New Techniques for DNA Sequence Classification

Jason T. L. Wang[*] Steve Rozen[†] Bruce A. Shapiro[‡]

Dennis Shasha[§] Zhiyuan Wang[¶] Maisheng Yin[||]

Corresponding author: Jason T. L. Wang (jason@cis.njit.edu)
Phone: (973) 596-3396; Fax: (973) 596-5777

Running head: New Techniques for DNA Sequence Classification

Key words: algorithms, consensus sequence, pattern matching, tools for computational biology, DNA sequence recognition

[*]Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA.

[†]Center for Genome Research, Whitehead Institute for Biomedical Research, Massachusetts Institute of Technology, One Kendall Square, Cambridge, MA 02139, USA.

[‡]Image Processing Section, Laboratory of Experimental and Computational Biology, Division of Basic Sciences, National Cancer Institute, National Institutes of Health, Frederick, MD 21702, USA.

[§]Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA.

[¶]Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA.

[||]Wyeth-Ayerst Research, Princeton, NJ 08543, USA.

ABSTRACT

DNA sequence classification is the activity of determining whether or not an unlabeled sequence S belongs to an existing class $\mathcal{C}$. This paper proposes two new techniques for DNA sequence classification. The first technique works by comparing the unlabeled sequence S with a group of active motifs discovered from the elements of $\mathcal{C}$ and by distinction with elements outside of $\mathcal{C}$. The second technique generates and matches gapped fingerprints of S with elements of $\mathcal{C}$. Experimental results obtained by running these algorithms on long and well conserved Alu sequences demonstrate the good performance of the presented methods compared with FASTA. When applied to less conserved and relatively short functional sites such as splice-junctions, a variation of the second technique combining fingerprinting with consensus sequence analysis gives better results than the current classifiers employing text compression and machine learning algorithms.

INTRODUCTION

DNA sequence classification is an important problem in computational biology (Gelfand, 1995). 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 (i.e., a composite subsequence created by taking the majority base at each position in a multiple alignment of the sequences in $\mathcal{C}$) which is then used to identify sequences in uncharacterized DNA (Staden, 1984; Galas *et al.*, 1985; Mulligan and McClure, 1986; Berg and von Hippel, 1987; Studnicka, 1987; O'Neill and Chiafari, 1989; Gelfand, 1995);

- 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 or not the unlabeled sequence S belongs to $\mathcal{C}$ (Quinqueton and Moreau, 1985; Sallantin *et al.*, 1985; Lukashin *et al.*, 1989; Lapedes *et al.*, 1990; Hirst and Sternberg, 1992; Shavlik *et al.*, 1992; Hirsh and Noordewier, 1994; Gelfand, 1995; Loewenstern *et al.*, 1995);

- sequence alignment – this approach aligns the unlabeled sequence S with members of $\mathcal{C}$ using an existing tool such as FASTA (Lipman and Pearson, 1985; Pearson and Lipman, 1988) and assigns S to $\mathcal{C}$ if the best alignment score for S is sufficiently high.

In this paper we propose two new techniques for calculating scores to classify DNA sequences. Our approach works by first randomly selecting (from a uniform distribution without replacement) 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," and calculate, for each positive training sequence, a score with respect to the base sequences. (Each technique has its own way of calculating the score.) 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 $B_{low} = \min\,\{L_p, U_n\}$. (In our cases, however, $L_p > U_n$.) When classifying the unlabeled sequence S, we calculate S's score with respect to the base sequences, denoted c. If $c \geq B_{high}$, then S is classified to be a member of $\mathcal{C}$. 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 the algorithms in detail in the following sections. To evaluate the performance of our algorithms, we apply them to two types of DNA sequences. When running on long and well conserved Alu sequences (Jurka and Milosavljevic, 1991; Jurka, 1993; Jurka *et al.*, 1993; Claverie and Makalowski, 1994), both of our classifiers work well. When running on less conserved and relatively short sequences such as splice-junctions (Brunak and Engelbrecht, 1991; Noordewier *et al.*, 1991), combining our techniques with a consensus sequence analysis achieves good performance.

ALGORITHM FOR CLASSIFIER I

Given the set $\mathcal{B}$ of base sequences, Classifier I first searches for *active motifs* that approximately match the sequences in $\mathcal{B}$ using our previously developed tool DISCOVER (Wang *et al.*, 1994; Chirn, 1996). A motif (or a segment) here is a consecutive substring of a nucleotide sequence. For example, consider the three base sequences in Fig. 1. Suppose the motif to be sought has length greater than 6 (i.e., it contains at least 7 nucleotides), occurrence number 3 (i.e., it matches all three base sequences), and mutation 1 (i.e., one mismatch, insertion or deletion of a nucleotide is allowed when matching the motif with a base sequence). Then `GCCGGGC` and `GCCAGGC` underlined in Fig. 1 are two qualified motifs.

$$\begin{array}{l}
\texttt{GGAGAG}\underline{\texttt{GCCGGGC}}\texttt{GTGTGCCGGTAC} \\
\texttt{G}\underline{\texttt{GCCAGGC}}\texttt{GGCAGATCTTGACCAGG} \\
\texttt{TGTAATCAGAGC}\underline{\texttt{GCCAGGC}}\texttt{AAACAT}
\end{array}$$

FIG. 1. Three base sequences and two active motifs underlined in the sequences. The active motifs have length greater than 6 (i.e., they contain at least 7 nucleotides), occurrence number 3 (i.e., they match all three base sequences), and mutation 1 (i.e., one mismatch, insertion or deletion of a nucleotide is allowed when matching the motifs with a base sequence).

Let $\mathcal{R}$ be a set of active motifs discovered from the set $\mathcal{B}$ of base sequences. Given a sequence S (which could be a training or an unlabeled sequence), the score between S and a motif $P \in \mathcal{R}$, 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 score of S with respect to the base sequences is defined as

$$score(S) = \max\{score(S, P) | P \in \mathcal{R}\} \times 100.$$

ALGORITHM FOR CLASSIFIER II

Like FASTA, Classifier II adopts a hash table method and calculates the score of a sequence by fingerprint matching (Blaisdell, 1986; Mironov and Alexandrov, 1988; Califano and Rigoutsos, 1993; Wang *et al.*, 1996). Let S be a sequence and let Seg be a segment (i.e., consecutive substring) of S. A gapped fingerprint f of Seg is a subsequence (possibly non-consecutive substring) of Seg that begins with the segment's first nucleotide. A gap at position p means that when forming f, we do not pick the nucleotide at position p in Seg. For example, let $S =$ `ACGTTGCA`. Then $Seg =$ `ACGTTG` is a segment of S and `ATT` is a fingerprint of Seg with two gaps (one gap at position 2 and one gap at position 3). The number of gaps in the fingerprints is bounded by a parameter gap. As an example, suppose $gap = 2$. Then, the set of

3-nucleotide fingerprints for Seg includes: 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).

Given the set $\mathcal{B}$ of base sequences, we pick the segments from each base sequence and hash each fingerprint of the segments into a file. (See the Appendix for an example of building and using fingerprint files.) 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, and compare them with the fingerprints generated from the base sequences. When a match between a fingerprint starting at the p^{th} position in S and a fingerprint starting at the q^{th} position in a base sequence B occurs, we add one to the score of position $q - p + 1$ in B. (See the Appendix for the details of the scoring algorithm.)

For example, consider the base sequence S_1 = ACGTTGCA and the sequence S = CGATGCAT in Fig. 2. Consider the 3-nucleotide fingerprint TGC starting at the 5^{th} position in S_1, and the same TGC starting at the 4^{th} position in S. Thus $q = 5$ and $p = 4$. We add one to the score of position q - p + 1 = 5 - 4 + 1 = 2 in S_1. Intuitively if we align the first nucleotide of S with the 2^{nd} nucleotide of S_1, we can see a match between the two corresponding fingerprints. In general, if one aligns the first nucleotide of S with a position in S_1 that obtains n scores in total, one can see n matches of fingerprints in the alignment. Thus, aligning the first nucleotide of S with the position in S_1 with the highest score may yield the best alignment between the two sequences. This technique was pioneered by Califano and Rigoutsos (1993) for finding the best alignment between two DNA sequences.

```
S1: ACGTTGCA
        |||
S:   CGATGCAT
```

FIG. 2. Illustration of the scoring algorithm used in Classifier II. By aligning the first nucleotide of S with the 2^{nd} nucleotide of the base sequence S_1, one can see a match of the fingerprint TGC. In this situation, the algorithm adds one to the score of position 2 in S_1.

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 of B after applying the scoring algorithm to the sequence S and the base sequences in $\mathcal{B}$. The score of B, denoted $score(B)$, is defined to be

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

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

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

EXPERIMENTS

The algorithms for the proposed classifiers were implemented in C on a Sun SPARCstation 20 running the operating system Solaris version 2.4. We compared the relative performance of the algorithms by applying them to classifying Alu sequences (Jurka *et al.*, 1993; Claverie and Makalowski, 1994). 327 Alu sequences were selected from NCBI's database (ftp at ncbi.nlm.nih.gov/pub/jmc/alu/ALU.327.dna.ref). Among them, 100 were used as base sequences, 100 were used as positive training sequences, and the other 127 were treated as unlabeled test sequences. The lengths of these sequences ranged from 69 bp to 379 bp. In constructing non-Alu data, following the studies in (Lukashin *et al.*, 1989; Demeler and Zhou, 1991), we randomly generated 1,200 sequences that retained the correct nucleotide frequencies using the simulation programming package SIMSCRIPT (Kiviat *et al.*, 1983). Among the data, 100 were used as negative training sequences and the other 1,100 were also treated as unlabeled test sequences. The lengths of these sequences ranged from 100 bp to 240 bp.

Classifier I found active motifs from the 100 base sequences using our previously developed tool DISCOVER (Wang *et al.*, 1994; Chirn, 1996). Given a set of sequences $\mathcal{S}$, the DISCOVER tool can locate all the active motifs P where P is within the allowed Mut mutations of at least $Occur$ sequences in $\mathcal{S}$ and $|P| \geq Length$, where $|P|$ represents the number of nucleotides in P. (Mut, $Occur$ and $Length$ are user-specified parameters.) The motifs used here had length greater than or equal to 11, occurrence number greater than or equal to 15 and mutation 0 (i.e., these motifs matched at least 15 base sequences without mutation). There were 556 motifs, with lengths ranging from 11 bp to 22 bp. Classifier II fixed the segment length at 8 and gap at 0. Table 1 summarizes the parameters and their base values used in the experiments.

Parameter	Value	Description
$\lvert\mathcal{B}\rvert$	100	Number of base sequences (Alu)
$\lvert\mathcal{T}_P\rvert$	100	Number of positive training sequences (Alu)
$\lvert\mathcal{T}_N\rvert$	100	Number of negative training sequences (non-Alu)
$NumTest$	1,227	Number of unlabeled test sequences (Alu & non-Alu)
$Length$	11	Minimum length of active motifs used in Classifier I
$Occur$	15	Minimum occurrence number of active motifs used in Classifier I
Mut	0	Allowed mutation between an active motif and a base sequence used in Classifier I
n	8	Segment length used in Classifier II
gap	0	Number of gaps allowed in Classifier II

TABLE 1. Experimental parameters and their base values used in performance analysis of the two studied classifiers.

The metrics used to evaluate the effectiveness of our classification algorithms were 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, 1,227 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 to belong to $\mathcal{C}$ by that algorithm.)

Our experimental results showed that for Classifier I, $B_{high} = 1100$, $B_{low} = 1000$; for Classifier II, $B_{high} = 68$, $B_{low} = 65$. Classifier I's $PR = 99.5\%$ and $NR = 0.0\%$. Classifier II's $PR = 99.3\%$ and $NR = 0.08\%$. For comparison purposes, we also ran FASTA v 3.0t6 (Smith and Waterman, 1981; Lipman and Pearson, 1985; Pearson and Lipman, 1988; Waterman, 1995) on the same base and training sequences. We recorded the highest Smith-Waterman score for the negative training data (i.e., the negative upper bound) and the lowest Smith-Waterman score for the positive training data (i.e., the positive lower bound). The negative upper bound obtained was 98 and the positive lower bound was 237. As for the proposed classifiers, FASTA classified an unlabeled test sequence by matching it with the base data and comparing its score with the two bounds. The PR obtained was 98.04% and the NR was 1.96%. As can be seen, the two proposed classifiers beat FASTA in PR by more than 1%.

Table 2 shows the number of false positives, false negatives, no-opinion$^{(-)}$'s, and no-opinion$^{(+)}$'s for each studied classifier. False positives were randomly generated sequences that a classifier falsely classified as Alu's. False negatives were Alu's that the classifier falsely claimed as non-Alu's. No-opinion$^{(-)}$'s (no-opinion$^{(+)}$'s, respectively) were randomly generated sequences (Alu's, respectively) to which the classifier gave the "no opinion" verdict. It was found that gb|X12818_HSAL000908 (Alu-J) in the file ALU.327.dna.ref was a false negative of Classifier I while receiving the "no opinion" verdict from FASTA. The Alu sequence gb|X51502_HSAL000586 was a false negative of Classifier II.

	number of false positives	number of false negatives	number of no-opinion$^{(-)}$'s	number of no-opinion$^{(+)}$'s
Classifier I	5	1	0	0
Classifier II	6	1	1	0
FASTA	0	0	23	1

TABLE 2. Performance analysis of the three studied classifiers. False positives are randomly generated sequences that a classifier falsely classifies as Alu's. False negatives are Alu's that the classifier falsely claims as non-Alu's. No-opinion$^{(-)}$'s (no-opinion$^{(+)}$'s, respectively) are randomly generated sequences (Alu's, respectively) to which the classifier gives the "no opinion" verdict. The two proposed classifiers beat FASTA in precision rates; the latter is a more conservative classifier in the sense that it generates no errors while giving 24 "no opinion" verdicts.

We next 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 1. It was found that both Classifiers I and II behaved stably with varying $|\mathcal{B}|$, $|\mathcal{T}_P|$ and

$|\mathcal{T}_N|$. The relative sizes of the base, positive training and negative training data sets had little impact on the performance of the classifiers, provided that these sets were sufficiently large (e.g., with size ≥ 100). The performance of Classifier I degraded as the occurrence number required for the motifs became large. We found that the discovered motifs in Alu sequences generally had low occurrence numbers. When the occurrence number used by Classifier I was, for example, fixed at 30, very few motifs were discovered and thus they could not well characterize the sequences. Likewise, using short active motifs (e.g., with their length fixed at 5) and large mutations (e.g., $Mut = 3$) yielded poor performance, since these segments might appear, by chance, in both Alu and non-Alu sequences. No trend was evident with regard to n and gap used in Classifier II. However, programs using a small gap (e.g., $gap = 0$) ran much faster than programs using a large gap (e.g., $gap = 4$).

In sum, to achieve good performance, one should use sufficiently many (e.g., 100) sequences as base, positive training and negative training data when constructing the classifiers. For Classifier I, using long active motifs (e.g., with $Length = 11$) with a small mutation value (e.g., with $Mut \leq 1$) is good. Such motifs appear quite uniquely in Alu sequences and therefore characterize the sequences well. For Classifier II, using segments of a reasonable length (e.g., with the length greater than or equal to 6) without gaps is good, as it requires less running time and space while achieving an acceptably high precision rate.

DISCUSSION

The two proposed classifiers worked well for long and conserved Alu sequences. An interesting question was whether or not their performance changed for arbitrarily chosen DNA sequences and for relatively short, less conserved functional sites. We considered some newly sequenced chromosomes in the libraries Lib.5 and Lib.373 available from NCBI's web site `http://inhouse.ncbi.nlm.nih.gov/cgi-bin/UniGene/lbrowse?ORG=Hs`. 327 Hippocampus II sequences were chosen, among which 100 were used as base data, 100 were used as positive training data, and the other 127 were treated as unlabeled test data. The lengths of the Hippocampus II sequences ranged from 120 bp to 702 bp. In addition, 188 stratagenes, subtracted Hippocampus were chosen, among which 100 were used as negative training data and the other 88 were treated as unlabeled test data. The lengths of the stratagenes ranged from 138 bp to 953 bp. It was found that the pattern-based classifier suffered for these sequences (with $PR < 70\%$). On the other hand, using $n = 6$ and $gap = 0$, the fingerprint-based classifier achieved a 97.2% PR and a 2.8% NR, with $B_{high} = 204$ and $B_{low} = 34$.

We also tested our algorithms on a set of splice-junction donors. 767 donors were selected from UC Irvine's database (at ftp.ics.uci.edu/pub/machine-learning-databases/molecular-biology). This dataset is substantially the same as used in (Noordewier *et al.*, 1991; Loewenstern *et al.*, 1995). Among the data, 250 were used as base sequences, 250 were used as positive training sequences, and the rest were treated as unlabeled test sequences. In addition, 1,655 non-donor sequences were selected from the same database, among which 250 were used as negative training sequences and the other 1,405 were also treated as unlabeled test sequences. All sequences were 60 bp long.

Our experimental results showed that when running on these splice-junction donors, using active segments or fingerprints alone had unsatisfactory performance; their precision rates were below 70%. However, a hybrid technique combining fingerprints and consensus sequences achieved good performance. Consensus comparison has been used by many researchers for recognizing sequence family membership (Scherer *et al.*, 1978; Galas *et al.*, 1985; Mulligan and McClure, 1986; O'Neill, 1989; O'Neill and Chiafari, 1989). A common practice is to pre-filter sequences for highly repetitive and/or low information content subsequences prior to identifying overall sequence similarity (Altschul *et al.*, 1990; Karlin *et al.*, 1990). In our case, splice-junction donors contained consensus sequences of 9 bp with GT dinucleotide appearing in almost all of the consensus sequences (Lerner *et al.*, 1980; Mount, 1982; Iida, 1987; Kudo *et al.*, 1992; Alberts *et al.*, 1994). In our empirical study, we found that the donors were aligned with the GT dinucleotide being at the same positions 31-32 in the UCI database. We filtered out irregular donors (i.e. those without GT in the positions 31-32) from the base and positive training sequences.

We define the *interesting site* of a sequence to be a substring of it with length 9, comprising the GT dinucleotide, together with three nucleotides on the left and four nucleotides on the right of the GT dinucleotide. Each base and positive training donor sequence contains a unique interesting site whose GT dinucleotide is taken from positions 31-32 of the sequence. On the other hand, an unlabeled test sequence or a negative training non-donor sequence may contain no interesting site, i.e. no GT is found in the sequence. Or, there may exist, by chance, several interesting sites, all containing the GT dinucleotide, in the sequence.

Our hybrid algorithm combining fingerprints and consensus sequences works as follows. When generating fingerprints from a base sequence, we focus only on its interesting site, segment the interesting site, and discard segments with length less than 6. (The parameter value 6 is chosen based on the previous experimental results.) From each resulting segment of length n, $6 \leq n \leq 9$, we generate non-gapped fingerprints from it with lengths ranging from 6 to n. This results in ten fingerprints for each base sequence. These fingerprints are then hashed into fingerprint files, where each fingerprint is marked by its starting position in the interesting site. Duplicate fingerprints with the same starting position are stored only once.

When calculating the score of a sequence S (whether it is a training or an unlabeled test sequence), we take its interesting site(s), generate fingerprints from the site(s) and hash the fingerprints using the same hash functions as for the base sequences. Let f be a fingerprint generated from an interesting site of S. Define the score of f to be 2^m where m is the length of f if f matches a fingerprint f' of some base sequence, or 0 otherwise. Two fingerprints f and f' match if they have the same starting position and the same nucleotides. The score of the interesting site equals the sum of the scores of the ten fingerprints generated from it. For an unlabeled test sequence or a negative training non-donor sequence, if no GT is found, its score is defined as 0. If many interesting sites exist in the sequence, each interesting site obtains a score and the score of the sequence equals the maximum of the scores of all the interesting sites. For a positive training donor sequence, its score equals the score of the unique interesting site in it.

Our experimental results showed that this hybrid algorithm achieved a 93% precision rate and a 2.5% no-opinion rate. This is higher than the 88 – 89% precision rates obtained from the current best splice-junction donor classifiers on the same dataset that use text compression and machine learning methods (Loewenstern *et al.*, 1995). We have made the code of the algorithm available on the Internet; please visit the web site at `http://www.cis.njit.edu/~discdb` for details. Interested readers may also contact the authors directly to get the programs (email jason@cis.njit.edu or shasha@cs.nyu.edu).

APPENDIX

Algorithm for Building Fingerprint Files

Let S be a sequence in the base dataset $\mathcal{B}$. We take every segment, denoted by Seg, of length n from S and generate all gapped fingerprints from Seg of length from 2 to $n-1$. Each fingerprint 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 nucleotide of f occurs at the y^{th} position in that sequence. We use a hash function h_k, $2 \le k \le n-1$, to hash all fingerprints of length k to a fingerprint file $\mathcal{F}_k$.

Example 1
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`. Table 3 shows all gapped fingerprints generated from the segment `ACGTTG`, with $gap = 2$.

Let f = `XYZ` be a fingerprint of length 3. Suppose the hash function h_3 is $h_3(f) = (num(\texttt{X}) \times 4^2 + num(\texttt{Y}) \times 4^1 + num(\texttt{Z})) \bmod 7$, where $num(\texttt{X})$ is `X`'s ASCII value minus 64. Figure 3 shows the fingerprint file $\mathcal{F}_3$ for the three 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 at the 2^{nd} position in S_3.
End of Example

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

TABLE 3. Gapped fingerprints (of lengths 2, 3, 4, 5, respectively) generated from the segment `ACGTTG`. The segment length is 6 and the number of gaps allowed in generating the fingerprints is 2 (i.e., $gap = 2$).

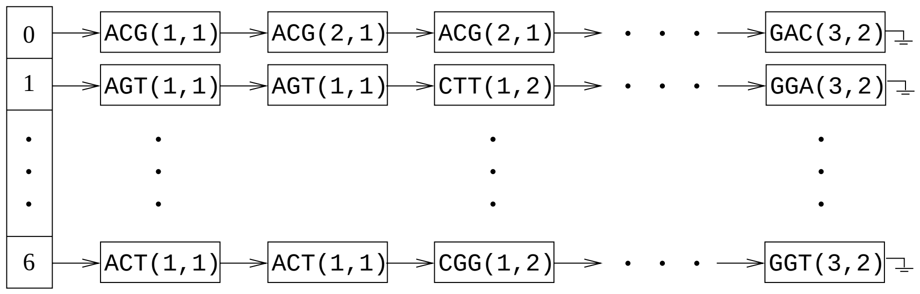

FIG. 3. The fingerprint file $\mathcal{F}_3$ for the three base sequences S_1 = ACGTTGCA, S_2 = ACCAGTG and S_3 = CGGACTA in Example 1. This file contains all fingerprints of length 3 generated from the base sequences. Each fingerprint f (e.g. GGA) is associated with a pair of integers (x, y). This pair serves as the position marker for f, where x indicates that f is generated from a segment of the x^{th} base sequence and y means that the first nucleotide of f occurs at the y^{th} position in that sequence. Thus, GGA(3, 2) for example means that the fingerprint GGA is generated from S_3 and it starts at the 2^{nd} position in S_3. Note that some entries are duplicate (e.g., AGT(1, 1) in bucket 1); they represent fingerprints with different numbers of gaps.

Algorithm for Scoring

Let S be a sequence (whether it is a training or an unlabeled test sequence). When comparing S's figureprints with the fingerprints generated from the base sequence set $\mathcal{B}$, we give scores to appropriate positions in the base sequences as shown in Figure 4. The result is a histogram of scores on the base sequences. The score of a base sequence B is the maximum of the scores of the positions in B. The score of S with respect to the base sequences of $\mathcal{B}$ is the maximum of the scores of all sequences in $\mathcal{B}$, multiplied by a scaling factor $100/|S|$.

```
Input: A sequence $S$, a set $\mathcal{B}$ of base sequences and $\mathcal{B}$'s fingerprint files.
Output: A histogram of scores 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 to the score of position $q - p + 1$ in the $i^{th}$ base sequence in $\mathcal{B}$;
            end;
    end;
```

FIG. 4. Algorithm Scoring used in Classifier II. By matching the fingerprints of the given sequence S with the fingerprints of base sequences in the set $\mathcal{B}$, this algorithm builds a histogram of scores on the base sequences in $\mathcal{B}$. The score of a base sequence B is the maximum of the scores of the positions in B. The score of S with respect to the base sequences of $\mathcal{B}$ is the maximum of the scores of all sequences in $\mathcal{B}$, multiplied by a scaling factor $100/|S|$.

Example 2

Suppose we are given a sequence S = CGATGCAT. Figure 5 shows the histogram obtained after matching S's fingerprints with the fingerprints of the base sequences in Example 1.

End of Example

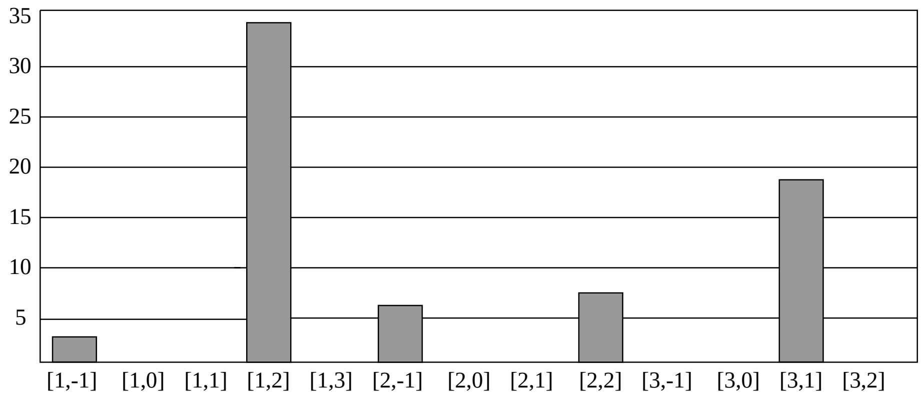

FIG. 5. The histogram obtained after matching fingerprints of the sequence S = CGATGCAT in Example 2 with the fingerprints of the base sequences S_1 = ACGTTGCA, S_2 = ACCAGTG and S_3 = CGGACTA in Example 1. The fingerprints are generated from segments of length 6 and the number of gaps allowed in generating the fingerprints is 2. Each $[i, q]$ on the x-axis represents the q^{th} position in the i^{th} base sequence. The y-axis shows scores on the positions of the base sequences. In the figure, [1, -1] scores 3, [1, 2] scores 34, [2, -1] scores 6, [2, 2] scores 7, and [3, 1] scores 19. All the other positions score 0. Thus S_1 scores 34, S_2 scores 7 and S_3 scores 19. The score of S with respect to the base sequences is $(34/8) \times 100 = 425$.

ACKNOWLEDGMENTS

The authors thank the anonymous referees for their constructive criticism and useful comments which helped to improve both the quality and presentation of this paper. We also thank Dr. Gung-Wei Chirn of Hoffmann-La Roche Inc. for his contributions in the early phase of this project as well as Drs. Nat Goodman and Lincoln Stein of the Whitehead Institute for helpful discussions. This work was supported by the National Science Foundation under grant numbers IRI-9224601, IRI-9224602, IRI-9531548, IRI-9531554, and by the Office of Naval Research under grant number N00014-92-J-1719.

References

[1] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J.D. 1994. *Molecular Biology of the Cell*, 3rd ed. Garland Publishing, Inc., New York and London.

[2] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. A basic local alignment search tool. *J. Mol. Biol.* 215, 403–410.

[3] Berg, O.G., and von Hippel, P.H. 1987. Selection of DNA binding sites by regulatory proteins: Statistical-mechanical theory and application to operators and promoters. *J. Mol. Biol.* 193, 723–750.

[4] Blaisdell, B.E. 1986. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proc. Nat. Acad. Sci. USA* 83, 5155–5159.

[5] Brunak, S., and Engelbrecht, J. 1991. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *J. Mol. Biol.* 220, 49–65.

[6] Califano, A., and Rigoutsos, I. 1993. FLASH: A fast look-up algorithm for string homology. *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, 56–64.

[7] Chirn, G.W. 1996. Pattern discovery in sequence databases: Algorithms and applications to DNA/protein classification. Ph.D. Dissertation, Department of Computer and Information Science, New Jersey Institute of Technology.

[8] Claverie, J.-M., and Makalowski, W. 1994. Alu alert. *Nature* 371, 752.

[9] Demeler, B., and Zhou, G. 1991. Neural network optimization for *E. coli* promoter prediction. *Nucleic Acids Res.* 19, 1593–1599.

[10] Galas, D.J., Eggert, M., and Waterman, M.S. 1985. Rigorous pattern-recognition methods for DNA sequences: Analysis of promoter sequences from *Escherichia coli. J. Mol. Biol.* 186, 117–128.

[11] Gelfand, M.S. 1995. Prediction of function in DNA sequence analysis. *J. Comput. Biol.* 2, 87–115.

[12] Hirsh, H., and Noordewier, M. 1994. Using background knowledge to improve inductive learning of DNA sequences. *Proceedings of the IEEE Conference on Artificial Intelligence for Applications.*

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

[14] Iida, Y. 1987. DNA sequences and multivariate statistical analysis – Categorical discrimination approach to 5' splice site signals of mRNA precursors in higher eukaryotes' genes. *Comput. Appl. Biosci.* 3, 93–98.

[15] Jurka, J. 1993. A new subfamily of recently retroposed human Alu repeats. *Nucleic Acids Res.* 21, 2252.

[16] Jurka, J., Kaplan, D.J., Duncan, C.H., Walichiewicz, J., Milosavljevic, A., Murali, G., and Solus, J.F. 1993. Identification and characterization of new human medium reiteration frequency repeats. *Nucleic Acids Res.* 21, 1273–1279.

[17] Jurka, J., and Milosavljevic, A. 1991. Reconstruction and analysis of human Alu genes. *J. Mol. Evol.* 32, 105–121.

[18] Karlin, S., Blaisdell, B.E., and Brendel, V. 1990. Identification of significant sequence patterns in proteins. *Methods in Enzymology* 183, 388–402.

[19] Kiviat, P.J., Markowitz, H.M., and Villanueva, R. 1983. *SIMSCRIPT II.5 Programming Language.* CACI, Los Angeles, CA.

[20] Kudo, M., Kitamura-Abe, S., Shimbo, M., and Iida, Y. 1992. Analysis of context of 5'-splice site sequences in mammalian mRNA precursors by subclass method. *Comput. Appl. Biosci.* 8, 367–376.

[21] Lapedes, A., Barnes, C., Burks, C., Farber, R., and Sirotkin, K. 1990. Application of neural networks and other machine learning algorithms to DNA sequence analysis, 157–182. *In* Bell, G.I., and Marr, T.G., eds., *Computers and DNA* (SFI Studies in the Sciences of Complexity, vol. VII: Proc. Workshop on Interface between Computation Science and Nucleic Acid Sequencing, Santa Fe, 1988). Addison-Wesley, Reading, MA.

[22] Lerner, M.R., Boyle, J.A., Mount, S.M., Wolin, S.L., and Steriz, J.A. 1980. Are snRNPs involved in splicing? *Nature* 283, 220–224.

[23] Lipman, D.J., and Pearson, W.R. 1985. Rapid and sensitive protein similarity searches. *Science* 227, 1435–1441.

[24] Loewenstern, D., Hirsh, H., Yianilos, P., and Noordewier, M. 1995. DNA sequence classification using compression-based induction. DIMACS Tech. Report 95-04, Rutgers University.

[25] Lukashin, A.V., Anshelevich, V.V., Amirikyan, B.R., Gragerov, A.I., and Frank-Kamenetskii, M.D. 1989. Neural network models for promoter recognition. *J. Biomol. Struct. Dynam.* 6, 1123–1133.

[26] Mironov, A.A., and Alexandrov, N.N. 1988. Statistical method for rapid homology search. *Nucleic Acids Res.* 16, 5169–5173.

[27] Mount, S.M. 1982. A catalogue of splice junction sequences. *Nucleic Acids Res.* 10, 459–472.

[28] Mulligan, M.E., and McClure, W.R. 1986. Analysis of the occurrence of promoter-sites in DNA. *Nucleic Acids Res.* 14, 109–126.

[29] Noordewier, M.O., Towell, G.G., and Shavlik, J.W. 1991. Training knowledge-based neural networks to recognize genes in DNA sequences. *In* Lippmann, R.P., Moody, J.E., and Touretzky, D.S., eds., *Neural Information Processing Systems 3*, Morgan Kaufmann, Palo Alto, CA.

[30] O'Neill, M.C. 1989. Consensus methods for finding and ranking DNA binding sites. *J. Mol. Biol.* 207, 301–310.

[31] O'Neill, M.C., and Chiafari, F. 1989. *Escherichia coli* promoters. II. A spacing class-dependent promoter search protocol. *J. Biol. Chem.* 264, 5531–5534.

[32] Pearson, W.R., and Lipman, D.J. 1988. Improved tools for biological sequence comparison. *Proc. Nat. Acad. Sci. USA* 85, 2444–2448.

[33] Quinqueton, J., and Moreau, J. 1985. Application of learning techniques to splicing site recognition. *Biochimie* 67, 541–548.

[34] Sallantin, J., Haiech, J., and Rodier, F. 1985. Search for promoter sites of prokaryotic DNA using learning techniques. *Biochimie* 67, 549–553.

[35] Scherer, G.E.F., Walkinshaw, M.D., and Arnott, S. 1978. A computer-aided oligonucleotide analysis provides a model sequence for RNA polymerase-promoter recognition in *E. coli. Nucleic Acids Res.* 5, 3759–3773.

[36] Shavlik, J.W., Towell, G.G., and Noordewier, M.O. 1993. Using knowledge-based neural networks to refine existing biological theories, 377–390. *In* Lim, H.A., Fickett, J.W., Cantor, C.R., and Robbins, R.J., eds., *Proc. 2nd Int. Conf. on Bioinformatics, Supercomputing and Complex Genome Analysis*, World Scientific, Singapore.

[37] Smith, T.F., and Waterman, M.S. 1981. Identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.

[38] Staden, R. 1984. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.* 12, 505–519.

[39] Studnicka, G.M. 1987. Nucleotide sequence homologies in control regions of prokaryotic genomes. *Gene.* 58, 45–57.

[40] Wang, J.T.L., Marr, T.G., Shasha, D., Shapiro, B., and Chirn, G.-W. 1994. Discovering active motifs in sets of related protein sequences and using them for classification. *Nucleic Acids Res.* 22, 2769–2775.

[41] Wang, J.T.L., Marr, T.G., Shasha, D., Shapiro, B., Chirn, G.-W., and Lee, T.Y. 1996. Complementary classification approaches for protein sequences. *Protein Engng.* 9, 381–386.

[42] Waterman, M.S. 1995. *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman & Hall.