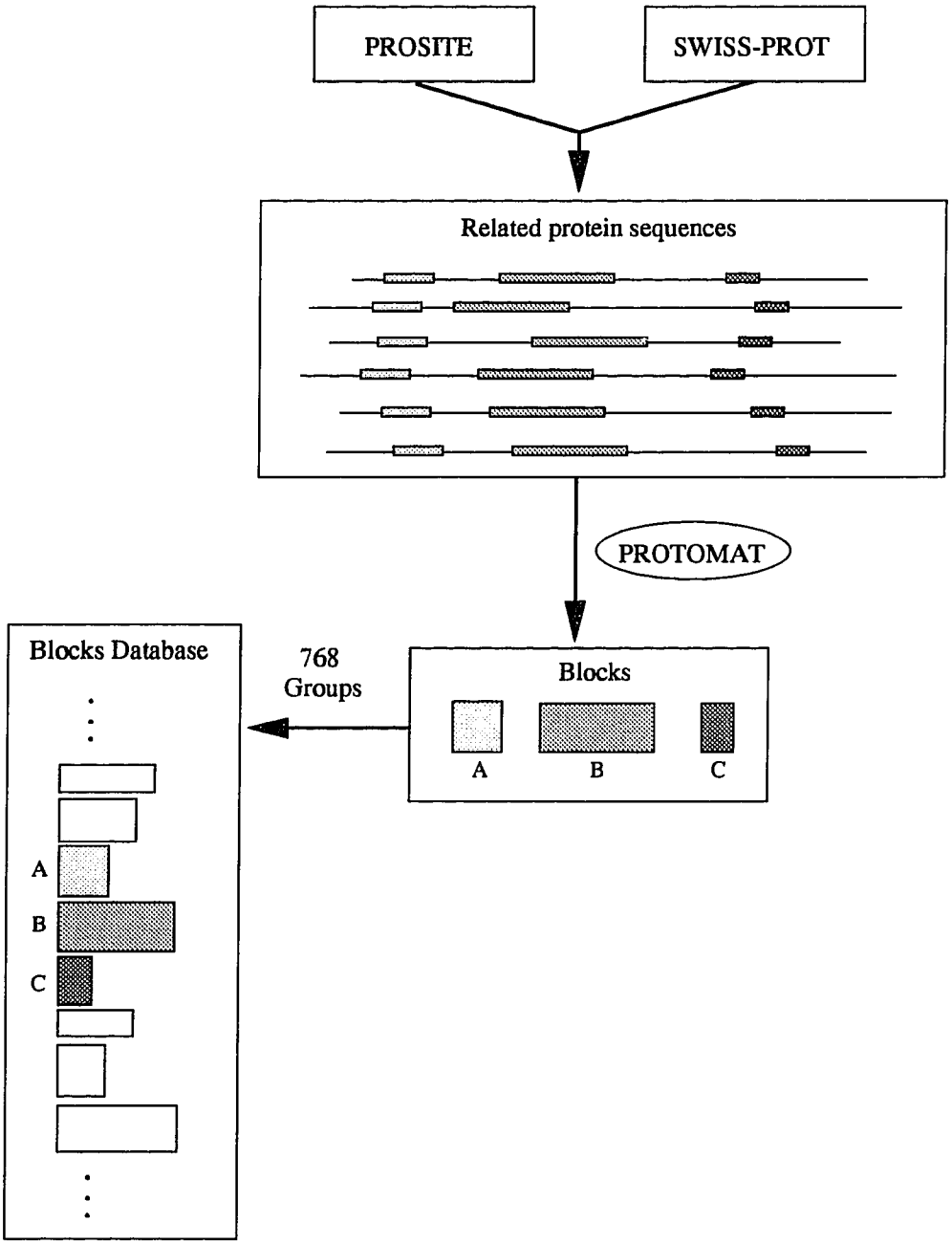

Figure 6.1 Overall strategy of the BLOCKS system [30, 33] for constructing a database of blocks. The PROTOMAT system [30, 33] is applied to a family of protein sequences, resulting in a set of blocks. The BLOCKS database consists of successive application of PROTOMAT to unique groups catalogued in PROSITE [7], including calibration of each block based on the results of searching SWISS-PROT [8].

```
WCATTPNFDQDQRWGYC
WCGTTTDYDTDKLFGYC
WCATTANYDDDRKWGFC
WCATTTNYDDDRKWGFC
WCATTTNYDDDRKWGFC
WCATTANYDRDKLFGFC
WCATTHNYDRDRAWGYC
WCATTANYDRDHEWGFC
WCSTTADYDRDHEWGFC
WCGTTQNYDADQKFGFC
WCGTTQNYDADQKFGFC
WCGTTQNYDADQKFGFC
WCSLSPNYDKDRAWKYC
WCSLSSNYDEDGVWKYC
```

Figure 6.2 An example block.

We start with the blocks stored in the current BLOCKS database and apply a modified version of the algorithm developed by Tatusov, Altschul and Koonin of the National Institutes of Health, Bethesda, Maryland [81, 82] to expand the blocks and produce a set of new blocks.

6.1 Blocks

A block B of a PROSITE group $\mathcal{F}$ is an $N \times L$ matrix in which each row is an ungapped segment of width L [30]. (N is the length and L is the width of the block.) Each segment in B is taken from a distinct sequence in $\mathcal{F}$, i.e., it is a subsequence made up of consecutive amino acids of the sequence. For example, Figure 6.2 shows a 14×17 block (AC#: BL00023) of the Type II fibronectin collagen-binding domain proteins in the PROSITE catalog. In general, $\mathcal{F}$ may have several blocks. There may also exist some sequences in $\mathcal{F}$ which do not have any segment appearing in any of the blocks.

6.2 Transformation of Blocks to Weight Matrices

Like Henikoff and Henikoff's BLOCKS system [30], in calculating the score between the query sequence and a block, we first transform the block to a weight matrix (reminiscent of the profiles described in [27]). A weight matrix W generated from an $N \times L$ block B has 20 rows (one for each possible amino acid) and L columns. Let $W_{m,n}$, $1 \leq m \leq 20$, $1 \leq n \leq L$, be an entry in W. There are several ways to calculate $W_{m,n}$. We first review Henikoff and Henikoff's approach to calculating $W_{m,n}$. Then we describe three new methods for calculating $W_{m,n}$; all of these three methods were suggested by Tatusov *et al.* in their efforts to discover conserved protein segments [81, 82]. Table 6.1 defines the terms and notation we use, where we fix a numbering of the amino acids from 1 to 20.

The BLOCKS tool [34] associates each segment h in a block B with a weight d_h, $1 \leq h \leq N$, and calculate $W_{m,n}$ using the formula:

$$W_{m,n} = 100 \times \frac{R_{m,n}}{\sum_{i=1}^{20} R_{i,n}}$$

where

$$\begin{aligned} R_{m,n} &= \frac{C'_{m,n}/N}{p_m} \\ &= \frac{C'_{m,n}/p_m}{N} \\ &= \frac{C'_{m,n}/p_m}{\sum_{i=1}^{20} C'_{i,n}} \end{aligned}$$

and

$$C'_{m,n} = \sum_{h=1}^{N} \eta_h$$

where

$$\eta_h = \begin{cases} d_h & \text{if } b_{m,n} = m \\ 0 & \text{otherwise} \end{cases}$$

and p_m is the background probability that the m^{th} amino acid occurs in general protein positions [10, 22, 48, 71, 79].

Table 6.1 Notation and meaning.

Notation	Meaning
B	an $N \times L$ block of segments where N is the length and L is the width of the block
$B_{h,n}$	the amino acid at the h^{th} row, $1 \leq h \leq N$, and the n^{th} column, $1 \leq n \leq L$, in the block B
$b_{h,n}$	the number of the amino acid occurring at the position $B_{h,n}$
d_h	sequence weight of the h^{th} segment, $1 \leq h \leq N$, of the block B used by BLOCKS [34], which is normalized so that $\sum_{i=1}^{N} d_i = N$
W	the weight matrix generated from the block B
$W_{m,n}$	the entry at the m^{th} row, $1 \leq m \leq 20$, and the n^{th} column, $1 \leq n \leq L$, in W
$q_{m,n}$	the probability for the m^{th} amino acid, $1 \leq m \leq 20$, to occur in the n^{th} column, $1 \leq n \leq L$, in the block B
p_m	the background probability that the m^{th} amino acid, $1 \leq m \leq 20$, occurs in general protein positions (or the expected frequency of the m^{th} amino acid in a protein database); $\sum_{i=1}^{20} p_i = 1.0$
t_m	the number of times the m^{th} amino acid, $1 \leq m \leq 20$, occurs in the SWISS-PROT database
T	the total number of amino acids in the SWISS-PROT database
C	the occurrence matrix for the block B
$C_{m,n}$	the entry at the m^{th} row, $1 \leq m \leq 20$, and the n^{th} column, $1 \leq n \leq L$, in C, representing the number of times the m^{th} amino acid occurs in the n^{th} column in the block B; $\sum_{i=1}^{20} C_{i,n} = N$ in each column n
$C'_{m,n}$	sequence-weighted count of the number of times the m^{th} amino acid, $1 \leq m \leq 20$, occurs in the n^{th} column, $1 \leq n \leq L$, in the block B used by BLOCKS [34]; $\sum_{i=1}^{20} C'_{i,n} = N$ in each column n
$\vec{C}_n$	the n^{th} column vector, $1 \leq n \leq L$, of the occurrence matrix C
M	a substitution matrix, such as the BLOSUM
$M_{u,v}$	the entry at the u^{th} row, $1 \leq u \leq 20$, and the v^{th} column, $1 \leq v \leq 20$, in M, indicating the similarity between the u^{th} amino acid and the v^{th} amino acid
$S[i,j]$	the segment starting at the i^{th} position and ending at the j^{th} position of the sequence S
a_i	the number of the amino acid occurring at the i^{th} position of the sequence S
$\|S\|$	the total number of amino acids in the sequence S
$\Gamma(x)$	the gamma function

In contrast, we calculate $W_{m,n}$ by considering the probability for the m^{th} amino acid to occur in the n^{th} column in the block B. Specifically, we calculate $W_{m,n}$ using the formula:

$$W_{m,n} = \log \frac{q_{m,n}}{p_m}.$$

We let $p_m = t_m/T$. Our methods differ in estimating the $q_{m,n}$. The first algorithm estimates $q_{m,n}$ using the Bayesian formula:

$$q_{m,n} = \frac{C_{m,n} + D \times p_m}{N + D}$$

and chooses $D = \sqrt{N}$, a parameter value suggested by Lawrence *et al.* [48].

The second algorithm modifies the first algorithm's formula by taking into account the similarity scores, such as the PAM [21, 40, 72] or BLOSUM [31, 32], between the amino acids. (In the study presented here, we adopted the BLOSUM 62 substitution matrix as shown in Figure 6.3. In Figure 6.3, for instance, the cost for substituting an amino acid A by an amino acid R is -2. Note that the numbers in this matrix can be shifted in such a way that all the numbers become positive.) The algorithm estimates $q_{m,n}$ as

$$q_{m,n} = \frac{C_{m,n} + D \times \frac{p_m}{N} \sum_{i=1}^{20} C_{i,n} e^{\lambda M_{i,m}}}{N + D}$$

where λ is the natural scale for the substitution matrix [44].

The third algorithm estimates $q_{m,n}$ using a mixture of multiple Dirichlet distributions [13]. Here we assume that the amino acids in the n^{th} column of the block B are generated independently at random according to an underlying probability distribution $\vec{q} = (q_1, \ldots, q_{20})$ over the 20 amino acids, where $\vec{q}$ is chosen independently from a Dirichlet mixture density ρ of the form

$$\rho = \beta_1 \rho_1 + \ldots + \beta_k \rho_k.$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	6	-2	-2	-3	-1	-1	-1	0	-2	-2	-2	-1	-1	-3	-1	2	0	-4	-3	0
R	-2	8	-1	-2	-5	1	0	-3	0	-4	-3	3	-2	-4	-3	-1	-2	-4	-3	-4
N	-2	-1	8	2	-4	0	0	-1	1	-5	-5	0	-3	-4	-3	1	0	-6	-3	-4
D	-3	-2	2	9	-5	0	2	-2	-2	-5	-5	-1	-5	-5	-2	0	-2	-6	-5	-5
C	-1	-5	-4	-5	13	-4	-5	-4	-4	-2	-2	-5	-2	-4	-4	-1	-1	-3	-4	-1
Q	-1	1	0	0	-4	8	3	-3	1	-4	-3	2	-1	-5	-2	0	-1	-3	-2	-3
E	-1	0	0	2	-5	3	7	-3	0	-5	-4	1	-3	-5	-2	0	-1	-4	-3	-4
G	0	-3	-1	-2	-4	-3	-3	8	-3	-6	-5	-2	-4	-5	-3	0	-2	-4	-5	-5
H	-2	0	1	-2	-4	1	0	-3	11	-5	-4	-1	-2	-2	-3	-1	-3	-4	3	-5
I	-2	-4	-5	-5	-2	-4	-5	-6	-5	6	2	-4	2	0	-4	-4	-1	-4	-2	4
L	-2	-3	-5	-5	-2	-3	-4	-5	-4	2	6	-4	3	1	-4	-4	-2	-2	-2	1
K	-1	3	0	-1	-5	2	1	-2	-1	-4	-4	7	-2	-5	-2	0	-1	-4	-3	-3
M	-1	-2	-3	-5	-2	-1	-3	-4	-2	2	3	-2	8	0	-4	-2	-1	-2	-1	1
F	-3	-4	-4	-5	-4	-5	-5	-5	-2	0	1	-5	0	9	-5	-4	-3	1	4	-1
P	-1	-3	-3	-2	-4	-2	-2	-3	-3	-4	-4	-2	-4	-5	11	-1	-2	-5	-4	-4
S	2	-1	1	0	-1	0	0	0	-1	-4	-4	0	-2	-4	-1	6	2	-4	-3	-2
T	0	-2	0	-2	-1	-1	-1	-2	-3	-1	-2	-1	-1	-3	-2	2	7	-4	-2	0
W	-4	-4	-6	-6	-3	-3	-4	-4	-4	-4	-2	-4	-2	1	-5	-4	-4	16	3	-4
Y	-3	-3	-3	-5	-4	-2	-3	-5	3	-2	-2	-3	-1	4	-4	-3	-2	3	10	-2
V	0	-4	-4	-5	-1	-3	-4	-5	-5	4	1	-3	1	-1	-4	-2	0	-4	-2	6

Figure 6.3 The BLOSUM 62 substitution matrix (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/blocks/unix/blosum").

Each ρ_j is a Dirichlet density, called a component of the mixture, and $\beta_1, \ldots, \beta_k$ are mixture coefficients, which are positive numbers and sum to one.

Let $\alpha_i^{(j)}$, $1 \leq i \leq 20$, be unknown parameters for the Dirichlet density ρ_j. The value of ρ_j at a particular point $\vec{q}$ is given by:

$$\rho_j(\vec{q}) = \frac{\prod_{i=1}^{20} q_i^{\alpha_i^{(j)}-1}}{Z}$$

where Z is the normalizing constant such that ρ_j integrates to unity. Using the standard expectation-maximization algorithm [13, 23], one can estimate the k mixture coefficients β_j, $1 \leq j \leq k$, and the Dirichlet parameter vectors $\alpha_1^{(j)}, \ldots, \alpha_{20}^{(j)}$. (In the study presented here, we adopted the same β, α and k, which was set to 8, as used in Tatusov *et al.*'s work [13, 81].)

Let $Prob(j|\vec{C}_n)$ denote the probability that the distribution $\vec{q}$ that produced the observed counts $\vec{C}_n$ was chosen from the j^{th} component of the Dirichlet mixture. Using Bayes rule,

$$Prob(j|\vec{C}_n) = \frac{\beta_j Prob(\vec{C}_n|\rho_j)}{\sum_{l=1}^{k} \beta_l Prob(\vec{C}_n|\rho_l)}.$$

When $s = \sum_{i=1}^{20} C_{i,n}$ and $\alpha^{(j)} = \sum_{i=1}^{20} \alpha_i^{(j)}$,

$$Prob(\vec{C}_n|\rho_j) = \frac{\Gamma(s+1)\Gamma(\alpha^{(j)})}{\Gamma(s+\alpha^{(j)})} \prod_{i=1}^{20} \frac{\Gamma(C_{i,n}+\alpha_i^{(j)})}{\Gamma(C_{i,n}+1)\Gamma(\alpha_i^{(j)})}.$$

The third algorithm estimates $q_{m,n}$ as

$$q_{m,n} = \sum_{j=1}^{k} Prob(j|\vec{C}_n) \frac{C_{m,n}+\alpha_m^{(j)}}{\sum_{i=1}^{20}(C_{i,n}+\alpha_i^{(j)})}$$

6.3 Extension of Blocks Using a Statistical Approach

In general, it is desirable to have the blocks contain as many diverse segments as possible. Here, we start with the blocks in the BLOCKS database version 8.0 developed by Henikoff and Henikoff and expand them by using a modified version of the iterative algorithm developed by Tatusov *et al* [81, 82]. For each $N \times L$ block B of a group $\mathcal{F}$ in the BLOCKS database, we slide its weight matrix W along each sequence S in $\mathcal{F}$ and align W with every segment of width L in S. The score obtained by aligning W with the segment $S[i+1, i+L]$, $0 \leq i \leq |S| - L$, is

$$\sum_{j=1}^{L} W_{a_{i+j},j}.$$

If the segment $S[i+1, i+L]$ is not in the block B but gets a score higher than a pre-determined *cutoff* value, we expand B by appending the segment to it provided that none of the current segments in B is taken from S. This results in a new block and therefore a new weight matrix. We then align this new matrix with every segment taken from the sequences in the group $\mathcal{F}$ again. This procedure is repeated until the block B cannot be expanded any further.

Let $P_k(x)$ be the probability to get score x from the first kth columns in W. Then

$$P_k(x) = \sum_{i=1}^{20} P_{k-1}(x - W_{i,k})p_i$$

where the initialization is given by

$$P_1(x) = \sum_{i=1}^{20} \{p_i | W_{i,1} = x\}$$

If the total number of segments in the family is g, the expect number of segments with score x is $P_L(x) \times g$. For a given *cutoff* score, a segment is called *false positive* if its score according to W is greater than *cutoff* but this segment is not related to W. That is, it gets the score by chance. On the other hand, a segment is called *true positive* if its score is greater than *cutoff* and is related to W. Let F be the

number of false positive segments, T be the number of true positive segments and G be the total number of segments with score greater than *cutoff*. We have $G = T + F$ and estimate F as $P_{I,}(x > cutoff) \times g$. By sliding W along all the sequences in the family, we can get G directly. The ratio $R = F/T$ is the parameter used to set *cutoff*. In general a smaller R causes a higher *cutoff* value and limits the growth of the block but makes the blocks predictions more conservative. In our work, the R was set to 0.02, chosen based on Tatusov *et al.*'s experience [81].

CHAPTER 7

ALGORITHMS FOR PROTEIN CLASSIFICATION

The blocks developed in Chapter 6 can be used for classifying proteins. Protein classification is the activity of assigning a given unlabeled protein sequence (or a query) into an appropriate protein family. To facilitate the classification task, each family is often associated with some representatives [6]. Typical representatives may include profiles [27, 78, 83], AACC (amino acid class covering) patterns [76], consensus patterns [37, 63], blocks [30, 33], etc. To classify the given query sequence, one compares the sequence with the family representatives and finds the most relevant family.

As described in the beginning of Chapter 6, we are interested in the proteins in the PROSITE catalog keyed to the SWISS-PROT protein sequence databank [8]. Currently the best classifier for these proteins is the BLOCKS database developed by Henikoff and Henikoff of the Howard Hughes Medical Institute Seattle, Washington [30]. To classify a query sequence, the BLOCKS system uses a program, called BLIMPS (formerly PATMAT, which can be obtained from the ftp site under the directory "/repository/blocks/unix/blimps"), to align the query with all the blocks in the database and display a collection of blocks, ranked based on their relevance to the query [87]. The classifier can analyze the results of BLIMPS by collecting the alignments for individual blocks belonging to a group and evaluating the group as a whole using a statistical technique [33]. Figure 7.1 shows the BLOCKS and BLIMPS system diagram.

In this chapter, we propose 4 protein classifiers based on the algorithms described in previous chapters. The first classifier uses the pattern discovery algorithm in Chapter 2, combined with the fingerprint algorithm in Chapter 4. The other classifiers use the block-based algorithms presented in Chapter 6.

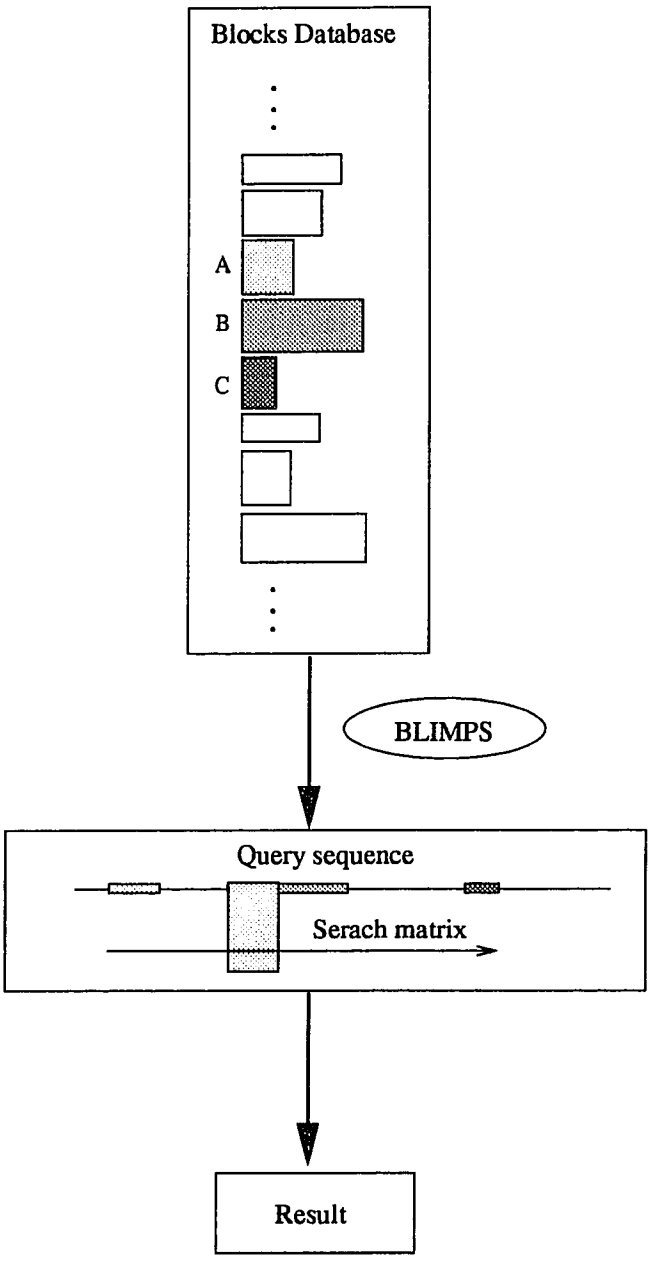

Figure 7.1 BLIMPS [33] converts each block to a search matrix and scores all possible alignments of the query with all blocks in the database, saving the top scoring alignments in rank order.

7.1 Motif-Fingerprint Protein Classifier (PC-1)

We apply our pattern discovery tool DISCOVER (cf. Chapter 2 and 3) to all 768 groups of related proteins documented in the PROSITE catalog keyed to the SWISS-PROT protein sequence databank [8]. We select 70% of the sequences in each group at random to serve as a training sample. Then process the training sequences in two ways:

- Find 50 characteristic patterns from the training sample of each group. The patterns are regular expressions with the form $*X*$. They 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.[1] To reduce the effect made by 'chance patterns,' we associate each characteristic pattern with a weight based on Zipf's Law [96]. If a pattern occurs in m groups, its weight is assigned as $\log_2\lceil(M/m)\rceil$, where M is the total number of groups, 768 in our case.

- Hash the training sequences using the gapped fingerprint technique (cf. Section 4.2). The length of the fingerprints are 5 and *gap* is 1.

A scoring scheme is then developed for comparing the query sequence with all the characteristic patterns. When classifying a query sequence T, we first compare T with all the characteristic patterns. After comparison, each group obtains a raw score, which equals the sum of the weights of the group's characteristic patterns occurring in T. The raw score for a group is normalized by dividing it by the total weight of all the characteristic patterns 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

[1] We choose this length and this number of patterns because this seems to give good results. These decisions can be changed easily and are compile-time parameters of our system.

greater than an experimentally determined threshold. (In the study presented here, the threshold was set to 20. Our experimental results showed that about 65% of our test sequences obtained a score higher than the threshold.) Otherwise we proceed to the second phase.

In the second phase, we hash T, 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. If two sequences have the same highest vote, the shorter one is favored. We refer to this classifier as PC-1.

7.2 Protein Classifiers Using Block-based Algorithms (PC-2, PC-3 and PC-4)

Given a query sequence S and a database of blocks, our classifiers calculate a score between S and every block B in the database by sliding B's weight matrix W along S and aligning W with each segment $S[i+1, i+L]$, $0 \leq i \leq |S| - L$, in S. Each alignment results in a score, as calculated in the previous subsection. Let δ_i denote the score obtained by aligning W with the segment $S[i+1, i+L]$. The score between the sequence S and B is defined as

$$\max_i \{\delta_i | 0 \leq i \leq |S| - L\}.$$

Our classifiers give the highest rank to the group $\mathcal{F}$ containing the highest scoring block and assign the query sequence S to $\mathcal{F}$.

We refer to the classifier using the Bayesian formula as PC-2, the classifier using the Bayesian formula with the BLOSUM substitution matrix as PC-3 and the classifier using the Dirichlet distribution approach as PC-4 (cf. Chapter 6).
Remark: In contrast to our algorithm, Henikoff and Henikoff [30, 33] use a slightly different procedure for classification. Specifically, they associate each block B with a calibrated value c and calculate the score between the sequence S and B as follows:

$$\max_{-(L-2)\leq i\leq |S|} \frac{\sum_{j=1}^{L} W_{a_{i+j-1},j} \times 1000}{c}$$

where $W_{a_{i+j-1},j} = 0$ if $i + j - 1 \leq 0$ or $i + j - 1 > |S|$.

CHAPTER 8

EXPERIMENTAL RESULTS OF PROTEIN CLASSIFICATION

8.1 Block Database Construction Using PC-2, PC-3 and PC-4

We applied the iterative algorithms described in Chapter 6 to expand the blocks in the BLOCKS database version 8.0 [30]. There are 2,884 blocks in the version 8.0 database. Our algorithms extended the lengths of these blocks, without changing their widths and the total number of the blocks. The results were three databases of blocks, one for each classifier. It took about 2 hours to generate a database.

Table 8.1 shows the statistics, for each classifier respectively, concerning the growth rates (GR) of the blocks in the BLOCKS database. The growth rate GR for a block B is defined as $(N_f - N_i)/(N_i) \times 100\%$ where N_i is the initial length of B and N_f is the final length of B after expanding it. It can be seen that about 20% of the blocks grew over one fourth of their original size (with $GR \geq 25\%$). Among the blocks, the block with AC# BL00282 in the Kazal serine protease inhibitors family grew the largest ($N_i = 24$ and $N_f = 86$).

Table 8.1 Growth rates for the blocks in the BLOCKS database after applying the proposed iterative algorithm to expand them.

GR	PC-2	PC-3	PC-4
0%	1,269	1,260	1,254
0% - 25%	1,061	1,063	1,077
25% - 50%	396	404	401
50% - 75%	87	85	86
75% - 100%	42	43	38
100% - 200%	21	21	23
200% - 300%	8	8	5

8.2 Classification of Proteins in the PROSITE Groups

We next compared the relative performance of the four proposed classifiers with the previously published one: Henikoff and Henikoff's BLOCKS classifier [30, 33] which is accessible via the Internet. User can send an e-mail with subject "HELP" to "blocks@howard.fhcrc.org" to obtain the information about the BLOCKS classifier. We applied all the five classifiers to the 768 groups of related proteins in the PROSITE catalog v. 12.0. There are, in total, 16,823 different sequences in the groups. (We did not consider the Heat shock hsp20 proteins family profile (AC#: PS01031) and Globins profile (AC#: PS01033) in the experiment, as these two groups are special in that PROSITE provides them with a Gribskov-like profile rather than a PROSITE pattern.)

We selected 70% of the sequences in each group at random to serve as a training sample and used the other 30% as test sequences. In running the PC-1, we found the characteristic motifs and gaped fingerprints from the 70% training sequences. However, the blocks databases were built from all (training as well as test) sequences [30].

Then, we checked whether the 30% test sequences were classified correctly according to the five studied classifiers (i.e. BLOCKS: developed by Henikoff and Henikoff; PC-1: developed based on the motif-fingerprint algorithm; PC-2: developed based on the blocks obtained from the Bayesian formula; PC-3: developed based on the blocks obtained from the Bayesian formula with the BLOSUM substitution matrix; PC-4: developed based on the blocks obtained from the Dirichlet distribution approach). A test sequence is said to be classified correctly if its PROSITE group is hit, i.e., ranked highest, by the corresponding classifier (assuming the classifications in the PROSITE catalog are all correct). It took about 50 hours for each method to classify all the test sequences. Note that this experiment favors block-based methods

Table 8.2 Precision rates for the five studied classifiers. Note that given a query sequence, the BLOCKS classifier displays a collection of blocks, ranked based on their relevance to the query. We considered the group containing the block ranked highest by PATMAT [30, 87] as the one hit by the classifier.

Methods	N_c	PR
BLOCKS	4,843	96.0%
PC-1	4,653	92.2%
PC-2	4,873	96.6%
PC-3	4,892	96.9%
PC-4	4,857	96.3%

since the blocks databases are built from all sequences including the 30% that the PC-1 treats as unknowns.

Table 8.2 summarizes the classification results. The measure used to evaluate the effectiveness of the classifiers is the precision rate (PR), defined as $(N_c/N_t) \times 100\%$. N_c is the number of test sequences classified correctly; N_t is the total number of the test sequences, 5,046 in our case. (In the PROSITE catalog, it is possible that a protein sequence is placed in more than one group. In that case, the sequence is said to be classified correctly by a classifier, if the classifier hits any one of these groups.) The table shows that the new blocks are more diagnostic than those in the BLOCKS database.[1]

We next examined when the classifiers agreed and disagreed on their rankings. The five classifiers are said to agree on the rankings of a test sequence, if all classifiers assign the sequence to the same group. The five classifiers are said to disagree on the rankings of a test sequence, if at least two classifiers assign the sequence to different groups. Table 8.3 summarizes the results. The table shows that when the five classifiers agree, the classification has a high likelihood of being correct. Specifically, the correct agreed-upon classification divided by all (incorrect as well as correct)

[1] We also evaluated the effectiveness of applying the fingerprint method alone, as opposed to combining the motifs and fingerprints like that used in the PC-1, to classify the test sequences. Its precision rate was about 85%.

Table 8.3 Complementarity between the five studied classifiers. (Note: The percentages in the table add up to 100%.)

Classification results	Percentage of the test sequences
All classifiers agreed and all were correct	90.59%
All classifiers agreed and all were wrong	0.10%
The classifiers disagreed and one of them was correct	9.21%
The classifiers disagreed and all were wrong	0.10%

agreed-upon classification is 90.59%/(90.59% + 0.10%) = 99.89%. On the other hand, if the classifiers disagree, then the likelihood that one is right is 9.21%/(9.21% + 0.10%) = 98.93%.

It is interesting to note that every sequence of the 0.1% of test data (5 sequences) that were misclassified was assigned to the same family by all the five methods. Table 8.4 lists the SWISS-PROT ID for these sequences, their descriptions, the sequences' original groups documented in the PROSITE catalog and their groups hit by the classifiers. The table suggests that the sequences' family memberships should be re-examined or their PROSITE groups have close relationships to those hit by the classifiers.

8.3 Discussion

The techniques presented in this chapter are an attempt towards automatic classification of protein sequences. Both the blocks and weighted characteristic motifs can be considered as protein family representatives. When classifying a given query sequence, comparing the query with the representatives is much faster than comparing the query with every sequence in the family. (In general, it takes about 50 seconds to classify a given query sequence using the five studied methods once the representatives are set up.) Furthermore, the experimental results reported

Table 8.4 The 5 misclassified protein sequences in SWISS-PROT 29. For these sequences, their groups documented in the PROSITE catalog differ from those hit by the five studied classifiers.

SWISS-PROT ID	Protein description	Group AC#	Group description	Hit group AC#	Hit group description
COG7_HUMAN	matrilysin precursor (EC 3.4.24.23) (PUMP-1 protease) (uterine metalloproteinase) (matrix metalloproteinase-7) (MMP-7) (matrin)	PS00546	matrixins cysteine switch	PS00024	hemopexin domain signature
GLNA_METVO	glutamine synthetase (EC 6.3.1.2) (glutamate-ammonia ligase)	PS00180	glutamine synthetase signature 1	PS00182	glutamine synthetase class-I adenylation site
GUN5_THEFU	endoglucanase E-5 precursor (EC 3.2.1.4) (ENDO-1,4-beta-glucanase E-4) (cellulase E-5)	PS00659	glycosyl hydrolases family 5 signature	PS00561	cellulose-binding domain, bacterial type
MCAS_MYCBO	mycocerosic acid synthase	PS00012	phosphopantetheine attachment site	PS00606	beta-ketoacyl synthases active site
MTB1_BACAM	modification methylase bamhi (EC 2.1.1.113) (N-4 cytosine-specific methyltransferase bamhi) (M.bamhi)	PS00092	N-6 adenine-specific DNA methylases signature	PS00093	N-4 cytosine-specific DNA methylases signature

demonstrated that such classification methods achieve considerably high precision rates.

The three statistical methods used for computing the weight matrices are suggested by Tatusov, Altschul and Koonin of the National Library of Medicine [81]. However, those authors were concerned with discovering blocks of conserved segments, rather than classifying protein sequences. For each motif explored, they applied iterative database searches which converged on an aligned block of segments containing the motif. The different purposes and usages of the statistical analyses led to different results. Tatusov *et al.* showed empirically that, among the three statistical methods, the one using a mixture of Dirichlet distributions was most effective in terms of the discriminating power for computing weight matrices. On the other hand, our results indicated that all the three statistical methods have comparable performance for protein classification, with the method exploiting the similarities among the amino acids and the Bayesian formula being slightly better than the other two methods (cf. Table 8.2). These different results occur probably due to two reasons. First, our background probability p_m is calculated by considering the proteins in the SWISS-PROT database (cf. Table 6.1), which differs from that used by Tatusov *et al.* Second, we start with the blocks in the BLOCKS database, whereas Tatusov *et al.* constructed their initial blocks using local alignments obtained by applying BLAST [1, 2, 26] to related protein sequences.

The techniques of PC-2, PC-3 and PC-4 follow in spirit the BLOCKS database approach developed by Henikoff and Henikoff [30, 33]. While both approaches exploit the block searching techniques, they differ in two significant ways.

First, the all-or-none classification approach proposed here displays only one group, i.e., the one containing the highest scoring block, for a given query sequence. On the other hand, the BLOCKS classifier hits multiple blocks with high scores [30] and displays the chance probability of the sequence aligning correctly with the blocks

representing a group [33]. Thus our output is relatively simple, but can enhance the confidence level when used together with the BLOCKS classifier.

Second, the BLOCKS classifier incorporates a sequence weight d_i into the calculation of weight matrices and considers the background probability for amino acids to occur in general protein positions (cf. Section 6.2). In contrast, our classifiers analyze not only the background probability but also the probability for amino acids to occur in a particular position of a block, without considering any sequence weights. By combining the statistical analyses and an iterative algorithm for expanding the blocks, we were able to improve the quality of the blocks and weight matrices, thus enhancing precision rates of classification. This performance improvement occurs probably because a better weight matrix expands a block into a better one and a better block generates an even better weight matrix. (We have also run our classification algorithm on the test sequences using the original blocks in the BLOCKS database. The precision rate was about 95.7%, lower than those of using expanded blocks, indicating the significance of the block expansion procedure.)

Notice that while some of the newly included segments are fragments which were purposely left out by the BLOCKS classifier, many come from nonfragmentary sequences. An example is the BL00038A block in the group Myc-type helix-loop-helix DNA-binding domain proteins sign (AC#: PS00038). Its initial length is 87 (i.e., it contains 87 segments in the BLOCKS database). After expanding the block by CP-3, its length becomes 102. Out of the 15 newly included segments, 9 are fragments. The other 6 nonfragmentary sequences include hairy protein (HAIR_DROME) and myc transforming proteins (MYC_AVIM2, MYC_AVIMC, MYC_AVIMD, MYC_AVIOK and MYC_FLVTT).

We have used the BLOSUM 62 substitution matrix [31, 32] to calculate the similarity scores between amino acids. Our empirical study indicated that using a less diverged matrix may improve classification results. For example, the precision

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	11	5	8	8	5	7	8	9	5	7	5	6	6	4	9	9	9	1	4	8
R	5	14	7	5	4	9	5	4	9	6	4	10	7	4	7	7	6	9	2	5
N	8	7	12	10	3	8	9	8	10	6	4	9	5	4	6	9	8	3	6	5
D	8	5	10	13	1	9	11	8	8	5	3	7	4	1	6	8	7	0	3	5
C	5	4	3	1	17	1	1	3	4	5	1	1	2	2	5	7	5	0	7	6
Q	7	9	8	9	1	14	10	5	11	5	6	8	7	2	8	6	6	2	3	5
E	8	5	9	11	1	10	13	7	7	5	4	7	4	2	7	7	6	0	4	5
G	9	4	8	8	3	5	7	13	4	4	3	5	4	3	6	9	7	0	2	6
H	5	9	10	8	4	11	7	4	15	4	5	6	4	6	7	6	5	3	7	5
I	7	6	6	5	5	5	5	4	4	14	9	6	9	8	5	6	8	1	6	11
L	5	4	4	3	1	6	4	3	5	9	13	4	11	8	5	4	5	3	5	9
K	6	10	9	7	1	8	7	5	6	6	4	13	8	2	6	7	7	3	2	4
M	6	7	5	4	2	7	4	4	4	9	11	8	16	7	5	6	7	1	4	9
F	4	4	4	1	2	2	2	3	6	8	8	2	7	16	3	5	4	7	12	5
P	9	7	6	6	5	8	7	6	7	5	5	6	5	3	14	9	7	1	2	6
S	9	7	9	8	7	6	7	9	6	6	4	7	6	5	9	11	10	6	5	6
T	9	6	8	7	5	6	6	7	5	8	5	7	7	4	7	10	12	2	5	8
W	1	9	3	0	0	2	0	0	3	1	3	3	1	7	1	6	2	20	7	0
Y	4	2	6	3	7	3	4	2	7	6	5	2	4	12	2	5	5	7	16	5
V	8	5	5	5	6	5	5	6	5	11	9	4	9	5	6	6	8	0	5	13

Figure 8.1 The PAM 120 substitution matrix (which can be obtained from the ftp site "ncbi.nlm.nih.gov" under the directory "/repository/blocks/unix/patmat").

rate for PC-3 changes from 96.9% to 97.2% after replacing BLOSUM 62 by a PAM 120 matrix. The PAM 120 matrix is shown in Figure 8.1. In Figure 8.1, for instance, the cost for substituting an amino acid A by an amino acid R is 5. This result shows that the effectiveness of the classification methods depends on not only the probability formulae, but also the substitution scores used in calculating the weight matrices.

It is worth pointing out that the four block-based methods studied here, while achieving similar precision rates, misclassify substantially different sets of sequences. Referring to Table 8.2, PC-2 misclassifies 173 sequences, PC-3 misclassifies 154 sequences, PC-4 misclassifies 189 sequences and the BLOCKS classifier misclassifies 203 sequences. Out of these four sets of misclassified sequences, only 41 are common

in all the four sets. Referring to Table 8.3, when the four classifiers, together with PC-1, agree on a result, the likehood of the result being correct is almost 100%. This is higher than those obtained from using any single block-based classifier with PC-1 (the motif-fingerprint method). We have repeated our experiments 10 times, each time using a different randomly chosen set of sequences as the training data. The results are consistent with those of Tables 8.2 and 8.3.

In the PROSITE catalog, it is possible that a sequence belongs to multiple groups. In that case, a classifier is said to correctly classify the sequence, if it hits any of these groups. From Table 8.4, we see that GLNA_METVO is listed as a glutamine synthetase signature 1 and is detected as a glutamine synthetase class-I adenylation site. This may not be a new discovery because this sequence belongs to these two groups (though it is not explicitly placed in both groups in PROSITE). However, membership of the other sequences in the table and their group relationships may merit further study.

With the rapid growth in sequence database sizes, we anticipate that block searching techniques will become increasingly important in determining the biological function of a newly determined protein sequence. The methods proposed in this chapter are able to provide complementary information to existing ones. By combining these tools, biologists can obtain either high confidence classifications or alternative hypotheses.

CHAPTER 9

PATTERN DISCOVERY AND CLASSIFICATION TOOLS

We have developed an e-mail server DISCOVER-CLASSIFY version 2.0 accessible on the Internet for sequence discovery as well as protein and DNA classification. The user can send the queries via e-mail to discover@village.njit.edu and get on-line responses.

The DISCOVER-CLASSIFY e-mail server provides two tools: DISCOVER and CLASSIFY. It supports:

- Discovery of active motifs in sequence database;
- Classification of a protein sequence into appropriate (sub)families in PROSITE catalog v. 12.0 keyed to the SWISS-PROT protein sequence databank version 29.
- Classification of a DNA sequence and tells if it is an Alu or a non-Alu sequence.

Figure 9.1 shows the system components of the DISCOVER-CLASSIFY e-mail server.

9.1 General Information of DISCOVER-CLASSIFY

9.1.1 Obtaining Help

To obtain the general information concerning our server, send a blank message with the single word "HELP" on the subject line as follows:

To: discover@village.njit.edu

Subject: HELP

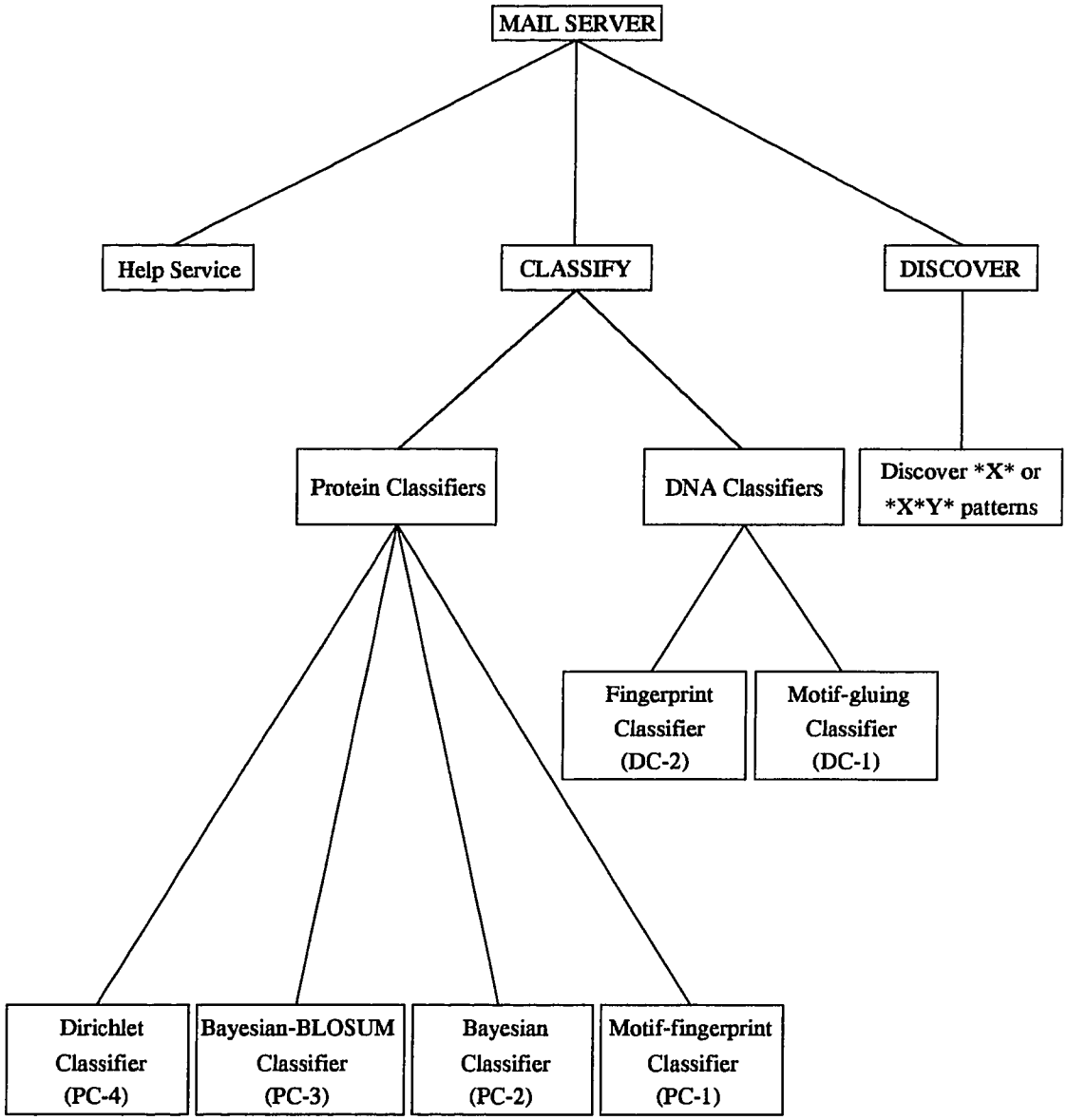

Figure 9.1 DISCOVER-CLASSIFY system components.

9.1.2 Obtaining Software

For information on availability of Sun SPARCstation executable programs used by discover-classify, send a blank message with the single word "software" on the subject line as follows:

To: discover@village.njit.edu

Subject: software

9.1.3 Obtaining On-line Reprints of Papers

Users can obtain reprints (in PostScript) of relevant papers by sending a message with the single word "paper" on the subject line and a body containing:

SIGMOD-94 Returns to the originator of the request a copy of the paper that describes the algorithms used by the DISCOVER and CLASSIFY tools (this paper appeared in ACM SIGMOD Record, Vol. 23, No. 2, June 1994, pp. 115-125; also appeared in Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 1994)

NAR-94 Returns to the originator of the request a copy of the paper that describes the application of the tools to discover motifs in protein families and to classify proteins in the PROSITE groups (this paper appeared in Nucleic Acids Research, Vol. 22, No. 14, 1994, pp. 2769-2775)

PE-96 Returns to the originator of the request a copy of the paper that describes protein classification in the PROSITE groups (this paper will appear in Protein Engineering, 1996)

SEKE-96 Returns to the originator of the request a copy of the paper that describes a system for pattern matching and discovery in scientific databases (this paper will appear in Proceedings of the Eighth International Conference on Software Engineering and Knowledge Engineering, Lake Tahoe, Nevada, June 1996)

9.2 Accessing the DISCOVER Tool

9.2.1 Request Format for DISCOVER

Send a message with the single word "sequence discovery" on the subject line as follows:

To: discover@village.njit.edu

Subject: sequence discovery

A typical message-body looks like this (see explanation below):

```
MOTIF FORM  1          (mandatory)
MAXIMUM LENGTH  12     (optional)
MINIMUM LENGTH  10     (optional)
OCCURRENCE NUMBER  2   (optional)
MUTATION  1            (optional)
BEGIN                  (mandatory)
>FA10_BOVIN COAGULATION FACTOR X PRECURSOR (EC 3.4.21.6)
MAGLLHLVLLSTALGGLLRPAGSVFLPRDQAHRVLQRARRANSFLEEVKQGNLERECLEE
ACSLEEAREVFEDAEQTDEFWSKYKDGDQCEGHPCLNQGHCKDGIGDYTCTCAEGFEGKN
CEFSTREICSLDNGGCDQFCREERSEVRCSCAHGYVLGDDSKSCVSTERFPCGKFTQGRS
RRWAIHTSEDALDASELEHYDPADLSPTESSLDLLGLNRTEPSAGEDGSQVVRIVGGRDC
AEGECPWQALLVNEENEGFCGGTILNEFYVLTAAHCLHQAKRFTVRVGDRNTEQEEGNEM
AHEVEMTVKHSRFVKETYDFDIAVLRLKTPIRFRRNVAPACLPEKDWAEATLMTQKTGIV
SGFGRTHEKGRLSSTLKMLEVPYVDRSTCKLSSSFTITPNMFCAGYDTQPEDACQGDSGG
PHVTRFKDTYFVTGIVSWGEGCARKGKFGVYTKVSNFLKWIDKIMKARAGAAGSRGHSEA
PATWTVPPPLPL
>OSTC_HUMAN OSTEOCALCIN PRECURSOR
MRALTLLALLALAALCIAGQAGAKPSGAESSKGAAFVSKQEGSEVVKRPRRYLYQWLGAP
VPYPDPLEPRREVCELNPDCDELADHIGFQEAYRRFYGPV
>THRB_RAT PROTHROMBIN PRECURSOR (EC 3.4.21.5).
MLHVRGLGLPGCLALAALASLVHSQHVFLAPQQALSLLQRVRRANSGFLEELRKGNLERE
CVEEQCSYEEAFEALESPQDTDVFWAKYTVCDSVRKPRETFMDCLEGRCAMDLGLNYHGN
VSVTHTGIECQLWRSRYPHRPDINSTTHPGADLKENFCRNPDSSTSGPWCYTTDPTVRRE
ECSIPVCGQEGRTTVKMTPRSRGSKENLSPPLGECLLERGRLYQGNLAVTTLGSPCLAWD
SLPTKTLSKYQNFDPEVKLVQNFCRNPDRDEEGAWCFVAQQPGFEYCSLNYCDEAVGEEN
HDGDESIAGRTTDAEFHTFFDERTFGLGEADCGLRPLFEKKSLTDKTEKELLDSYIDGRI
VEGWDAEKGIAPWQVMLFRKSPQELLCGASLISDRWVLTAAHCILYPPWDKNFTENDLLV
RIGKHSRTRYERNVEKISMLEKIYIHPRYNWRENLDRDIALLKLKKPVPFSDYIHPVCLP
DKQTVTSLLQAGYKGRVTGWGNLRETWTTNINEIQPSVLQVVNLPIVERPVCKASTRIRI
TDNMFCAGFKVNDTKRGDACEGDSGGPFVMKSPYNHRWYQMGIVSWGEGCDRNGKYGFYT
HVFRLKRWMQKVIDQHR
```

```
>MGP_MOUSE MATRIX GLA-PROTEIN PRECURSOR (MGP).
MKSLLPLAILAALAVATLCYESHESMESYEISPFINRRNANTFMSPQQRWRAKAQKRVQE
RNKPAYEINREACDDYKLCERYAMVYGYNAAYNRYFRQRRGAKY
***                    (mandatory)
```

The MOTIF FORM, MAXIMUM LENGTH, MINIMUM LENGTH, OCCURRENCE NUMBER, and MUTATION must all be in upper case. They can appear in any order but they must precede the BEGIN command. The MOTIF FORM line allows one to specify the form of interesting motifs: 1 means *X* and 2 means *X*Y*. The MAXIMUM LENGTH line allows one to specify the maximum length of interesting motifs. When it is absent, the default value is 8 because that works well for proteins using the *X* pattern. The MINIMUM LENGTH line allows one to specify the minimum length of interesting motifs. When it is absent, the default value is 3. The OCCURRENCE NUMBER line allows one to specify the minimum occurrence number for interesting motifs. When it is absent, the default value is 1.

The MUTATION line allows one to specify the number of mutations allowed when matching a motif with a sequence. When it is absent, the default value is 0. The maximum number of mutations allowed in searching for the motifs is 10.

The sequences following the BEGIN constitute the set in which one wants to discover interesting motifs. The sequences must be in FASTA formats [50] (the line beginning with ">" is not recognized as sequence). The set can contain up to 50 sequences, where the maximum length of a sequence is 5,000.

9.2.2 Interpreting Results of DISCOVER

The results are organized into the following format:

Occurrence number	Motif
2	*MGIVSWGEGC*
2	*GNLERECLEE*
2	*GNLERECVEE*
2	*GIVSWGEGCA*

```
2              *GIVSWGEGCAR*
2              *GIVSWGEGCD*
2              *GIVSWGEGCDR*
2              *TGIVSWGEGC*
2              *RRANSGFLEE*
2              *DACQGDSGGP*
2              *DACEGDSGGP*
2              *IVSWGEGCAR*
2              *IVSWGEGCDR*
```

The results show the interesting motifs and their occurrence numbers with respect to the user specified parameter values.

DISCOVER is also a part of an X-window version tool: Visualization Tool for Pattern Matching and Discovery (VisualPMD) [16]. The screen layouts are shown in Figure 9.2 and 9.3.

DISCOVER is also implemented into an interactive tool which accepts parameter values one by one.

Example 9.1 (Interactive Version of DISCOVER)

```
% Enter the file name of sequences
  (an example file can be found in file SAMPLE;
  maximum number of sequences in the file is 50;
  maximum length of sequences is 5000) [SAMPLE]: SAMPLE

===> 3 sequences found in file <SAMPLE>

% Enter the form of interesting motifs 1 or 2
  1 means *X*; 2 means *X*Y*) [1] ? 1

% Enter the maximum length of interesting motifs
  (default is 8) [8] ? 8

% Enter the minimum length of interesting motifs
  (default is 3) [3] ? 3

% Enter the minimum occurrence number for interesting motifs
  (the occurrence number of an interesting motif refers to
  the number of sequences in which the motif approximately
```

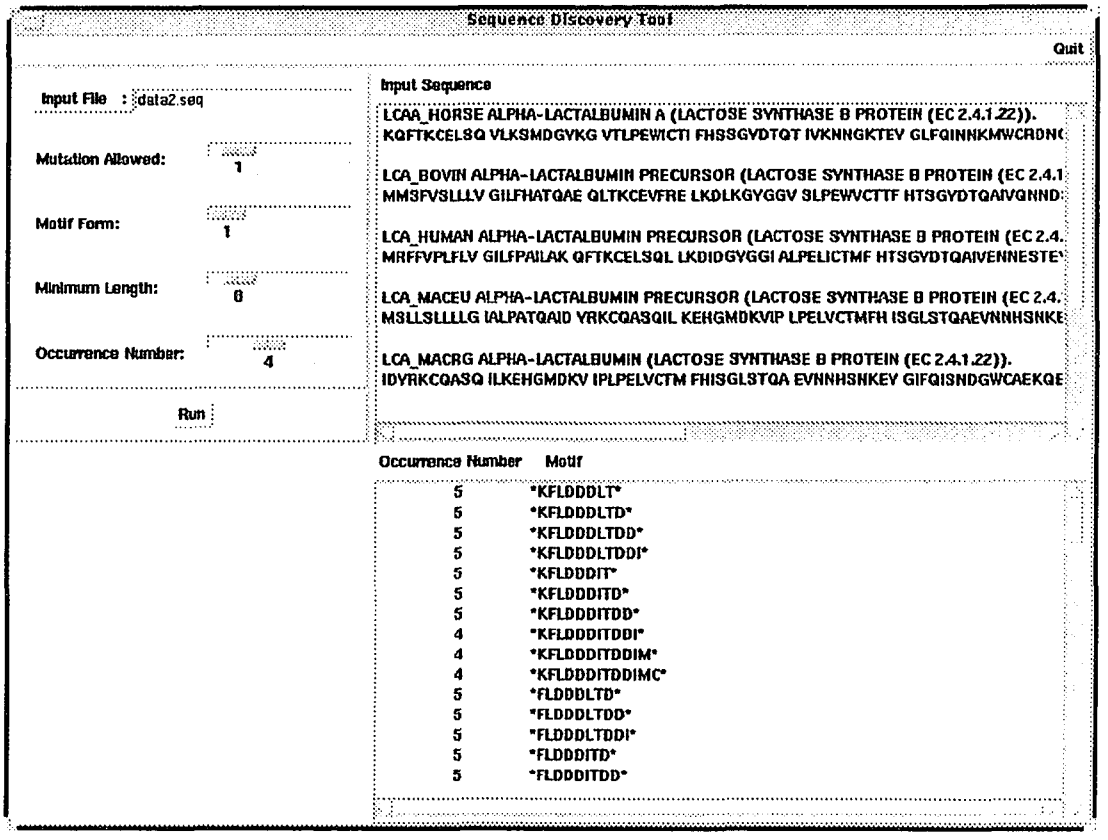

Figure 9.2 A screen layout of DISCOVER in VisualPMD, illustrating the discovery of patterns with the form *X* from a set of protein sequences. These protein sequences are obtained from SWISS-PROT [8].

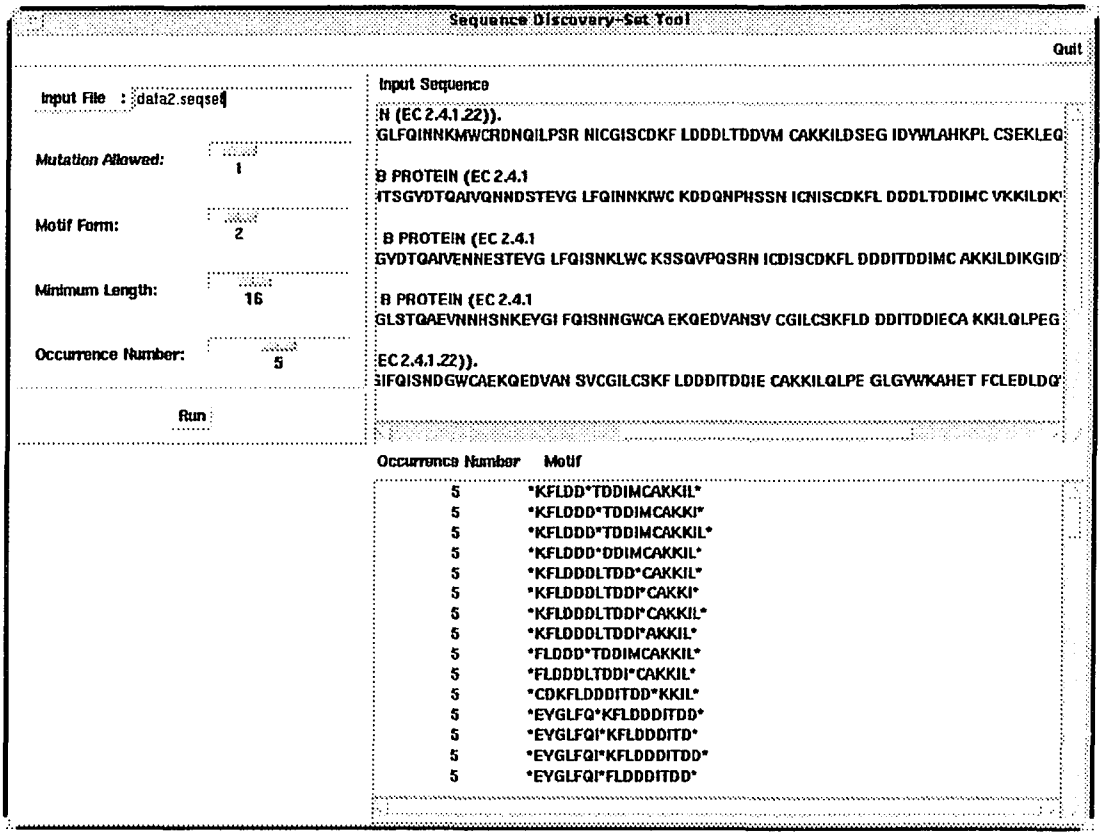

Figure 9.3 A screen layout of DISCOVER in VisualPMD, illustrating the discovery of patterns with the form *X*Y* from a set of protein sequences. These protein sequences are obtained from SWISS-PROT [8].

```
  occurs; default is 1) [1] ? 2

% Enter the number of mutations allowed in searching for
  similar motifs (default is 0; maximum number is 10) [1] ? 1

% Where the result should be stored
  (enter the file name) [data.out] ? data.out
```

□

9.3 Accessing the CLASSIFY Tool

9.3.1 Request Format for CLASSIFY

To classify a DNA sequence, send a message with the words "sequence classification" on the subject line and a body containing the keyword "#DNA" and one of the following two methods:

#motif-gluing DC-1 – Using the motif-gluing method to classify the DNA sequence;

#fingerprint DC-2 – Using the fingerprint method to classify the DNA sequence;

and then followed by one (and only one) test DNA sequence in FASTA formats.

To classify a protein sequence: send a message with the single word "sequence classification" on the subject line and a body containing the keyword "#PROTEIN" and one of the following four methods:

#motif-fingerprint PC-1 – Using the motif-fingerprint method to classify the protein sequence;

#bayesian PC-2 – Using the extended BLOCKS database produced by the Bayesian method to classify the protein sequence;

#bayesian-blosum PC-3. Using the extended BLOCKS database produced by the Bayesian and BLOSUM method to classify the protein sequence.

#dirichlet PC-4 – Using the extended BLOCKS database produced by the Dirichlet distribution method to classify the protein sequence;

and then followed by one (and only one) test protein sequence in FASTA formats.

Example 9.2 (Input format for CLASSIFY)

The input format for CLASSIFY is as follows:

```
To: discover@village.njit.edu
Subject: sequence classification
#PROTEIN
#motif-fingerprint
>CG2A_DAUCA G2/MITOTIC-SPECIFIC CYCLIN C13-1 (A-LIKE CYCLIN)
APSMTTPEPASKRRVVLGEISNNSSAVSGNEDLLCREFEVPKCVAQKKRKRGVKEDVGVD
FGEKFDDPQMCSAYVSDVYEYLKQMEMETKRRPMMNYIEQVQKDVTSNMRGVLVDWLVEV
SLEYKLLPETLYLAISYVDRYLSVNVLNRQKLQLLGVSSFLIASKYEEIKPKNVADFVDI
TDNTYSQQEVVKMEADLLKTLKFEMGSPTVKTFLGFIRAVQENPDVPKLKFEFLANYLAE
LSLLDYGCLEFVPSLIAASVTFLARFTIRPNVNPWSIALQKCSGYKSKDLKECVLLLHDL
QMGRRGGSLSAVRDKYKKHKFKCVSTLSPAPEIPESIFNDV
```

To facilitate visual inspection, user may group 10 letters in the query into a block, separated by a blank:

```
>CG2A_DAUCA G2/MITOTIC-SPECIFIC CYCLIN C13-1 (A-LIKE CYCLIN)
APSMTTPEPA SKRRVVLGEI SNNSSAVSGN EDLLCREFEV PKCVAQKKRK RGVKEDVGVD
FGEKFDDPQM CSAYVSDVYE YLKQMEMETK RRPMMNYIEQ VQKDVTSNMR GVLVDWLVEV
SLEYKLLPET LYLAISYVDR YLSVNVLNRQ KLQLLGVSSF LIASKYEEIK PKNVADFVDI
TDNTYSQQEV VKMEADLLKT LKFEMGSPTV KTFLGFIRAV QENPDVPKLK FEFLANYLAE
LSLLDYGCLE FVPSLIAASV TFLARFTIRP NVNPWSIALQ KCSGYKSKDL KECVLLLHDL
QMGRRGGSLS AVRDKYKKHK FKCVSTLSPA PEIPESIFND V
```

□

9.3.2 Interpreting Results of CLASSIFY

For DNA classification, CLASSIFY answers "Alu", "non-Alu" or "no opinion" as well as the score of the query sequence. The following example shows the output returned from the motif-gluing method of DC-1.

Example 9.3 (Return file from DC-1)

```
>gb|L05920_HSAL001628 (Alu-J)
GGCTGGCACAGTGGCTCATGCCTGTAATCCTTTGGGAGGTTGAGGTAGGAAGATTGCTTG
AGGCCAGGAGTTCGAGACTAGCCTGGGCCGAACATACATACCAAGACCTTGTCTCTACCA
AAAAAAAAAAAAAAAAAAATTAGCTGAGTGTGATGGTGCACGTTTGTAGTCCTTAATACT
CAGGAGGCTGAGGTGGGAGGATCACTTGAGCCCAGAAGGTCGAGGCTGCAGTAAGCCTTA
ATCATATCATTGCACTTTAGCCTGGGCAATAGGAAAAAAAAAAGAAAGAAAGAAAGGGAG
GAAAGAAAGAAAGGAAGGAGAAAGAAA

Search Method: Motif-gluing method

Score: 23

Classified as: Alu sequence.

PS. In the Motif-gluing method, if the score is less than 14, the
    sequence is classified as non-Alu. If the score is greater than
    14, the sequence is classified as Alu. Otherwise, the answer is
    "no opinion".
```

□

For protein classification, CLASSIFY compares the query with the characteristic motifs and fingerprints for each PROSITE group (if the motif-fingerprint method PC-1 is used) or compares the query with an extended BLOCKS database. The system displays a group G to which the query should belong, together with the group's documentation listed in the PROSITE catalog.

In addition, one of the following three outputs is displayed:

- If the query protein is classified in the first phase of the motif-fingerprint method, then G's score is shown and the positions of G's characteristic motifs occurring in the query are highlighted.
- If the query protein is classified in the second phase of the motif-fingerprint method, then the training sequence in G with the highest vote is displayed.
- If the query protein is classified by using the extended BLOCKS databases, then G's score is shown.

Here is an example output returned from phase 1 of the motif-fingerprint method PC-1.

Example 9.4 (Return file from phase 1 of PC-1)

```
>FA10_BOVIN COAGULATION FACTOR X PRECURSOR (EC 3.4.21.6)
MAGLLHLVLL STALGGLLRP AGSVFLPRDQ AHRVLQRARR ANSFLEEVKQ GNLERECLEE
ACSLEEAREV FEDAEQTDEF WSKYKDGDQC EGHPCLNQGH CKDGIGDYTC TCAEGFEGKN
CEFSTREICS LDNGGCDQFC REERSEVRCS CAHGYVLGDD SKSCVSTERF PCGKFTQGRS
RRWAIHTSED ALDASELEHY DPADLSPTES SLDLLGLNRT EPSAGEDGSQ VVRIVGGRDC
AEGECPWQAL LVNEENEGFC GGTILNEFYV LTAAHCLHQA KRFTVRVGDR NTEQEEGNEM
AHEVEMTVKH SRFVKETYDF DIAVLRLKTP IRFRRNVAPA CLPEKDWAEA TLMTQKTGIV
SGFGRTHEKG RLSSTLKMLE VPYVDRSTCK LSSSFTITPN MFCAGYDTQP EDACQGDSGG
PHVTRFKDTY FVTGIVSWGE GCARKGKFGV YTKVSNFLKW IDKIMKARAG AAGSRGHSEA
PATWTVPPPL PL

Search Method: Motif-fingerprint method

Groups Searched=768

Group Hit=PS00011, GLU_CARBOXYLATION;
          Vitamin K-dependent carboxylation domain.

Group Score=72

The query with the group's characteristic motifs occurring it (the
motifs are highlighted):

MAGLLHLVLLSTALGGLLRPAGSVFLPRDQAHRVLQRARRANSFLEEVKQGNLERECLEE
                                          ^^^^^^^^^   ^^^^^^^

ACSLEEAREVFEDAEQTDEFWSKYKDGDQCEGHPCLNQGHCKDGIGDYTCTCAEGFEGKN
    ^^^^^^^^             ^^^^^

CEFSTREICSLDNGGCDQFCREERSEVRCSCAHGYVLGDDSKSCVSTERFPCGKFTQGRS
            ^^^^

RRWAIHTSEDALDASELEHYDPADLSPTESSLDLLGLNRTEPSAGEDGSQVVRIVGGRDC

AEGECPWQALLVNEENEGFCGGTILNEFYVLTAAHCLHQAKRFTVRVGDRNTEQEEGNEM
                   ^^^^       ^^^^^^^

AHEVEMTVKHSRFVKETYDFDIAVLRLKTPIRFRRNVAPACLPEKDWAEATLMTQKTGIV
                                                            ^

SGFGRTHEKGRLSSTLKMLEVPYVDRSTCKLSSSFTITPNMFCAGYDTQPEDACQGDSGG
^                                       ^^^^^^^       ^^^^^^
```

```
PHVTRFKDTYFVTGIVSWGEGCARKGKFGVYTKVSNFLKWIDKIMKARAGAAGSRGHSEA
PATWTVPPPLPL

**************************************************************
ID GLU_CARBOXYLATION; PATTERN.
AC PS00011;
DT APR-1990(CREATED); APR-1990(DATA UPDATE); OCT-1993(INFO UPDATE).
DE Vitamin K-dependent carboxylation domain.
PA x(12)-E-x(3)-E-x-C-x(6)-[DEN]-x-[LIVMFY]-x(9)-[FYW].
NR /RELEASE=26,33329;
NR /TOTAL=36(36); /POSITIVE=33(33); /UNKNOWN=0(0); /FALSE_POS=3(3);
NR /FALSE_NEG=0(0);
CC /TAXO-RANGE=??E??; /MAX-REPEAT=1;
CC /SITE=2,carboxylation; /SITE=4,carboxylation;
DR P02820,OSTC_BOVIN,T;P02822,OSTC_CHICK,T;P15504,OSTC_DRONO,T;
DR P02821,OSTC_FELCA,T;P02818,OSTC_HUMAN,T;P02819,OSTC_MACFA,T;
DR P04641,OSTC_MOUSE,T;P04640,OSTC_RAT   ,T;P02823,OSTC_XIPGL,T;
DR P07507,MGP_BOVIN ,T;P08493,MGP_HUMAN ,T;P19788,MGP_MOUSE ,T;
DR P08494,MGP_RAT   ,T;P07224,PRTS_BOVIN,T;P07225,PRTS_HUMAN,T;
DR P00744,PRTZ_BOVIN,T;P22891,PRTZ_HUMAN,T;P00743,FA10_BOVIN,T;
DR P00742,FA10_HUMAN,T;P25155,FA10_CHICK,T;P22457,FA7_BOVIN ,T;
DR P08709,FA7_HUMAN ,T;P00741,FA9_BOVIN ,T;P19540,FA9_CANFA ,T;
DR P00740,FA9_HUMAN ,T;P16294,FA9_MOUSE ,T;P00745,PRTC_BOVIN,T;
DR P04070,PRTC_HUMAN,T;P31394,PRTC_RAT   ,T;P00735,THRB_BOVIN,T;
DR P00734,THRB_HUMAN,T;P19221,THRB_MOUSE,T;P18292,THRB_RAT   ,T;
DR P16295,FA9_CAVPO ,P;P16293,FA9_PIG   ,P;P16292,FA9_RABIT ,P;
DR P16296,FA9_RAT   ,P;P16291,FA9_SHEEP ,P;P28317,OSTC_LEPMA,P;
DR P23799,ESA8_TRYBB,F;P26337,ESA8_TRYEQ,F;P33114,SPAB_BACSU,F;
DO PDOC00011;
{PDOC00011}
{PS00011; GLU_CARBOXYLATION}
{BEGIN}
*****************************************
* Vitamin K-dependent carboxylation domain *
*****************************************

Vitamin K-dependent carboxylation [1,2] is the post-translational
modification of glutamic residues to form gamma-carboxyglutamate
(Gla). The proteins known to contain Gla are listed below.

 - A number of plasma proteins involved in blood coagulation. These
   proteins are: prothrombin, coagulation factors VII, IX and X,
   proteins C, S, and Z.
```

- Two proteins that occur in calcified tissues: osteocalcin (also known as bone-Gla protein, BGP), and matrix Gla-protein (MGP).
- Cone snails venom peptides: conantokin-G and -T, and conotoxin GS [3].

With the exception of the snail toxins all these proteins contain a N-terminal module of about forty amino acids where the majority of the Glu residues are carboxylated. This domain is responsible for the high-affinity binding of calcium ions. The Gla-domain starts at the N-terminal extremity of the mature form of these proteins and ends with a conserved aromatic residue; a conserved Gla-x(3)-Gla-x-Cys motif [4] is found in the middle of the domain which seems to be important for the substrate recognition by the carboxylase.

-Consensus pattern: x(12)-E-x(3)-E-x-C-x(6)-[DEN]-x-[LIVMFY]-x(9)-[FYW]
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: Trypanosoma ESAG8 protein and Bacillus subtilis spaB.
-Note: all the glutamic residues that are present in the domain are potential carboxylation sites; in coagulation proteins all of them are modified to Gla, while in BGP and MGP some are not modified.
-Expert(s) to contact by email: Price P.A.
pprice@ucsd.edu
-Last update: December 1992 / Text revised.

[1] Friedman P.A., Przysiecki C.T.
Int. J. Biochem. 19:1-7(1987).
[2] Vermeer C.
Biochem. J. 266:625-636(1990).
[3] Haack J.A., Rivier J.E., Parks T.N., Mena E.E., Cruz L.J., Olivera B.M.
J. Biol. Chem. 265:6025-6029(1990).
[4] Price P.A., Fraser J.D., Metz-Virca G.
Proc. Natl. Acad. Sci. U.S.A. 84:8335-8339(1987).
{END}

□

Here is an example output returned from phase 2 of the motif-fingerprint method PC-1.

Example 9.5 (Return file from phase 2 of PC-1)

```
>CANR_HUMAN CANNABINOID RECEPTOR.
MKSILDGLAD TTFRTITTDL LYVGSNDIQY EDIKGDMASK LGYFPQKFPL TSFRGSPFQE
KMTAGDNPQL VPADQVNITE FYNKSLSSFK ENEENIQCGE NFMDIECFMV LNPSQQLAIA
VLSLTLGTFT VLENLLVLCV ILHSRSLRCR PSYHFIGSLA VADLLGSVIF VYSFIDFHVF
HRKDSRNVFL FKLGGVTASF TASVGSLFLT AIDRYISIHR PLAYKRIVTR PKAVVAFCLM
WTIAIVIAVL PLLGWNCEKL QSVCSDIFPH IDETYLMFWI GVTSVLLLFI VYAYMYILWK
AHSHAVRMIQ RGTQKSIIIH TSEDGKVQVT RPDQARMDIR LAKTLVLILV VLIICWGPLL
AIMVYDVFGK MNKLIKTVFA FCSMLCLLNS TVNPIIYALR SKDLRHAFRS MFPSCEGTAQ
PLDNSMGDSD CLHKHANNAA SVHRAAESCI KSTVKIAKVT MSVSTDTSAE AL

Search Method: Motif-fingerprint method

Groups Searched=768

Group Hit=PS00237, G_PROTEIN_RECEPTOR;
          G-protein coupled receptors signature.

Highest Vote Training Sequence=>

EDG1_HUMAN PROBABLE G PROTEIN-COUPLED RECEPTOR EDG-1.

Sequence Vote=141

MGPTSVPLVK AHRSSVSDYV NYDIIVRHYN YTGKLNISAD KENSIKLTSV VFILICCFII
LENIFVLLTI WKTKKFHRPM YYFIGNLALS DLLAGVAYTA NLLLSGATTY KLTPAQWFLR
EGSMFVALSA SVFSLLAIAI ERYITMLKMK LHNGSNNFRL FLLISACWVI SLILGGLPIM
GWNCISALSS CSTVLPLYHK HYILFCTTVF TLLLLSIVIL YCRIYSLVRT RSRRLTFRKN
ISKASRSSEN VALLKTVIIV LSVFIACWAP LFILLLLDVG CKVKTCDILF RAEYFLVLAV
LNSGTNPIIY TLTNKEMRRA FIRIMSCCKC PSGDSAGKFK RPIIAGMEFS RSKSDNSSHP
QKDEGDNPET IMSSGNVNSS S

**********************************************************************
ID G_PROTEIN_RECEPTOR; PATTERN.
AC PS00237;
DT APR-1990(CREATED);OCT-1993(DATA UPDATE);OCT-1993(INFO UPDATE).
```

```
DE G-protein coupled receptors signature.
PA [GSTALIVMYWC]-[GSTAPDNE]-{EDPKRH}-x(2)-[LIVMNGA]-x(2)-
PA [LIVMFT]-[GSTANC]-[LIVMFYWAST]-[DEN]-R-[FYWCSH]-x(2)-[LIVM].
NR /RELEASE=26,33329;
NR /TOTAL=299(299);/POSITIVE=283(283);/UNKNOWN=0(0);
NR /FALSE_POS=16(16);/FALSE_NEG=5(5);
CC /TAXO-RANGE=??E?V; /MAX-REPEAT=1;
DR P08908,5H1A_HUMAN,T;P19327,5H1A_RAT  ,T;P28222,5H1B_HUMAN,T;
DR P28334,5H1B_MOUSE,T;P28564,5H1B_RAT  ,T;P11614,5H1D_CANFA,T;
DR P28221,5H1D_HUMAN,T;P28565,5H1D_RAT  ,T;P28566,5H1E_HUMAN,T;
DR P30939,5H1F_HUMAN,T;Q02284,5H1F_MOUSE,T;P30940,5H1F_RAT  ,T;
DR P18599,5H2A_CRIGR,T;P28223,5H2A_HUMAN,T;P14842,5H2A_RAT  ,T;
DR Q02152,5H2B_MOUSE,T;P30994,5H2B_RAT  ,T;P28335,5H2C_HUMAN,T;
DR P08909,5H2C_RAT  ,T;P30966,5H5A_MOUSE,T;P31387,5H5B_MOUSE,T;
DR P31388,5H6_RAT   ,T;P32304,5H7_MOUSE ,T;P32305,5H7_RAT   ,T;
DR P20905,5HT1_DROME,T;P28285,5HTA_DROME,T;P28286,5HTB_DROME,T;
DO PDOC00210;
{PDOC00210}
{PS00237; G_PROTEIN_RECEPTOR}
{BEGIN}
*****************************************
* G-protein coupled receptors signature *
*****************************************

G-protein coupled receptors [1 to 4] (also called R7G) are an
extensive group of hormones, neurotransmitters, odorants and light
receptors which transduce extracellular signals by interaction with
guanine nucleotide-binding (G) proteins. The receptors that are
currently known to belong to this family are listed below.

 - 5-hydroxytryptamine (serotonin) 1A to 1F, 2A to 2C, 5A, 5B and 6
   [5].
 - Acetylcholine, muscarinic-type, M1 to M5.
 - Adenosine A1, A2A, A2B and A3 [6].
 - Adrenergic alpha-1A to -1C; alpha-2A to -2D;  beta-1 to -3 [7].
 - Angiotensin II type I.
 - B2 bradykinin.
 - C5a anaphylatoxin.
 - Cannabinoid.
 - Cholecystokinin-A.
 - Cholecystokinin-B/Gastrin.
 - Dopamine D1 to D5 [8].
 - Endothelin ET-a and ET-b [9].
 - fMet-Leu-Phe (fMLP) (N-formyl peptide).
```

- Follicle stimulating hormone (FSH-R) [10].
- Gastrin-releasing peptide (GRP-R).
- Gonadotropin-releasing hormone (GNRH-R).
- Histamine H1 and H2 (gastric receptor I).
- Interleukin-8 (IL-8R).
- Lutropin-choriogonadotropic hormone (LSH-R) [10].
- Melanocyte stimulating hormone (MSH-R).
- Neuromedin B (NMB-R).
- Neuromedin K (NK-3R).
- Neuropeptide Y types 1 and 2.
- Neurotensin (NT-R).
- Octopamine (tyramine), from insects.
- Odorants [11].
- Opioids.
- Oxytocin (OT-R).
- Platelet activating factor (PAF-R).
- Prostaglandin E.
- Somatostatin types 1 to 5.
- Substance-K (NK-2R).
- Substance-P (NK-1R).
- Thrombin.
- Thromboxane A2.
- Thyrotropin (TSH-R) [10].
- Thyrotropin releasing factor (TRH-R).
- Vasopressin V1a and V2.
- Visual pigments (opsins and rhodopsin) [12].
- Proto-oncogene mas.
- A number of orphan receptors (whose ligand is not known).
- Three putative receptors encoded in the genome of cytomegalovirus: US27, US28, and UL33.
- ECRF3, a putative receptor encoded in the genome of herpesvirus saimiri.
- Slime mold cyclic AMP receptors.

The structure of all these receptors is thought to be identical. They have seven hydrophobic regions, each of which most probably spans the membrane. The N-terminus is located on the extracellular side of the membrane and is often glycosylated, while the C-terminus is cytoplasmic and generally phosphorylated. Three extracellular loops alternate with three intracellular loops to link the seven transmembrane regions. Most, but not all of these receptors, lack a signal peptide. The most conserved parts of these proteins are the transmembrane regions and the first two cytoplasmic loops. A conserved acidic-Arg-aromatic triplet is present in the N-terminal

extremity of the second cytoplasmic loop [13] and could be implicated in the interaction with G proteins.

To detect this widespread family of proteins we have developed a pattern that contains the conserved triplet and that also spans the major part of the third transmembrane helix.

-Consensus pattern: [GSTALIVMYWC]-[GSTAPDNE]-{EDPKRH}-x(2)-[LIVMNGA]
-x(2)-[LIVMFT]-[GSTANC]-[LIVMFYWAST]-[DEN]-R-
[FYWCSH]-x(2)-[LIVM]
-Sequences known to belong to this class detected by the pattern:
ALL, except for two Drosophila opsins, ECRF3, Xenopus Endothelin-3 receptor and for the slime mold cAMP receptors which do not really seem to belong to this R7G family.
-Other sequence(s) detected in SWISS-PROT: 16 other proteins.
-Expert(s) to contact by email: Chollet A.
arc3029@ggr.co.uk
Attwood T.K.
bph6tka@biovax.leeds.ac.uk
Kolakowski L.F. Jr.
kolakowski@helix.mgh.harvard.edu
-Last update: October 1993 / Pattern and text revised.

[1] Strosberg A.D.
Eur. J. Biochem. 196:1-10(1991).
[2] Kerlavage A.R.
Curr. Opin. Struct. Biol. 1:394-401(1991).
[3] Probst W.C., Snyder L.A., Schuster D.I., Brosius J., Sealfon S.C.
DNA Cell Biol. 11:1-20(1992).
[4] Savarese T.M., Fraser C.M.
Biochem. J. 283:1-9(1992).
[5] Branchek T.
Curr. Biol. 3:315-317(1993).
[6] Stiles G.L.
J. Biol. Chem. 267:6451-6454(1992).
[7] Friell T., Kobilka B.K., Lefkowitz R.J., Caron M.G.
Trends Neurosci. 11:321-324(1988).
[8] Stevens C.F.
Curr. Biol. 1:20-22(1991).
[9] Sakurai T., Yanagisawa M., Masaki T.
Trends Pharmacol. Sci. 13:103-107(1992).
[10] Salesse R., Remy J.J., Levin J.M., Jallal B., Garnier J.
Biochimie 73:109-120(1991).
[11] Lancet D., Ben-Arie N.

```
     Curr. Biol. 3:668-674(1993).
[12] Applebury M.L., Hargrave P.A.
     Vision Res. 26:1881-1895(1986).
[13] Attwood T.K., Eliopoulos E.E., Findlay J.B.C.
     Gene 98:153-159(1991).
{END}
```

□

The following example shows an output returned by using the extended BLOCKS database produced by the Bayesian method (PC-2).

Example 9.6 (Return file from PC-2)

```
>CG2A_DAUCA G2/MITOTIC-SPECIFIC CYCLIN C13-1 (A-LIKE CYCLIN)
APSMTTPEPA SKRRVVLGEI SNNSSAVSGN EDLLCREFEV PKCVAQKKRK RGVKEDVGVD
FGEKFDDPQM CSAYVSDVYE YLKQMEMETK RRPMMNYIEQ VQKDVTSNMR GVLVDWLVEV
SLEYKLLPET LYLAISYVDR YLSVNVLNRQ KLQLLGVSSF LIASKYEEIK PKNVADFVDI
TDNTYSQQEV VKMEADLLKT LKFEMGSPTV KTFLGFIRAV QENPDVPKLK FEFLANYLAE
LSLLDYGCLE FVPSLIAASV TFLARFTIRP NVNPWSIALQ KCSGYKSKDL KECVLLLHDL
QMGRRGGSLS AVRDKYKKHK FKCVSTLSPA PEIPESIFND V

Search Method: Extended BLOCKS database using the Bayesian method

Groups Searched: 768

Group Hit: PS00292

Score of Hit Group: 92.00

************************************************************
ID CYCLINS; PATTERN.
AC PS00292;
DT APR-1990(CREATED);JUN-1992(DATA UPDATE);OCT-1993(INFO UPDATE).
DE Cyclins signature.
PA R-x(2)-[LIVM]-x(2)-[FYW]-[LIVM]-x(8)-[LIVMC]-x(4)-[LIVMFY]
PA -x(2)-[STAGC]-[LIVMFYQ]-x-[LIVMFY](2)-D-[RK]-[LIVMFYW].
NR /RELEASE=26,33329;
NR /TOTAL=46(46);/POSITIVE=46(46);/UNKNOWN=0(0);/FALSE_POS=0(0);
NR /FALSE_NEG=5(5);
CC /TAXO-RANGE=??E?V; /MAX-REPEAT=1;
```

```
DR P30274,CG2A_BOVIN,T;P14785,CG2A_DROME,T;P20248,CG2A_HUMAN,T;
DR P24861,CG2A_PATVU,T;P04962,CG2A_SPISO,T;P18606,CG2A_XENLA,T;
DR P07818,CG2B_ARBPU,T;P18063,CG2B_ASTPE,T;P29332,CGB2_CHICK,T;
DR P20439,CG2B_DROME,T;P14635,CGB1_HUMAN,T;P15206,CG2B_MARGL,T;
DR P24860,CGB1_MOUSE,T;P30276,CGB2_MOUSE,T;P30277,CGB1_RAT  ,T;
DR P24862,CG2B_PATVU,T;P13952,CG2B_SPISO,T;P13350,CGB1_XENLA,T;
DR P13351,CGB2_XENLA,T;P25010,CG2A_DAUCA,T;P30183,CG2B_ARATH,T;
DR P30278,CG2B_MEDSA,T;P25011,CG21_SOYBN,T;P25012,CG22_SOYBN,T;
DR P10815,CG21_SCHPO,T;P24865,CG22_SCHPO,T;P24868,CG21_YEAST,T;
DR P24869,CG22_YEAST,T;P24870,CG23_YEAST,T;P24871,CG24_YEAST,T;
DR P30283,CGS5_YEAST,T;P32943,CGS6_YEAST,T;P30284,CG21_EMENI,T;
DR P20437,CG11_YEAST,T;P20438,CG12_YEAST,T;P13365,CG13_YEAST,T;
DR P24866,CG11_CANAL,T;P24385,CGD1_HUMAN,T;P25322,CGD1_MOUSE,T;
DR P30279,CGD2_HUMAN,T;P30280,CGD2_MOUSE,T;Q04827,CGD2_RAT  ,T;
DR P30281,CGD3_HUMAN,T;P30282,CGD3_MOUSE,T;P24864,CG1E_HUMAN,T;
DR Q01043,CGH2_HSVSA,T;
DR P25009,CG1P_SCHPO,N;P25008,CG1C_DROME,N;P24863,CG1C_HUMAN,N;
DR P24867,CG16_YEAST,N;P25693,CG17_YEAST,N;
DR P30286,CG1_MEDSA ,P;
DO PDOC00264;
{PDOC00264}
{PS00292; CYCLINS}
{BEGIN}
**********************
* Cyclins signature *
**********************
```

Cyclins [1,2,3] are eukaryotic proteins which play an active role in controlling nuclear cell division cycles. Cyclins, together with the p34 (cdc2) or cdk2 kinases, form the Maturation Promoting Factor (MPF). There are two main groups of cyclins:

- G2/M cyclins, essential for the control of the cell cycle at the G2/M (mitosis) transition. G2/M cyclins accumulate steadily during G2 and are abruptly destroyed as cells exit from mitosis (at the end of the M-phase).
- G1/S cyclins, essential for the control of the cell cycle at the G1/S (start) transition.

In most species, there are multiple forms of G1 and G2 cyclins. For example, in vertebrates, there are two G2 cyclins, A and B, and at least three G1 cyclins, C, D, and E.

A cyclin homolog has also been found in herpesvirus saimiri [4].

The best conserved region is in the central part of the cyclins' sequences, known as the 'cyclin-box', from which we have derived a 32 residue pattern.

-Consensus pattern: R-x(2)-[LIVM]-x(2)-[FYW]-[LIVM]-x(8)-[LIVMC]-x(4)-[LIVMFY]-x(2)-[STAGC]-[LIVMFYQ]-x-[LIVMFY](2)-D-[RK]-[LIVMFYW]
-Sequences known to belong to this class detected by the pattern: ALL, except for G1/S cyclins C from Human and Drosophila, puc1 from fission yeast and HCS26 from yeast.
-Other sequence(s) detected in SWISS-PROT: NONE.
-Last update: October 1993 / Text revised.

[1] Nurse P.
Nature 344:503-508(1990).
[2] Norbury C., Nurse P.
Curr. Biol. 1:23-24(1991).
[3] Lew D.J., Reed S.I.
Trends Cell Biol. 2:77-81(1992).
[4] Nicholas J., Cameron K.R., Honess R.W.
Nature 355:362-365(1992).
{END}

□

CHAPTER 10

CONCLUSIONS AND FUTURE WORKS

10.1 Summary of the Dissertation

Combinatorial pattern discovery is useful for discovering internal structural properties that result in the common physical manifestations of a group of physical objects. In thesis, we have presented two examples of pattern discovery: (1) discovery of active patterns (also known as motifs) from a database of DNA and protein sequences; and (2) discovery of conserved blocks for a family of proteins. In the first example, the general strategy we proposed here is first to find patterns satisfying structural constrains (of length and form) in a small sample, and then to evaluate these on the whole database. To improve the efficiency, we developed two optimization heuristics:

- Evaluate only those patterns that pass a statistical test in the sample;
- Eliminate patterns if certain combinations of simpler ones have already been evaluated and have been shown to be irrelevant.

We applied the proposed techniques to discover active patterns in generated data and functionally related protein sequences. Our experimental results indicated that the discovery algorithms are sensitive to the data, sample size and the distance allowed in matching a pattern with a sequence. When looking for exact matches (distance of 0), small samples work well. On the other hand for inexact matches (distance of 1), the sample needs to be large. The reason is that in some data, there are pattern which exactly appear in very few sequences, but which approximately occur, within distance 1, in many sequences. Unless the sample is so large that it contains at least one of those very few sequences, our algorithms cannot find the active patterns.

In the second example, we start with existing blocks in the BLOCKS database developed by Henikoff and Henikoff of the Howard Hughes Medical Institute, Seattle, Washington [30, 33] and extend the blocks using a modified version of the iterative algorithm developed by Tatusov, Altschul and Koonin of the National Institutes of Health, Bethesda, Maryland [81, 82].

We then applied our pattern discovery algorithms to build classifiers. When we applied these classifiers to DNA and protein sequences, we found that our classifiers are slightly better than the best classifier available today and provide complementary information to them, thus indicating the potential of the proposed methods.

10.2 Future Works

The work described here is part of a project for pattern matching and discovery in scientific, program and document databases [73, 74, 89, 90, 91, 95]. Our future works will focus on:

- Application of our pattern discovery techniques to trees and graphs and using the discovered patterns to do classification of RNA secondary structures (represented as trees) [15, 49, 53, 55];
- Development of the discovering algorithms for high dimensional data structures such as free trees and graphs, which are commonly used to represent chemical compounds [3, 4];
- Development of new index structures for supporting pattern matching and discovery queries in scientific databases.

REFERENCES

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403-410, 1990.

2. S. F. Altschul and D. J. Lipman, "Protein database searches for multiple alignments," *Proc. Nat. Acad. Sci. USA.*, vol. 87, pp. 5509-5513, 1990.

3. J. E. Ash, P. A. Chubb, S. E. Ward, S. M. Welford, and P. Willett, *Communication, Storage and Retrieval of Chemical Information*, Ellis Horwood, Chichester, England, 1985.

4. J. E. Ash and E. Hyde, *Chemical Information Systems*, Ellis Horwood, Chichester, England, 1975.

5. D. J. Bacon and W. J. Anderson, "Multiple sequence alignment," *Journal of Molecular Biology*, vol. 191, pp. 153-161, 1986.

6. A. Bairoch, "Prosite: a dictionary of protein sites and patterns," *EMBL*, vol. release 4, 1989.

7. A. Bairoch, "PROSITE: A dictionary of sites and patterns in proteins," *Nucleic Acids Research*, vol. 20, pp. 2013-2018, 1992.

8. A. Bairoch and B. Boeckmann, "The SWISS-PROT protein sequence data bank," *Nucleic Acids Research*, vol. 20, pp. 2019-2022, 1992.

9. O. G. Berg, "Selection of dna binding sites by regulatory protein: The lexa protein and the arginie repressor use different strategies for functional specificity," *Nucleic Acids Research*, vol. 16, pp. 5089-5105, 1988.

10. O. G. Berg and P. H. von Hippel, "Selection of dna binding sites by regulatory proteins," *Journal of Molecular Biology*, vol. 193, pp. 723-750, 1987.

11. R. S. Boyer and J. S. Moore, "A fast string matching algorithm," *Communications of the ACM*, vol. 20, pp. 262-272, 1977.

12. R. J. Britten, W. F. Baron, D. Stout, and E. H. Davidson, "Sources and evolution of human Alu repeated sequences," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 4770-4774, 1988.

13. M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjolander, and D. Haussler, "Using dirichlet mixture priors to derive hidden markov models for protein families," in *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, pp. 47-55, AAAI, 1993.

14. A. Califano and I. Rigoutsos, "Flash: A fast look-up algorithm for string homology," in *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, Bethesda, MD, July 1993.

15. T. R. Cech, "Conserved sequences and structures of group I introns: building an active site for RNA catalysis – a review," *Gene*, vol. 73, pp. 259–271, 1988.

16. G. J. S. Chang, J. T. L. Wang, G. W. Chirn, and C. Y. Chang, "A visualization tool for pattern matching and discovery in scientific databases," in *Proceedings of the Eighth International Conference on Software Engineering and Knowledge Engineering*, Lake Tahoe, NV, June 1996.

17. B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo, "Sequence landscapes," *Nucleic Acid Research*, vol. 14, pp. 141–158, 1986.

18. A. L. Cobb, "Fast identification of approximately matching substrings," in *Combinatorial Pattern Matching, Lecture Notes in Computer Science*, pp. 64–74, 1994.

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

20. H. Curtis, *Biology*, Worth Publishers, New York, 1975.

21. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, "Atlas of protein sequence and structure," in *Natl. Biomed. Res. Found., Washington, DC, Vol. 5, Suppl. 3*, pp. 345–352, 1978.

22. I. B. Dodd and J. B. Egan, "Improved detection of helix-turn-helix dna-binding motifs in protein sequences," *Nucleic Acids Research*, vol. 18, no. 17, pp. 5019–5026, 1990.

23. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, NY, 1973.

24. D. Galas, W. Eggert, and M. Waterman, "Rigorous pattern-recognition methods for dna sequences. analysis of promoter sequences from escherichia coli," *Journal of Molecular Biology*, vol. 186, pp. 117–128, 1985.

25. M. S. Gelfand, "Prediction of function in dna sequence," *Journal of Computational Biology*, vol. 2, pp. 87–115, 1995.

26. W. Gish and D. J. States, "Identification of protein coding regions by database similarity search," *Nature Genetics*, vol. 3, pp. 266–272, 1993.

27. M. Gribskov, A. D. McLachlan, and D. Eisenberg in *Proc. Natl. Acad. Sci, USA*, pp. 4355–4358, 1987.

28. P. J. Haas and A. N. Swami, "Sequential sampling procedures for query size estimation," in *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, San Diego, CA, pp. 341–350, June 1992.

29. S. K. Hanks, A. M. Quinn, and T. Hunter, "The protein kinase family: Conserved features and deduced phylogeny of the catalytic domains," *Science*, vol. 241, pp. 42–52, 1988.

30. S. Henikoff and J. G. Henikoff, "Automated assembly of protein blocks for database searching," *Nucleic Acids Research*, vol. 19, no. 23, pp. 6565–6572, 1991.

31. S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Natl. Acad. Sci. USA*, vol. 89, pp. 10915–10919, 1992.

32. S. Henikoff and J. G. Henikoff, "Performance evaluation of amino acid substitution matrices," *Proteins: Structure, Function and Genetics*, vol. 17, pp. 49–61, 1993.

33. S. Henikoff and J. G. Henikoff, "Protein family classification based on searching a database of blocks," *Genomics*, vol. 19, pp. 97–107, 1994.

34. S. Henikoff, J. C. Wallace, and J. P. Brown, "Finding protein similarities with nucleotide sequence databases," *Methods in Enzymology*, vol. 183, pp. 111–132, 1990.

35. H. Hirsh and M. Noordewier, "Using background knowledge to improve inductive learning of dna sequences," in *Proceedings IEEE Conference on AI for Application*, 1994.

36. J. D. Hirst and M. J. E. Sternberg, "Prediction of structural and functional features of protein and nucleic acid sequences by artificial neural networks," *Biochemistry*, vol. 31, pp. 7211–7219, 1992.

37. T. C. Hodgman, "The elucidation of protein function by sequence motif analysis," *Computer Applications in Biosciences*, vol. 5, pp. 1–13, 1989.

38. W. C. Hou and G. Ozsoyoglu, "Statistical estimators for aggregate relational algebra queries," *ACM Transactions on Database Systems*, vol. 16, no. 4, pp. 600–654, December 1991.

39. L. C. K. Hui, "Color set size problem with applications to string matching," in *Combinatorial Pattern Matching, Lecture Notes in Computer Science, 644* (A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, eds.), Springer-Verlag, New York, NY, pp. 230–243, 1992.

40. D. T. Jones, W. R. Taylor, and J. M. Thornton, "The rapid generation of mutation data matrices from protein sequences," *Computer Applications in Biosciences*, vol. 8, pp. 275–282, 1992.

41. J. Jurka, D. J. Kaplan, C. H. Duncan, J. Walichiewicz, A. Milosavljevic, G. Murali, and J. F. Solus, "Identification and characterization of new human medium reiteration frequency repeats," *Nucleic Acids Research*, vol. 21, no. 5, pp. 1273–1279, 1993.

42. J. Jurka and A. Milosavljevic, "Reconstruction and analysis of human alu genes," *Journal of Molecular Evolution*, vol. 32, pp. 105–121, 1991.

43. J. Jurka and T. Smith, "A fundamental division in the Alu family of repeated sequences," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 4775–4778, 1988.

44. S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," *Proc. Natl. Acad. Sci. USA*, vol. 87, pp. 2264–2268, 1990.

45. A. Kornberg, *DNA replication*, W. H. Freeman, New York, NY, 1980.

46. G. M. Landau and U. Vishkin, "Fast parallel and serial approximate string matching," *Journal of Algorithms*, vol. 10, no. 2, pp. 157–169, 1989.

47. A. Lapedes, C. Barnes, C. Burks, R. Farber, and K. Sirotkin, "Application of neural networks and other machine learning algorithms to dna sequence analysis," in *Computers and DNA, SFI Studies in the Sciences of Complexity VII*, 1989.

48. C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, 1993.

49. S.-Y. Le, K. M. Currey, R. Nussinov, and J. V. Maizel, "Studies of frequently recurring substructures in human alpha-like globin mRNA precursors," *Compututer Methods Medicine*, vol. 20, pp. 563–582, 1987.

50. D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches," *Science*, vol. 227, pp. 1435–1441, 1985.

51. R. J. Lipton, J. F. Naughton, and D. A. Schneider, "Practical selectivity estimation through adaptive sampling," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, Atlantic City, NJ, pp. 1–12, May 1990.

52. A. V. Lukashin, V. V. Anshelevich, B. R. Amirikyan, A. I. Gragerov, and M. D. Frank-Kamenetskii, "Neural network models for promoter recognition," *Journal of Biomolecular Structure Dynamics*, vol. 6, pp. 1123–1133, 1989.

53. H. Margalit, B. A. Shapiro, A. B. Oppenheim, and J. V. M. Jr, "Detection of common motifs in RNA secondary structures," *Nucleic Acids Research*, vol. 17, no. 12, pp. 4829–4845, 1989.

54. E. M. McCreight, "A space-economical suffix tree construction algorithm," *Journal of the ACM*, vol. 23, pp. 262–272, 1976.

55. B. Michot and J.-P. Bachellerie, "Secondary structure of the 5' external transcribed spacer of vertebrate pre-rRNA – presence of phylogenetically conserved features," *European Journal of Biochemistry*, vol. 195, pp. 601–609, 1991.

56. A. Milosavljevic, "Discovering dependencies via algorithmic mutual information: a case study in dna sequence comparisons," *Machine Learning*, vol. 21, pp. 35–50, 1995.

57. A. Milosavljevic, "DNA sequence recognition by hybridization to short oligomers," *Journal of Computational Biology*, vol. 2, pp. 355–370, 1995.

58. A. Milosavljevic and J. Jurka, "Discovering simple DNA sequences by the algorithmic significance method," *Computer Applications in Biosciences*, vol. 9, pp. 407–411, 1993.

59. A. Milosavljevic and J. Jurka, "Discovery by minimal length encoding: a case study in molecular evolution," *Machine Learning*, vol. 12, pp. 71–87, 1993.

60. A. Milosavljevic, Z. Strezoska, M. Zeremski, D. Grujic, T. Paunesku, and R. Crkvenjakov, "Clone clustering by hybridization," *Genomics*, vol. 27, pp. 83–89, 1995.

61. M. E. Mulligan and W. R. McClure, "Analysis of the occurrence of promoter-sites in dna," *Nucleic Acids Research*, vol. 14, pp. 109–126, 1986.

62. P. O'Farrell and P. Leopold, *Cold Spring Harbor Symposia on Quantitative Biology*, vol. LVI, ch. A consensus of cyclin sequences reveals homology with the ras oncogene, Cold Spring Harbor Press, Cold Spring Harbor, NY, 1991.

63. A. Ogiwara, I. Uchiyama, Y. Seto, and M. Kanehisa, "Construction of a dictionary of sequence motifs that characterize groups of related proteins," *Protein Engineering*, vol. 5, no. 6, pp. 479–488, 1992.

64. F. Olken and D. Rotem, "Random sampling from B^+ trees," in *Proceedings of the 15th International Conference on Very Large Data Bases*, Amsterdam, The Netherlands, pp. 269–278, Auguest 1989.

65. M. C. O'Neill and F. Chiafari, "Escherichia coli promoters. ii. a spacing-class dependent promoter search protocol," *Journal of Biological Chemistry*, vol. 264, pp. 5531–5534, 1989.

66. W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," in *Proc. Natl. Acad. Sci. USA*, pp. 2444-2448, 1988.

67. Y. Quentin, "The Alu family developed through successive waves of fixation closely connected with primate lineage history," *Journal of Molecular Evolution*, vol. 27, pp. 194–202, 1988.

68. J. Quinqueton and J. Moreau, "Application of learning techniques to splicing site recognition," *Biochimie*, pp. 541–548, 1985.

69. M. A. Roytberg, "A search for common patterns in many sequences," *Computer Applications in Biosciences*, vol. 8, no. 1, pp. 57–64, 1992.

70. J. Sallantin, J. Haiech, and F. Rodier, "Search for promoter sites of prokaryotic dna using learning techniques," *Biochimie*, pp. 549–553, 1985.

71. T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht, "Information content of binding sites on nucleotide sequences," *Journal of Molecular Biology*, vol. 188, pp. 415–431, 1986.

72. R. M. Schwartz and M. O. Dayhoff, "Atlas of protein sequence and structure," in *Natl. Biomed. Res. Found., Washington, DC, Vol. 5, Suppl. 3*, pp. 353-358, 1978.

73. D. Shasha, J. T. L. Wang, K. Zhang, and F. Y. Shih, "Exact and approximate algorithms for unordered tree matching," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 668–678, April 1994.

74. D. Shasha and T. L. Wang, "New techniques for best-match retrieval," *ACM Transactions on Information Systems*, vol. 8, no. 2, pp. 140–158, April 1990.

75. J. W. Shavlik, G. G. Towell, and M. O. Noordewier, "Using knowledge-based neural networks to refine existing biological theories," in *Proc. 2nd Int. Conf. on Bioinformatics, Supercomputing and Complex Genome Analysis*, St. Petersburg, FL, pp. 377–390, July 1992.

76. R. F. Smith and T. F. Smith, "Automatic generation of primary sequence patterns from sets of related protein sequences," *Proc. Natl. Acad. Sci. USA*, vol. 87, pp. 118–122, 1990.

77. R. Staden, "Computer methods to locate signals in nucleic acid sequences," *Nucleic Acids Research*, vol. 12, pp. 505–519, 1984.

78. R. Staden *Meth. Enzymol.*, vol. 183, pp. 193–211, 1990.

79. G. D. Stormo and G. W. H. III, "Identifying protein-binding sites from unaligned dna fragments," *Proc. Natl. Acad. Sci. USA*, vol. 86, pp. 1183-1187, 1989.

80. G. M. Studnicka, "Nucleotide sequence homologies in control regions of prokaryotic genomes," *Gene*, vol. 58, pp. 45–57, 1987.

81. R. L. Tatusov, S. F. Altschul, and E. V. Koonin, "Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks," *Proc. Natl. Acad. Sci. USA*, vol. 91, pp. 12091–12095, 1994.

82. R. L. Tatusov and E. V. Koonin, "A simple tool to search for sequence motifs that are conserved in blast outputs," *Computer Applications in Biosciences*, vol. 10, pp. 457–459, 1994.

83. W. R. Taylor, "Identification of protein sequence homology by consensus template alignment," *Journal of Molecular Biology*, vol. 188, pp. 233–258, 1986.

84. E. Ukkonen, "Finding approximate pattern in strings," *Journal of Algorithms*, vol. 6, pp. 132–137, 1985.

85. M. Vingron and P. Argos, "A fast and sensitive multiple sequence alignment algorithm," *Computer Applications in Biosciences*, vol. 5, pp. 115–122, 1989.

86. R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974.

87. J. C. Wallace and S. Henikoff, "PATMAT: A searching and extraction program for sequence, pattern and block queries and databases," *Computer Applications in Biosciences*, vol. 8, no. 3, pp. 249–254, 1992.

88. J. T. L. Wang, G. W. Chirn, T. G. Marr, B. A. Shapiro, D. Shasha, and K. Zhang, "Combinatorial pattern discovery for scientific data: Some preliminary results," in *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, MN, pp. 115–125, May 1994.

89. J. T. L. Wang, T. G. Marr, D. Shasha, 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, pp. 2769–2775, 1994.

90. J. T. L. Wang, K. Zhang, and G. W. Chirn, "Algorithms for approximate graph matching," *Information Sciences*, vol. 82, pp. 45–74, 1995.

91. J. T. L. Wang, K. Zhang, K. Jeong, and D. Shasha, "A system for approximate tree matching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 4, pp. 559–571, August 1994.

92. M. S. Waterman, ed., *Mathematical Methods for DNA Sequence Analysis*, CRC Press, Boca Raton, FL, 1989.

93. C. Willard, H. T. Nguyen, and C. W. Schmid, "Existence of at least three distince Alu subfamilies," *Journal of Molecular Evolution*, vol. 26, pp. 180–186, 1987.

94. S. Wu and U. Manber, "Fast text searching allowing errors," *Communications of the ACM*, vol. 35, no. 10, pp. 83–91, October 1992.

95. K. Zhang, D. Shasha, and J. T. L. Wang, "Approximate tree matching in the presence of variable length don't cares," *Journal of Algorithms*, vol. 16, no. 1, pp. 33–66, January 1994.

96. G. K. Zipf, *Human Behavior and the Principle of Least Effort*, Addison Wesley, Reading, MA, 1949.