[33] J. Shavlik, G. Towell, and M. Noordewier. Using artificial neural networks to refine existing biological knowledge. *International Journal of Human Genome Research*, 1:81–107, 1992.

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

[35] J. A. Steitz. *Science*, 257:888, 1991.

[36] J.A. Storer. *Data Compression: Methods and Theory*. Computer Science Press, 1988.

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

[38] John C. Wootton and Scott Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Computers Chem.*, 17(2):149–163, 1993.

[39] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. Information Theory*, 24:530–536, 1978.

[19] Aleksandar Milosavljević. Discovering sequence similarity by the algorithmic significance method. In *International Conference on Intelligent Systems for Molecular Biology*, number 1, pages 284–291, 1993.

[20] Aleksandar Milosavljević. Sequence comparisons via algorithmic mutual information. In *International Conference on Intelligent Systems for Molecular Biology*, number 2, 1994. to appear.

[21] Aleksandar Milosavljević and Jerzy Jurka. Discovering simple DNA sequences by the algorithmic significance method. *CABIOS*, 9:407–411, 1993.

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

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

[24] M. C. O'Neill. Consensus methods for finding and ranking DNA binding sites. *Journal of Molecular Biology*, 207:301–310, 1989.

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

[26] Pavel A. Pevzner. Nucleotide sequences versus Markov models. *Computers Chem.*, 16(2):103–106, 1992.

[27] Peter Salamon and Andrzej K. Konopka. A maximum entropy principle for the distribution of local complexity in naturally occurring nucleotide sequences. *Computers Chem.*, 16(2):117–124, 1992.

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

[29] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites in nucleotide sequences. *Journal of Molecular Biology*, 188:415–431, 1986.

[30] George Sebestyen. *Decision-Making Processes in Pattern Recognition*, chapter 2. Macmillian, New York, 1962.

[31] H. M. Seidel, D. L. Pompliana, and J. R. Knowles. *Science*, 257:1489, 1992.

[32] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:398–403, 1949.

[4] C. Burge, A. M. Campbell, and S. Karlin. Over- and under-representation of short oligonucleotides in DNA sequences. In *Proc. Natl. Acad. Sci.*, volume 89, pages 1358–1362, 1992.

[5] Gary A. Churchill. Hidden Markov chains and the analysis of genome structure. *Computers Chem.*, 16(2):107–115, 1992.

[6] B. Demeler and G. Zhou. Neural network optimization for E. coli promoter prediction. *Nucleic acids research*, 19:1593, 1991.

[7] G. Dykes, R. Bambara, K. Marians, and R. Wu. On the statistical significance of primary structural features found in DNA-protein interaction sites. *Nucleic Acids Research*, 2:327–345, 1975.

[8] Martin Farach, Michiel Noordewier, Serap Savari, Larry Shepp, Abraham Wyner, and Jacob Ziv. On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[9] David Galas, Mark Eggert, and Michael Waterman. Rigorous pattern-recognition methods for DNA sequences. *Journal of Molecular Biology*, (186):117–128, 1985.

[10] W. Gilbert and B. Muller-Hill. The *lac* operator in DNA. *Proceedings of the National Academy of Science*, 58:2415–2421, 1967.

[11] M. R. Green. *Annu. Rev. Cell Biol.*, 7:559, 1991.

[12] C. Guthrie. *Science*, 253:252, 1991.

[13] Calvin B. Harley and Robert P. Reynolds. Analysis of e. coli promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.

[14] R. Harr, M. Haggstrom, and P. Gustafsson. Search algorithms for pattern match analysis of nucleic acid sequences. *Nucleic Acids Research*, 11:2943–2957, 1983.

[15] H. Hirsh and M. Noordewier. Using background knowledge to improve inductive learning. *IEEE Expert*, 9(5):3–6, 1994.

[16] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York, 1993.

[17] A.V. Lukashin, V.V. Anshelevich, B.R. Amirikyan, A.I. Gragerov, and M.D. Frank-Kamenetskii. Neural network models of promoter recognition. *Journal of Biomolecular Structure and Dynamics*, 6:1123–1133, 1989.

[18] T. Maniatis and R. Reed. *Nature*, 325:673, 1987.

sequence can be of great value. We feel that compression-based induction (CBI) fills a need for a simple, general technique for learning in these domains, in much the same way that decision-tree induction fits domains in which the identification of fixed features is relatively simple and ordering information is irrelevant.

The use of such methods requires some care. For example, our earliest attempts applied the **zdiff** algorithm (described below) to the data used in [15]. However, it was discovered that there was a flaw — the estimated true error rate was less than 0.1%! Analysis revealed that the cause of this error was a bias in the negative training data. These data had been created from overlapping windows into a single long sequence of *E. coli* DNA which was known not to bind promoter. The result was that if a test sequence contained a long substring which matched a substring in at least one negative training sequence, **zdiff** could unambiguously identify the test sequence as negative. This result pointed out an advantage of the CBI technique, in that CBI was able to exploit a flaw in the data which other techniques hadn't ever noticed. However, since the results were so strong because of a flaw in the data, new data were required to properly refine and test the technique.

This paper has described on-going work, and it is continuing in a number of directions. First, we are investigating the reasons underlying the differential classification success between promoters and splice junctions. Secondly, the compression methods that we used are incremental, and it is important to understand how the convergence rates of the underlying compression techniques affect the results of CBI. (*cf* [8])

Finally, there are other efforts that bear many similarities to this work. Textual substitution methods have been used to characterize DNA in [21, 20]. Hidden Markov chain models have been to characterize DNA in [26, 5]. Complexity estimates based upon repetition vectors have been used to estimate entropy to characterize exons and introns [27] and protein residue clusters [38]. It is important to improve our understanding of the relationship between these efforts and our own.

Acknowledgements

We thank Peter Yianilos for helpful discussions.

References

[1] Timothy Bell, John Cleary, and Ian Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[2] O. G. Berg and P. H. von Hippel. Selection of DNA binding sites by regulatory proteins. *Journal of Molecular Biology*, 193:723–750, 1987.

[3] Sϕren Brunak and Jacob Engelbrecht. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, (220):49–65, 1991.

Table 2: Estimated true error rates for splice-junction data

category	algorithm	error rate (%)
baseline	C4.5	11.7
compression	zdiff	40.2
techniques	tryeach-hamming	22.1
ordering	tryeach-near!	22.8
	tryeach!x	27.5
	tryeach!req0.2	23.5
non-compressive	tryeach!nbias	34.5
restrictions	tryeach!rbias	18.9
	tryeach!bbias	18.6
	tryeach!fstar	23.8

be accurately removed to produce a translatable message. This message splicing occurs by a transesterification reaction in which the 5' and 3' splice sites are cleaved with the formation of a lariat intermediate [11, 12, 18]. Our understanding of the splicing process is guided in part by the observation that sequences surrounding the 5', 3' and branch sites are conserved in evolution, and resemble the various snRNA sequences thought to be involved. Such sequence conservation serves as the basis both for proposed mechanisms [35] as well as evolutionary inference [31]. The most successful efforts to date to create splice junction classifiers have employed artificial neural networks [3].

Our data set was a selection of 719 Splice-junction donors from the STATLOG DNA dataset, derived from Genbank 64.1 primate data, using nominal (A,C,G,T) rather than numeric attributes. This dataset is substantially the same used in [23]. Because the positive data were centered on the splice site itself, all had an easily-recognized feature (a GT pair at positions 25–26). Therefore, a negative set was created from the non-donor, non-acceptor data included in the same DNA dataset, with an equal number of sequences, with each one having a spurious GT pair at positions 25–26.

The results of applying various CBI classification algorithms to the splice-junction donor recognition task are listed in Table 2. Error rates were again estimated using 10-way cross-validation. In this case results were inferior to C4.5.

5 Concluding Remarks

The need to recognize and classify sequences appears in a host of interesting applications, among them: speech recognition, generation of summaries of textual documents, protein folding prediction, and plan recognition, as well as the classification of DNA sequences. In these domains, it is difficult to tell where the interesting part of the sequence begins and ends; key subsequences may vary somewhat among examples; and ordering information within a

frequencies [6, 17]. DNA, however, is known to be highly non-random [7] — therefore negative training examples were derived by selecting contiguous subsequences from randomly selected and non-overlapping coding sequences in the Genbank *E. coli* collection, specifically coding regions of ECORGNB, ECORPLRPM, ECORPOA, ECORPSOP, ECORPSRPO, ECORPST, ECORPSTA, ECORPSU, ECORRFX, ECORRNH, and their complements.

The 291 promoters were obtained by trimming all promoter sequences to a single length (98bp), deleting nucleotides equally from both ends. Promoter sequences shorter than 98bp were discarded. The negative data were selected to be equal in length and number to the promoter data.

The results of applying various CBI classification algorithms to the promoter recognition task are listed in Table 1. Error rates were estimated using 10-way cross-validation. All methods but two resulted in superior performance to the baseline decision-tree method C4.5.

Table 1: Estimated true error rates for promoter data

category	algorithm	error rate (%)
baseline	C4.5	38.2
compression techniques	zdiff	30.5
	basic-cbi	34.0
	full-cbi	26.0
	tryeach	15.1
	tryeach-hamming	16.9
ordering information	tryeach-near	13.0
	tryeach-near!	10.8
non-compressive restrictions	tryeach!pos	48.0
	tryeach!noneg	46.0
	tryeach!both	16.2
	tryeach!x	10.1
	tryeach!req0.2	16.9
	tryeach!req0.5	11.5
	tryeach!nbias	30.2
	tryeach!rbias	10.8
	tryeach!bbias	10.5
	tryeach!fstar	11.2

4.2 Splice-Junction Site Recognition

The majority of eukaryotic genes display a complex structure in which sequences which code for protein (exons) are interrupted by intervening, non-coding sequences (introns). Initial transcription of these genes results in a pre-message RNA molecule, from which introns must

that of learning classifiers for recognizing non-bacterial splice-junction sites in protein coding regions of DNA, we present results for a subset of these algorithms.

4.1 Promoter Recognition

The promoters of the bacteria *Escherichia coli* constitute the most extensively studied single class of DNA regulatory sequences. Different approaches have been employed to define the consensus sequence of various DNA binding sites, including promoters, and to then use this consensus to locate members of that sequence family. These include using frequency ratios [14], consensus frequency tables [22], information theory [29], analogies to statistical mechanics [2], and methods that combine these [34, 37, 24, 25].

Inspired by such consensus specifications, a number of algorithms have been proposed for the recognition of promoters [28, 9, 22]; most of which rely on specific agreement with conserved regions within the -10 and -35 regions of promoter sequences (numbering refers to the approximate position upstream of the first transcribed base). These consensus sequences are defined by both high conservation and the fact that promoter mutations and polymerase contacts cluster in these regions.

Unfortunately, O'Neill [24] has noted that the average *E. coli* bacterial or phage promoter matches only 3.9 bases of the -35 consensus (TTGACA) and 4.2 bases of the -10 consensus (TATAAT), whereas a sequence of at least 11 bases is nominally required for unique identification in a random sequence the size of the *E. coli* genome [10]. Even allowing for the fact that this criterion may be too stringent (the genome is not random DNA), the observed specification of 8 out of 12 bases is relatively quite degenerate. Were promoters constituted solely by the -35 and -10 sequence at this level of match stringency, they would occur in one or the other direction approximately every 200 bases in a random sequence [7] — a much higher frequency than is found. As a result, most of the analysis techniques referred to above generate a superset of candidate promoters, then use a series of *ad hoc filters* to reduce the number of false positives (for an example, see [25]).

A consensus sequence, therefore, cannot automatically be used as a pattern recognizer for genetic sequences. In fact, if the 12 bases of the "perfect" Hawley consensus is used as a screen, no promoters are matched in a recent compilation [13].

For our first task, we chose to distinguish bacterial promoter sequences from non-promoter sequences. We used 291 sample promoters and 291 nonpromoter sequences. The 291 sample promoters were obtained from a compilation produced by Harley and Reynolds [13], and were all *E. coli* promoters for which transcriptional start points had been determined by biochemical or genetic means. We note in passing that this is a more complete set than represented in other recent studies [6, 17], which use an older and more restricted set (typically only bacterial promoters, or only the pre-aligned -10 and -35 regions). This set thus includes transposon and plasmid promoters, as well as promoters created by fusion or mutation. An initial concern of ours was the construction of "negative" examples (sequences which contained no promoters). Most studies randomly permute sequences in an effort to derive examples which do not meet consensus criteria, but nonetheless retain the correct nucleotide

global offsets in the range $[o_{min}, o_{max}]$ were tried, where o_{min} and o_{max} were the most negative and most positive offsets, respectively. The global offset yielding the best compression was reported as $I(t|s_j)$.

3.3 Non-compressive Techniques for Restricting Compression

The next group of algorithms explored the distinction between S^+ and S^-. While there were expected to be features specific to S^-, they were expected to be rarer and more diffuse than those specific to S^+, since all members of S^+ come from a narrowly defined class of DNA, while members of S^- could come from a broader range. **Tryeach!pos** addressed this issue by requiring that any match to a sequence in S^+ match at least one other sequence in S^+. **Tryeach!noneg** required that any match to a sequence in S^+ match no sequence in S^-. Finally, as a test on the behavior of **tryeach!pos**, **tryeach!both** required that any match with S^+ match at least one other sequence in S^+, and any match with S^- match at least one other sequence in S^-.

Later research focussed on techniques for extracting more information from the training data, rather than adding biological information to the compression technique. **Tryeach!x** added to **tryeach-near!** a requirement that any match between t and a member of S^+ to be used for textual substitution match more members of S^+ than S^-, and conversely, any match between t and a member of S^- to be used for textual substitution match more members of S^- than S^+. **Tryeach!req0.2** required any match between t and a member of S to match at least 20% of all members of S, and **tryeach!req0.5** required any match between t and a member of S to match at least half.

Finally, we examined the idea of a "good" match. Implicitly in all of the above methods, longer matches are better than shorter, other things being equal. We tested the effect of relaxing this assumption. **Tryeach!nbias** extended **tryeach-near!** by replacing the pointers of form (i, l, o) with pointers of form (i, p, o), where p was a cumulative probability associated with the match, such that the length of p using arithmetic coding [1] would increase for matches found in few sequences and decrease for matches found in many. Similarly, **tryeach!rbias** used p such that the length of p increased for more common matches, and **tryeach!bbias** maximized the length of p for both rare and common matches. **Tryeach!fstar** used the F* statistic described in [4] to increase p as $|F*| \rightarrow 1$; that is, p was small when the match was much more common or much rarer than would be expected for a sequence of its length and composition.

4 Experiment Design and Results

This paper describes our on-going efforts at using compression-based induction methods on DNA sequence identification tasks. To explore the success of our range of CBI methods we have selected two DNA sequence identification problems. For the first, learning classifiers for distinguishing bacterial promoters from non-promoters, we explore the results of using all the algorithms described in the previous section. For the second, more recent problem,

each s_j, using pointers which were pairs of the form (i, l), with i and l defined as in **basic-cbi**. The overall compressed length of t was calculated as $I(t|S) = 1/n \sum_{j=1}^{n} I(t|s_j)$. This method is related to a method of calculating similarity seen, for example, in [30].

It was felt likely that the same biologically interesting feature could vary significantly from one sequence to another, as for example the various features in prokaryotic promoters [9]. To account for this variance, it was decided to explore replacing exact text compression with a lossy technique. In **tryeach-hamming** each pointer pair of **tryeach** was replaced with a triple of the form (i, l, h), where h was the Hamming distance between the substring in the test sequence t and the matching substring in s_j. The Hamming distance was expressed using a prefix code [16, p. 12], so that smaller Hamming distances were encoded as shorter codes.

3.2 Encoding Feature-Ordering Information in Pointers

The above methods use no biological information other than the fact that DNA sequences are, in fact, sequences of characters drawn from a finite alphabet. However, the best learning algorithms currently available for DNA sequence classification make use of biological information, either in the form of additional learning features, or as constraints on the types of models the learning algorithm may consider. Therefore, we incorporated into several algorithms general biologically-derived constraints on feature ordering.

It is well-known that certain biological features are only of interest when they appear either near another feature. For example, a promoter site normally occurs within 16–18 bases upstream of a Pribnow box. A learning algorithm which can recognize such constraints can be expected to produce better classifiers than one which cannot. Therefore, we propose to extend text compression to recognize ordering and distance constraints between commonly occurring subsequences.

Therefore, several compression programs were written to take advantage of simple biological information of this sort without overly biasing the algorithm toward any particular biological domain.

While all features could not be aligned for reasons already discussed, it was possible to choose a point at which at least some feature was aligned. In this case, the other features could be expected to be misaligned by a relatively small amount. Therefore, **tryeach-near** preferred matches between a test substring and a training substring which were at approximately the same position. This was accomplished by replacing the pointer pairs in **tryeach** with triples of the form (i, l, o), where o was the absolute value of the offset between test and training substring, encoded using a prefix code.

The method of **tryeach-near** was extended in **tryeach-near!**. Here, the goal was to prefer hypotheses in which several features are displaced by similar amounts (*e.g.* because one feature is missing, or is longer than usual, or because the entire sequence is misaligned). The method was to bias the algorithm to prefer misalignments in which most matches were misaligned by similar amounts, by subtracting from each offset a global offset, which was then encoded using a prefix code and added to the beginning of the compressed sequence. All

3 Description of Algorithms

We analyzed several variants of compression-based induction in order to find methods which converge rapidly on a good induction model on a range of biological data. In all cases, two sets of training examples, S^+ and S^-, were used to generate two separate models. These models were then applied to each test example t individually, yielding two compressed sequences with lengths $I(t|S^+)$ and $I(t|S^-)$. The class of t was then taken as the same as S^+ if $I(t|S^+) < I(t|S^-)$, or as S^- otherwise.

Our primary direction of research has been the application of domain knowledge while still generating algorithms general enough to be useful across a wide range of DNA sequence classification problems. The means of entry for domain knowledge shifted from the way compression is used in CBI, to the encoding of features of interest within the compression method, and finally to the merging of compression with non-compressive techniques for separating meaningful features from spurious ones.

3.1 Compression Variations

Our initial method, **zdiff**, is an application the well-known LZ78 algorithm [39]. The training examples for each category are simply concatenated (separated by line feeds), and compressed, yielding $I(S)$, where S is either S^+ or S^-. The concatenated training examples are then concatenated with each test example individually, and compressed, yielding $I(t, S)$. The compressed length of the test example given the training set is then calculated as $I(t|S) = I(t, S) - I(S)$.

All subsequent methods use a textual substitution method for compression [36], rather than the Ziv-Lempel dictionary method. This method was determined to be simpler to modify as we explored different methods of representing various types of domain knowledge. The basic method, **basic-cbi** finds $I(t|S)$ by encoding each character in t by one of five possible values: A, C, G, T, or an escape indicating that the next bits constitute a pointer into a corpus of training sequences S. The pointers are triples of the form (n, i, l), indicating the index number of the sequence in S, the index of the first character of a substring of this sequence matching the substring of t starting at the current position of t, and the length of the match. A substring is encoded by a pointer if the length of a pointer, in bits, is less than $l(log_2 5)$.

One limitation of **basic-cbi** was that there was a ceiling to the score of $I(t|S)$, namely $I(t|S) \leq |t| \log_2 5$. To explore the effect of this ceiling, we removed it by forcing every character to be encoded by a pointer, even if the resulting "compressed" sequence became larger than the uncompressed sequence due to the size of the pointers. The algorithm with this change is labelled **full-cbi**.

Because we found that **basic-cbi** did not take sufficient account of the frequency of commonly occurring substrings, we tried a variant of the textual substitution method in **tryeach**. **Tryeach** examines each set S of training examples as being composed of n separate sequences labelled s_1 through s_n. The compressed length of t was calculated separately for

the rationale behind our approaches. Section 3 describes the algorithms themselves. We have employed compression algorithms to examine two well-studied sequence classification problems, *E. coli* promoter recognition, and eukaryotic splice junction detection. The results of our experiments are presented in 4, followed with discussions of related work and future directions.

2 Text Compression and Compression-Based Induction

The application of text compression techniques to DNA sequence classification has a reasonable theoretical basis. Text compression is a practical application of the study of the entropy of stochastic sequences. One can generate a dictionary of phrases which may appear in any sequence. In this case, we may generate sequences at random by assigning a probability to each phrase, then choosing phrases at random and concatenating them until we have generated a sequence of desired length. The probability that a sequence could have been generated by choosing random phrases from our dictionary may be understood as $Pr(S|d)$, conditional probability of the sequence, given the dictionary [1]. The entropy of the sequence is $E = -log_2 Pr(s|d)$, and can be understood as the smallest number of bits to which the sequence may be compressed such that the compressed sequence and the dictionary together are sufficient to reconstitute the original sequence [32]. In place of a dictionary, any stochastic method for generating appropriate sequences may suffice, including Markov models [1], stochastic choice among an infinite set of algorithms [16], or a corpus of sequences [19]. Therefore, the number of bits to which a model can compress a sequence may be used to calculate the probability that the sequence could have been generated from the model.

The degree to which the dictionary or model used to compress one string is useful for compressing another can be analyzed as a simple function of the probability that the two strings were generated by the same underlying process. As applied to DNA sequences, if the model generated during compression of a set of training sequences of a given type compresses a test sequence well, then, by a maximum likelihood argument, the test sequence is likely to be of the same type.

Our proposed method for DNA sequence classification is therefore to build a compression model or dictionary for each of several classes of DNA having biological value, such as "introns" and "exons", and then classify each new sequence by determining which of the models better compresses the new sequence. Since we are inducing a model using text compression, we have termed our method "compression-based induction" (CBI).

1 Introduction

DNA sequence classification has the appearance of a good problem domain for well-understood machine learning techniques. The problem is typically stated as distinguishing two or more classes of sequences, given a corpus of labelled training examples. Many standard machine learning algorithms have been applied to problems within this domain, including C4.5 [15] and artificial neural networks [3, 33]. The majority of such work treats individual nucleotides positions as attributes and applies algorithms that attempt to construct a classifier that considers each attribute individually. This approach borrows from the study of evolutionary conservation for biological sequences, where sequences that serve to control gene expression are commonly examined by the construction of a "consensus" sequence — a process by which many sequences are aligned and a composite subsequence is created by taking the majority base at each position. Such a composite is used both to support inferences of mechanism, and as a discriminant tool for the recognition of sequence boundaries in the raw sequence data now being produced by the various genome projects.

Although such an approach has proven successful for diagnostic problems, such as medical diagnosis and electrical fault diagnosis, it is less appropriate for DNA sequence classification. In particular, in such work only feature values are important, and the ordering of features is irrelevant. For example, if a chart indicates that a patient has a temperature of 39 and an age of 77, it makes no sense for the physician to consider that the temperature is "before" the age. If the patient also has a blood alcohol level of 39 and a fasting glucose level of 77, this repetition of 39 and 77 should rightly be treated as no more than a coincidence.

However, for DNA sequence classification it is important to take into account the ordering of nucleotides and repetitions of subsequences; moreover, the absolute positions of nucleotides are of lesser importance and prone to error. In those cases where inductive learning has seen success DNA sequences are aligned to make it easier to treat nucleotides as independent features, but these alignments are often inexact and controversial, constituting a substantial source of error.

Our first approach to this problem proceeded from a simple intuition. If we took a corpus of DNA sequences, we could gain insight into the degree of similarity between a test sequence and the corpus by compressing the corpus with the test sequence appended, and subtracting the size of this compressed file from the size of the compressed corpus alone. We could classify a test sequence by following the above procedure with two different sample populations fo text, assigning the test sequence to the label of the population with which it compressed best.

This paper describes the use of text compression technology on the problem of DNA sequence classification. In text compression, common subsequences of characters are recognized so that they may be replaced by shorter codes when they are encountered again. The recognition of such common subsequences to allow effective compression depends on the relative ordering of sequences of text and repeated patterns within the text, rather than on the exact positions of such sequences. This paper describes our on-going work exploring several *compression-based induction* (CBI) methods to DNA classification. Section 2 explains

ABSTRACT

Inductive learning methods, such as neural networks and decision trees, have become a popular approach to developing DNA sequence identification tools. Such methods attempt to form models of a collection of training data that can be used to predict future data accurately. The common approach to using such methods on DNA sequence identification problems forms models that depend on the *absolute locations* of nucleotides and assume *independence* of consecutive nucleotide locations. This paper describes a new class of learning methods, called *compression-based induction* (CBI), that is geared towards sequence learning problems such as those that arise when learning DNA sequences. The central idea is to use text compression techniques on DNA sequences as the means for generalizing from sample sequences. The resulting methods form models that are based on the more important *relative locations* of nucleotides and on the *dependence* of consecutive locations. They also provide a suitable framework into which biological domain knowledge can be injected into the learning process. We present initial explorations of a range of CBI methods that demonstrate the potential of our methods for DNA sequence identification tasks.

DIMACS Technical Report 95-04
April 1995

DNA Sequence Classification Using Compression-Based Induction

by

David Loewenstern*, Haym Hirsh[1]*, Peter Yianilos†,
and Michiel Noordewier[2]*

*Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA

†NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA

[1]Permanent Member
[2]Permanent Member

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.