JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 2, Number 3, 1995
Mary Ann Liebert, Inc.
Pp. 473–485

Locating Protein Coding Regions in Human DNA Using a Decision Tree Algorithm

STEVEN SALZBERG

ABSTRACT

Genes in eukaryotic DNA cover hundreds or thousands of base pairs, while the regions of those genes that code for proteins may occupy only a small percentage of the sequence. Identifying the coding regions is of vital importance in understanding these genes. Many recent research efforts have studied computational methods for distinguishing between coding and noncoding regions, and several promising results have been reported. We describe here a new approach, using a machine learning system that builds decision trees from the data. This approach combines several coding measures to produce classifiers with consistently higher accuracies than previous methods, on DNA sequences ranging from 54 to 162 base pairs in length. The algorithm is very efficient, and it can easily be adapted to different sequence lengths. Our conclusion is that decision trees are a highly effective tool for identifying protein coding regions.

Key words: coding regions, decision trees, machine learning, exons

INTRODUCTION

The Human Genome Project and the many major sequencing efforts that fall under its demesne are generating ever-increasing amounts of DNA sequence data. Many of these data are raw, in the sense that the methods for generating the sequences do not indicate whether they code for proteins or not. As a consequence, the automatic and efficient recognition of genes in DNA sequences is now widely recognized as a central problem in computational molecular biology. Finding genes by laboratory procedures is costly and time-consuming, and any accurate procedure that could supplement or even supplant laboratory analysis would be extremely valuable. Human DNA sequence presents the most important challenge for automated gene identification, both because of its inherent interest and value and because many methods do not work as well on human DNA as they do on prokaryotic or simpler eukaryotic DNA (Fickett and Tung, 1992). Thus we have focused on human DNA for this study.

Over the past 12 years, researchers have identified a number of features of exonic DNA that appear to be useful in distinguishing between coding and noncoding regions (Fickett, 1982; Blaisdell, 1983; Hinds and Blake 1985; Lapedes *et al.*, 1989; Farber *et al.*, 1992; Uberbacher and Mural, 1991; Staden and McLachlan, 1982). These features include both statistical and information-theoretic measures, and in many cases are based on knowledge of the biology underlying DNA sequences and transcription processes. These features

Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218.

were summarized in a survey by Fickett and Tung (1992), who also developed several benchmark features and databases for future experiments on this problem.

Previous research on automatic identification of protein coding regions has considered methods such as linear discriminants (Fickett and Tung, 1992; Fickett, 1982) and neural networks (Farber *et al.,* 1992; Uberbacher and Mural, 1991). These systems have used measures such as codon frequencies, dicodon frequencies, fractal dimensions, repetitive hexamers, and other features to try to identify exons in relatively short DNA sequences. The standard experimental study considers a limited window (i.e., a subsequence) of a fixed length, for example 100 base pairs, and computes features based on that window alone. The goal is to identify the window as either all-coding or all-noncoding. We have followed this experimental method in our study, and we have used the benchmark human DNA databases developed by Fickett and Tung (1992).[1] The motivation for these benchmarks, which include sequences as short as 54 base pairs, is that classification of short sequences is by far the most difficult problem faced in this research area, but it is not so difficult as to be impossible.

Our approach uses a decision tree classifier as the primary algorithm, in contrast to previous studies that used linear discriminants and neural nets. Decision tree classifiers are a standard technique in the machine learning and statistics communities, and they have been shown to perform well on a wide range of tasks, including such diverse problems as breast cancer diagnosis, chess endgame analysis, speech understanding, and astronomical image understanding (Murthy *et al.,* 1994; Quinlan, 1986; Salzberg *et al.,* 1995). Decision trees are basically a series of tests organized in a tree-like structure, where each test is equivalent to a linear discriminant. This framework is inherently more powerful than a single linear discriminant (or a one-layer perceptron, which is equivalent to a linear discriminant). We experimented with two types of decision trees: those that test only a single feature at each node, and those that contain a linear combination of the features at each node, i.e., a hyperplane that separates the examples into two subsets.

The conclusions of our study are first, that decision trees are more accurate classifiers than any previously reported. Second, we found that decision trees are consistently better classifiers regardless of the length of the DNA sequence. This stands in contrast to the fact that previous techniques varied widely in their relative accuracy for different length DNA sequences. This advantage stems from the inherent ability of a decision tree algorithm to choose from among a set of features those that are best for a given task. The algorithm for building the trees is very efficient, and is embedded in a system called OC1 that we have made available over the Internet. Finally, we found that some of the trees were quite small, and as a result are both easy to use and to understand.

INDUCTION OF DECISION TREES

Decision tree algorithms have been described in many places, for example, ID3 and C4.5 (Quinlan, 1993), CART (Breiman *et al.,* 1984), and OC1 (Murthy *et al.,* 1993, 1994). Only the basic algorithm will be summarized here; the interested reader is directed to these sources for more details. All decision tree systems operate using the same underlying algorithm. The input is a set S of examples; in the current study, S consists of a set of nonoverlapping DNA subsequences. The algorithm is

1. Split the set S using a test on one or more features, $f_1, f_2, \ldots, f_d$.
2. Check the results of the split. If every partition is pure (all examples in the partition belong to the same class), then stop. Label each leaf node with the name of the class there.
3. Recursively split any partitions that are not pure.

The result of this algorithm is a decision tree with test nodes and leaf nodes. The leaf nodes contain class labels.

Decision trees vary primarily in how they choose a "split" of the data, and in how they prune the trees they produce. The data should be formatted as a set of features and a class label; for example, this study

[1]To obtain instructions on how to retrieve the benchmark data, as well as the discriminant vectors and source code to compute them, send two e-mail messages to bioserve@genome.lanl.gov. One message should contain only the text "cds_discriminant," and the other message should contain the text "cds_benchmark."

includes 21 coding measures (the features), such as dicodon frequencies, hexamer frequencies, and length of the longest open reading frame. Each of these features is a number computed on a fixed-length window of the DNA sequence. There are only two class labels: coding and noncoding.

Splitting rules

Given this numeric input, a decision tree algorithm must decide how to use the features to "split" the data. For example, if a feature is named "codon usage," one possible test might be codon usage < 2.45. We call this a univariate test because it uses only one feature (variable). With a univariate test, there are only $N - 1$ distinct tests to consider for N examples. These would occur at the midpoints between successive feature values. With N examples and D features, we have $D(N - 1)$ such tests to consider. Each test is scored by a **goodness** measure. The simplest such measure is the one that counts the number of features that would be misclassified by the test (Heath *et al.,* 1993). More sophisticated tests measure the entropy of the initial set and the subsets produced after splitting (Quinlan, 1993), or use statistical properties such as the probabilities of each class on either side of the split. (Note that a test always splits the data into two subsets.) The "twoing rule" (Breiman *et al.,* 1984) tries to create tests that maximize the purity of both subsets while at the same time splitting the data roughly in half. For definitions of these and other goodness measures, see Murthy *et al.* (1994).

Another, more complicated type of test is the linear discriminant, which was used by Fickett and Tung (1992). In the context of decision trees, this kind of test is called an *oblique* split. Instead of testing a single feature such as codon usage, these tests include a multivariate combination of the features $f_1, f_2, \ldots, f_d$, in the form

$$a_1 f_1 + a_2 f_2 + \cdots + a_d f_d \geq a_{d+1}$$

This test, while much more powerful than the simpler univariate test, is also more expensive to compute. However, recent work on randomized decision tree algorithms has resulted in very efficient methods for finding these oblique tests (Murthy *et al.,* 1994). Our OC1 system always finds the best axis-parallel split first, and then considers oblique splits and uses those if they work better. The system can be set to use axis-parallel splits only for even greater efficiency.

One interesting result that came out of the current study was that virtually all goodness measures are designed so as to maximize, in one form or another, the overall accuracy of the tree. Thus if one class is much larger than another, as is the case with Fickett and Tung's benchmark training sets, decision trees tend to optimize accuracy on the larger class. To correct this, we developed a new goodness measure called "average accuracy." (This measure is included with the source code.) Average accuracy measures the goodness of a split by simply computing the accuracy of each class for a given test, and then averaging the results across all classes. To use all the other goodness measures, though, we needed to produce training sets in which the sizes of the classes were roughly equal. From each of the benchmark data sets, therefore, we extracted a subset of the training data with equal class distributions, and used just the subset for building trees.

Pruning rules

Most decision tree systems provide some means for pruning the decision tree after it is built. The initial algorithm takes some training data and builds a complete tree, one that correctly classifies every example in the training set. When the data are noisy, this tree usually "overfits" the training data and performs poorly on additional data. A smaller tree often performs better, and offers the advantage of being simpler and faster to use.

Our decision tree system uses a pruning technique called cost complexity pruning (Breiman *et al.,* 1984). First it divides the training data into two sets, the training set (T) and the pruning set (P). After building a decision tree on T, it then produces a succession of smaller trees by looking at every nonleaf node in the tree, and measuring the cost complexity of that node. Cost complexity considers the number of examples that would be misclassified (the cost) if that node were made into a leaf, and it also measures the complexity of the subtree rooted at that node. These two are combined into the cost complexity measure; for details see Breiman *et al.* (1984). The node whose cost complexity is greatest is then deleted or "pruned," and the process continues. This continues until the tree is pruned down to a single node.

The system then examines the series of increasingly smaller trees and computes their accuracies on the pruning set P. In the simplest pruning model, called SE-0 pruning, it chooses the tree with the highest accuracy on the pruning set. The SE-1 pruning method computes the standard error (SE) of the accuracies of all the trees and then chooses the smallest tree whose accuracy is within one standard error of maximal. For this study we considered SE-0, SE-0.5, and SE-1 pruning.

DATA AND METHODS

The data used for this study are the human DNA data collected by Fickett and Tung (1992). All the sequences were taken from GenBank in May 1992. Fickett and Tung proposed these sequences as a benchmark for future studies of protein coding regions, and their benchmark data include exactly the same training and test sets used in their 1992 study. They have also provided the 21 different coding measures that they surveyed and compared.

The benchmark human data include three different data sets. For the first data set, nonoverlapping human DNA sequences of length 54 were extracted from all human sequences, with shorter pieces at the ends discarded. Every sequence was labeled according to whether it was entirely coding, entirely noncoding, or mixed, and the mixed sequences (i.e., overlapping the exon–intron boundaries) were discarded. The data set also included the reverse complement of every sequence. This means that one-half of the data is guaranteed to be from the "nonsense" strand of the DNA, which makes the problem of identifying coding regions somewhat harder. For the current study, we used the same division into training and test data as in the benchmark study (Fickett and Tung, 1992). The training set was used exclusively to construct a decision tree, and the tree was then used to classify the test set. In addition to the 54-base data set, we used data sets containing 108 and 162 bases. The sizes of these data sets are shown in Table 1, which gives the number of nonoverlapping "windows" in each set.

Note that no information about reading frames was used in this study. We were trying to solve the problem of finding coding regions in DNA about which nothing is known. Every window was either all-coding or all-noncoding, but the reading frame of each window was unknown. (Note that decision trees can easily be adapted to label windows containing mixed exons and introns, as we are planning to do in a follow-up study.) This choice of window lengths and experimental method follows that used by Fickett and Tung (1992), and the problem here is what they defined as "protein region coding." (Their other problem, phase-specific coding, uses the assumption that the window is aligned on an open reading frame.)

Tuning the parameters

The decision tree system used for all our experiments is called OC1, or Oblique Classifier 1. It has the ability to generate both oblique and univariate tests at every node, and it can use any one of seven different goodness measures, including information gain (Quinlan, 1986), the twoing rule (Breiman *et al.*, 1984), and average accuracy (defined above).

Decision trees, like many other machine learning methods (for example, neural network algorithms), have a number of parameters that can be adjusted to optimize their performance on any given problem.

TABLE 1. SIZES OF TRAINING AND TEST SETS FOR THE BENCHMARK DATA[a]

Data set	*Human 54*	*Human 108*	*Human 162*
Training set coding windows	20,456	7,086	3,512
Training set noncoding windows	125,132	58,118	36,502
Training set total	145,588	65,204	40,014
Test set coding windows	22,902	8,192	4,266
Test set noncoding windows	122,138	57,032	35,602
Test set total	145,040	65,224	39,868
Training + test sets	290,628	130,424	79,882

[a]A window is a subsequence of DNA of length 54, 108, or 162 bases. None of the windows overlapped within a given data set.

Any such technique that is run on a fixed test set, adjusted, and rerun repeatedly is inevitably going to show improvements. However, this is an unfair experimental method, because it does not model what would happen if you applied the algorithm to a new data set for which the class labels were unknown. When the test set is provided in advance, as with a benchmark, the system parameters must be tuned on the training set *only*.

Therefore, for the studies reported below, we first divided the training set into two subsets. One subset was used to build the tree, and the other was used to estimate its accuracy. We call this second subset the **tuning** set. After running a number of such experiments to find the best setting of parameters for OC1, we then ran OC1 on the full test set. For each result below, we report the parameter settings used to obtain OC1's trees. For example, on the 162-base DNA data, the training set contained 36,502 noncoding subsequences and just 3512 coding subsequences. From these data, we first removed 1000 coding subsequences and 10,000 noncoding subsequences to serve as a tuning set. Then we constructed OC1's training set using the remaining 2512 coding sequences and an equal number of noncoding sequences. This training set, containing 5024 DNA subsequences, was used to build all the decision trees for the Hum162 data. OC1 automatically used 10% of this data, or 502 examples, as a pruning set. After building and pruning the tree, the 11,000 examples in the tuning set were used to estimate the accuracy of the tree. Finally, the best trees were selected, and these were run against the benchmark test set. Not surprisingly, the accuracies on the tuning set were consistently much higher than those on the test set, but we report only the accuracies on the test set.

Coding measures

We used a selection of the 21 coding measures reviewed and summarized in Fickett and Tung (1992). These measures fall into several distinct groups, and within a group some measures are just variations of each other. Note that a single *measure* is not necessarily the same thing as a single feature. Typically, a coding measure is a vector of measurements on a DNA subsequence. For example, the **dicodon** measure (Farber *et al.*, 1992) is the list of the 4096 frequencies of every possible dicodon (six consecutive bases from the 4-letter alphabet), where the codons are aligned with the first base in the window. The **hexamer-1** and **hexamer-2** measures are identical to dicodon, except that they are offset by 1 and 2 bases, respectively, from the start of the window. Fickett and Tung used linear discriminant analysis to produce a linear separator computed on these 4096 frequencies (using the training set only, of course). In essence, this is a 4096-dimensional hyperplane. To convert the **dicodon** measure to a single number, the 4096 dicodon frequencies are computed on each window and plugged into the hyperplane equation. This gives a single number that becomes the dicodon *discriminant*. Note that this number measures a form of distance from the hyperplane, and is positive if and only if the example is above the hyperplane.

Some of the measures comprised fewer values. For example, the **Fourier** measure (Silverman and Linsker, 1986; Fickett and Tung, 1992) contains just eight values, corresponding to the autocorrelation function for bases separated by 2 through 9 positions. We experimented with using both the Fourier measure itself and the distinct eight values. One interesting result of our study was that we found that the Fourier coefficient corresponding to bases 3 apart was sufficient by itself to obtain results as good or better than those obtained using all eight values. When we gave all eight values to our decision tree system, it consistently just chose that one.

Brief definitions of all the measures we used are as follows. More complete definitions can be found in Fickett and Tung (1992). The dicodon, hexamer-1, and hexamer-2 measures are defined above. The **open reading frame** measure is simply the longest sequence of codons in the window that do not contain a stop codon. The **run** measure counts the number of repeats or "runs" of a single base or any set of bases from the set (a,c,g,t). Thus it includes the 14 nontrivial subsets of the four bases, and for each subset runs are counted separately. The **position asymmetry** measure contains four values. It counts for each of the four bases the frequency of the base in each of the three codon positions. Thus $f(b, i)$ is the frequency of base b in position i, and $\mu(b) = \sum_i f(b, i)/3$. Asymmetry is then defined by $\text{asymm}(b) = \sum_i [(f(b, i) - \mu(b)]^2$. The **codon usage** measure is simply the frequencies of the 64 possible codons in the test window. The **diamino acid usage** measure is the 441 values computed by translating the window into its amino acids (counting the stop codon as an additional amino acid) and counting all overlapping pairs of amino acids. Finally, the Fourier measure comes from Silverman and Linsker (1986), but was modified slightly by Fickett and Tung as follows. Let $E(x, y)$ be the equality predicate that has value 1 if $x = y$ and 0 otherwise. The nth Fourier coefficient for a window W of length $2M$ is then defined as

$$F(n) = \sum_p \sum_m [E(W_m, W_{m-p})] e^{\pi inp/M}$$

where W_m represents the mth base in the window. The **Fourier** measure is then just $F(2M/2)$, $F(2M/3)$, ..., $F(2M/9)$, which corresponds to the Fourier coefficients for periods 2 through 9.

Fickett and Tung recommended that future studies use just six of the measures they surveyed: dicodon, hexamer-1, hexamer-2, the open reading frame measure, the Fourier measure, and the run measure (Blaisdell, 1983; Fickett and Tung, 1992). For our study, we call this the **6-feature** set. We used the linear discriminant value for each of these measures, i.e., for those measures that comprise a vector of values, we used the single value produced by plugging the vector values into the hyperplane computed by Fickett and Tung. (These discriminant values, as well as the vectors, are provided with the benchmark data.) We wanted to consider a few other features based on the fact that they performed well in at least one of the benchmark studies: position asymmetry (Fickett, 1982), diamino acid usage, and codon usage (Staden and McLachlan, 1982), and we also wanted to use the separate components (rather than the discriminants) for the position asymmetry and Fourier measures. Thus we also created a set called **19-feature,** comprised of

1. Dicodon usage
2. Hexamer-1
3. Hexamer-2
4. Open reading frame
5. Diamino acid usage
6. Codon usage
7. Run

8–11. Position asymmetry for a, c, g, and t

12–19. Fourier coefficients for periods 2–9

Method of comparison

Our results are given in the next section along with the three best results for each sequence length as found by Fickett and Tung in their 1992 survey. Note that the measures that performed best varied with the size of the DNA sequence: the best measure for 54-bp DNA was the hexamer measure, while for 108-bp DNA it was position asymmetry, and for 162-bp DNA it was the Fourier measure.

For each data set, we report accuracies on the test set, broken up into accuracy on exons and introns. We also report average accuracy, which is just the average of these two numbers. This was the value reported by Fickett and Tung (1992). In addition to average accuracy, we report the more traditional "overall" accuracy measure, which is just the percent of correct predictions on the entire test set. The test set for each window length was significantly skewed, with exons comprising a very small percentage of the data. (Note that if we wanted to optimize overall accuracy, we would set our parameters differently, which would give us higher accuracy on the introns but lower accuracy on the relatively small class of exons.) We should emphasize again that all parameter adjustments to the OC1 algorithm were made on the training set, as described above. Only after the two or three best trees were selected did we run them on the test set.

EXPERIMENTAL RESULTS AND DISCUSSION

54-bp Human DNA

In this section we report the results on using decision trees for Fickett and Tung's Hum54 dataset. This dataset contains 290,628 examples, where each example is a window 54 base pairs long from a human DNA sequence. The windows do not overlap. For simplicity in the discussion, the term "exon" will denote a window (or subsequence) that is entirely contained within an exon, and "intron" respectively will denote a window that is all intron or noncoding.

The training set contains 20,456 exons and 125,100 introns. We wanted to create a balanced training set, and also to reserve a portion of the training set for tuning the OC1 decision tree system. So we first removed 15,000 exons and 15,000 introns to be our training set, and used the remaining 5456 exons

TABLE 2. ACCURACIES FOR LENGTH 54 HUMAN DNA SEQUENCES[a]

Algorithm/coding measure	*Coding (%)*	*Noncoding (%)*	*Average (%)*	*Overall (%)*	*Correlation coefficient*
OC1 (Tree 54-A)	72.4	74.2	73.3	73.9	0.361
OC1 (Tree 54-B)	**73.0**	72.8	72.9	72.8	0.352
OC1 (Tree 54-C)	66.3	**80.8**	**73.6**	**78.5**	**0.389**
Hexamer	71.7	69.4	70.5	69.8	0.310
Position asymmetry	69.6	70.9	70.2	70.7	0.309
Dicodon usage	70.7	69.7	70.2	69.9	0.306

[a]The three best measures from the original Fickett and Tung study are included for comparison. The highest accuracy in each column appears in boldface. The correlation coefficient is computed using the definition of Mathews (1975).

plus another 15,000 introns to be the "tuning" set. Thus our training set has 30,000 sequences and our tuning set has 20,456 sequences. After tuning the parameters of OC1 we used the tree with the highest average accuracy to classify the benchmark test set. We report results for the best tree on the parameter sets 6-feature and 19-feature. We also include in our tables the three best results using linear discriminant analysis on the individual coding measures (Fickett and Tung, 1992).

Our results for the Hum54 data are summarized in Table 2. Trees A and B were created using identical parameters for OC1. Both used SE-0 pruning, both used univariate tests exclusively, and both used the "average accuracy" goodness measure. The only difference was that Tree A was created using the 6-feature set and Tree B was created using the 19-feature set. Although Tree A has slightly higher accuracy, it is much more complex: it contains 130 leaf nodes versus just 20 leaf nodes for Tree B. Tree B uses only eight of the 19 features that OC1 had available. Because of its simplicity, this tree is reproduced in the appendix (Fig. 4).

Tree C is another small tree (17 test nodes) that used a different goodness measure, information gain (Quinlan, 1986), and the 6-feature set. It is interesting in that it has the highest average and overall accuracies, but it obtained this at the expense of the lowest accuracy on exons. Because of the skewed accuracies, we feel that such a tree is probably not ideal for this problem.

We also let OC1 construct numerous oblique trees. These require much more CPU time, but in our experiments on the tuning data they did not produce more accurate trees. Therefore we used only trees with univariate tests for the results reported in Table 2.

These trees are substantially better than any of the coding measures with a linear discriminant. The improvement in average accuracy for this data set is 3.1%, and for overall accuracy it was as high as 7.8%. It would be easy to increase the overall accuracy even further, because the test set contains over 84% introns, but this would significantly reduce the average accuracy. The correlation coefficients (Mathews, 1975) shown in the table show an even more striking difference between the decision trees and the coding measures.

108-bp Human DNA

In this section we report the results on Fickett and Tung's Hum108 dataset, which contains nonoverlapping windows 108 bases long. The benchmark training set contains 7086 exons and 58,118 introns. From this data, we extracted a balanced set containing 6000 exons and 6000 introns to train OC1. We used the remaining 1086 exons plus 10,000 of the remaining introns as a tuning set. As before, once we finished tuning, we used the best trees on the benchmark test set.

Our results for the Hum108 test data are summarized in Table 3. All the trees in this experiment used the average accuracy goodness measure and the 6-feature set of coding measures.

Trees 108-D and 108-E were both oblique trees, with the only difference being that Tree E was unpruned and contained 47 tests, while Tree D was a pruned version (using SE-0.5 pruning) of the same tree, with just 10 tests. Tree 108-D appears in the appendix (Fig. 3). While building these trees, OC1 used 20 randomized restarts at each node and 20 random jumps at every local minimum it encountered. (These are the two principal randomization parameters of OC1. Higher values result in longer run times, but usually

TABLE 3. ACCURACIES FOR LENGTH 108 HUMAN DNA SEQUENCES[a]

Algorithm/coding measure	*Coding* (%)	*Noncoding* (%)	*Average* (%)	*Overall* (%)	*Correlation coefficient*
OC1 (Tree 108-D)	74.6	**85.0**	**79.8**	**83.7**	**0.473**
OC1 (Tree 108-E)	76.3	83.0	79.7	82.2	0.457
OC1 (Tree 108-F)	**77.1**	82.0	79.5	81.4	0.449
Position asymmetry	75.3	78.0	76.6	77.6	0.391
Fourier	75.3	77.7	76.5	77.4	0.387
Hexamer	75.4	70.8	73.1	71.4	0.321

[a]The three best measures from the original Fickett and Tung study are included for comparison. The highest accuracy in each column appears in boldface.

produce better trees.) Execution time was about 45 min on a Silicon Graphics Indigo Extreme workstation, of which pruning takes only a few seconds. Note that while Tree D is much smaller, it is more accurate on everything except exons. The "average accuracy" measure does not give the complete picture, which is why we have presented more details in our tables. Tree F is an axis-parallel tree, unpruned, containing 68 tests. Note that although it performed slightly worse on the overall accuracy measures, it was the best tree for classifying exon sequences. Building this axis-parallel tree took OC1 less than 10 sec of CPU time.

Once again the improvement over previous results was about 3.0%. For overall accuracy, the improvement was greater, around 6%, and the Mathews correlation coefficient increased from 0.391 to 0.473. Fickett and Tung report that Uberbacher and Mural (1991) applied their GRAIL system in an informal test on the Hum108 data and were able to obtain 79% accuracy, which is just slightly less than that reported here.

162-bp Human DNA

In this section we report the results on using decision trees for the human DNA benchmark dataset containing windows of length 162. The training data were again just a subset of the complete training set, containing just 2512 exons and 2512 introns. The tuning set contained the remaining 1000 exons from the benchmark training set, plus another 10,000 introns from that set. Our results are summarized in Table 4. Tree G used the average accuracy goodness measure, SE-0 pruning, and the 6-feature set. This tree contained 16 univariate tests. Tree H used the twoing rule as its goodness measure, SE-1 pruning, and the 19-feature set. SE-1 pruning often produces very small trees, and in this case the tree size was just 5 tests! This tree appears graphically in Figure 1. Although 19 features were available, only four were used by OC1 here: hexamer-1, hexamer-2, Diamino acid usage, and the Fourier autocorrelation of positions 3 apart (one of the 8 coefficients used in the Fourier measure). Given the nature of the 3-letter code and the unequal usage of the codons within exonic DNA, one would expect that positions spaced 3 apart would be most useful for this task. The choice made here by OC1 confirms this intuition.

As with the shorter sequences, the decision trees created by OC1 are more accurate both in terms of average accuracy and the overall accuracy measure, The second tree, which uses five simple univariate

TABLE 4. ACCURACIES FOR OC1 FOR LENGTH 162 HUMAN DNA SEQUENCES[a]

Algorithm/coding measure	*Coding* (%)	*Noncoding* (%)	*Average* (%)	*Overall* (%)	*Correlation coefficient*
OC1 (Tree 162-G)	**81.0**	84.6	**82.8**	84.2	0.487
OC1 (Tree 162-H)	77.7	**87.3**	82.5	**86.3**	**0.535**
Fourier	79.2	82.3	80.8	82.0	0.443
Position asymmetry	79.2	82.0	80.6	81.7	0.440
Autocorrelation	79.5	74.5	77.0	75.0	0.360

[a]The three best measures from the original Fickett and Tung study are included for comparison. The highest accuracy in each column appears in boldface.

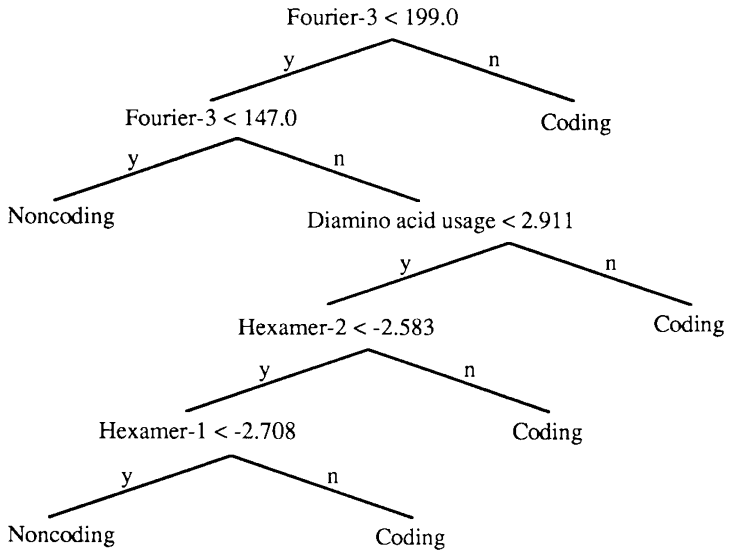

FIG. 1. Tree 162-H, which uses just four of 19 available features and contains just five test nodes.

tests, does remarkably well, although the accuracies are a bit more skewed in favor of introns than for the first tree. The improvement in average accuracy for these data is 2.0%, and for overall accuracy 4.3%.

Comparison to coding measures

Figure 2 shows, for all three subsequence lengths, how the best OC1 decision tree compares to the four best coding measures from Fickett and Tung's survey (Fickett and Tung, 1992). These measures are position asymmetry, Fourier, dicodon usage, and hexamer. As the figure makes clear, no single measure outperformed the others on all three data sets. All of the measures are comparable for the Hum54 data, while position asymmetry and Fourier seem to perform somewhat better on the longer DNA sequences. However, OC1 outperforms them all for all three sequence lengths. A comparison in terms of overall accuracy would reveal even more striking differences between OC1 and the individual coding measures. It is not clear why the coding measures themselves display such variation in performance with different sequence lengths; one explanation is that the measures of periodicity tend to perform best on the longer subsequences because the patterns of periodicity are too weak in the very short 54-bp windows.

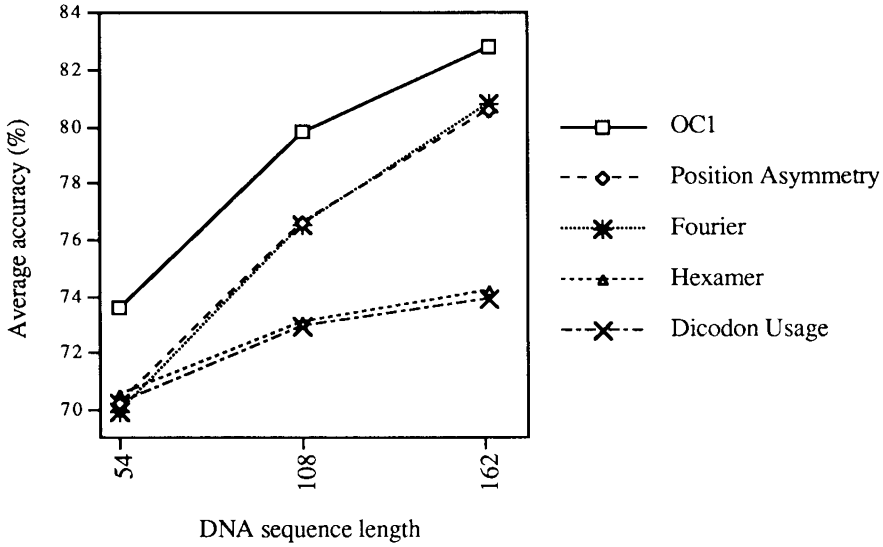

FIG. 2. A comparison of the average accuracy obtained by OC1 decision trees and by the four best coding measures.

CONCLUSION

The decision trees generated in this study were consistently more accurate at identifying protein coding regions on the benchmark data than the measures previously reported. One advantage they have over techniques such as linear discriminant analysis is that they do much of the work of feature selection automatically: the user can feed the system a large number of features, including irrelevant ones, and the decision tree algorithm will use only a subset in building the tree. If one or more features are redundant or interdependent, then once a feature is used at one place in the tree, the dependent features will not be used at lower nodes, because they will not provide further discriminatory power. The use of a series of tests is also likely to be more powerful than only a single test, such as that represented by a linear discriminant. For those who wish to use decision trees for similar experiments, we have made the OC1 source code available over the Internet, as described below.

Fickett and Tung (1992) provided an invaluable service by collecting and summarizing the various coding measures that have been studied over the past decade of research on detecting protein coding regions in DNA. They also suggested that six of the 21 measures were sufficient for any future attempts to distinguish exons from introns. This study confirms in part their findings, and also suggests some further conclusions.

First, it appears that fewer than six measures may be sufficient for performing the task, as some of our best decision trees used just four or five measures. Second, one of the most powerful measures they found was the Fourier coefficient. However, their study used all eight coefficients, corresponding to the autocorrelation functions for periods 2 through 9. By providing these eight values separately to OC1, we found that the only one necessary was the coefficient corresponding to period 3. This makes good sense in light of the fact that the genetic code uses 3 bases per codon; one would expect any periodicity or autocorrelation to show up most strongly in bases that are 3 apart. Using just a single coefficient is also much simpler than using a hyperplane defined by all eight coefficients.

Finally, we recognize that the results provided here may not represent the best values possible for recognizing exons and introns in eukaryotic DNA. Other coding measures may be uncovered that extract more information from the sequences, and larger data sets may also help to improve the accuracies. But within the constraints of this set of benchmarks—the coding measures and the data sets—we feel that substantial improvements over the accuracies given here will be very difficult to obtain. On the other hand, the numbers here represent the accuracies for a classifier that looks only at a very short DNA sequence. We are now working on embedding a decision tree in a coding region identifier system, similar to that described by Uberbacher and Mural (1991). The basic architecture is very simple: a DNA sequence will slide through the system, shifting one base at a time, and the system will identify the window centered on each base as either exon or intron. The base itself will then be labeled accordingly. Small decision trees such as those created in this study will operate extremely fast in such a framework, and should be highly accurate classifiers.

Source code

The code for OC1, including sources and documentation, is available via ftp or the World Wide Web. For those using ftp, the address is ftp.cs.jhu.edu. Connect to the directory pub/oc1 and transfer the files there. For those on the World Wide Web, connect to http://www.cs.jhu.edu/labs/ai/home.html and select the OC1 decision tree system listed there. These directories also contain a copy of the journal paper describing OC1 (Murthy *et al.* 1994).

APPENDIX

Decision trees created during this study

In addition to the tree shown pictorially in Figure 1, one tree for each of the other two data sets is shown here. These trees should be interpreted as follows: each node of a tree contains an equation, which is a test. If an example (a subsequence of DNA) passes the test, it is passed down to the left child node. Otherwise it goes to the right. Leaf nodes have one of the two labels, Coding or Noncoding.

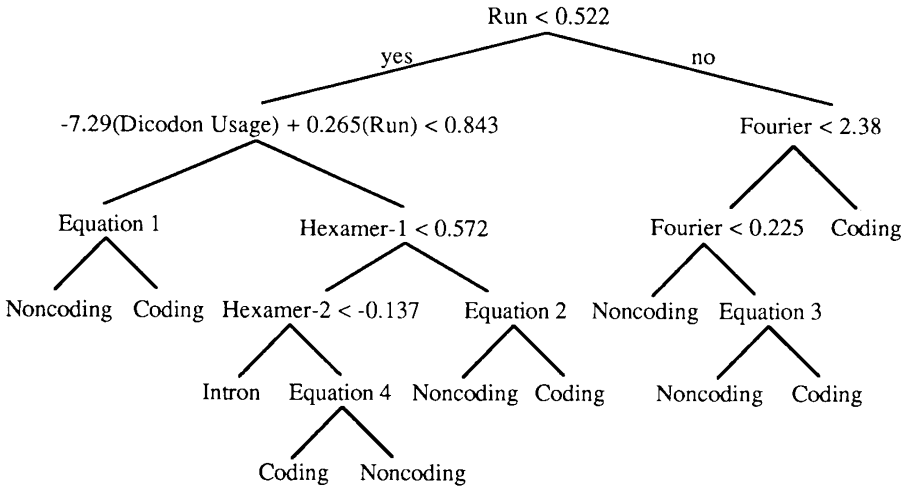

FIG. 3.

Tree for classifying human 108-bp DNA

This is the oblique tree described as Tree 108-D for the Hum108 data. Note that OC1 considers both univariate and multivariate tests when building oblique trees, so this tree has both kinds of tests at its nodes. For display purposes, the multivariate equations are written out fully below but abbreviated within the tree.

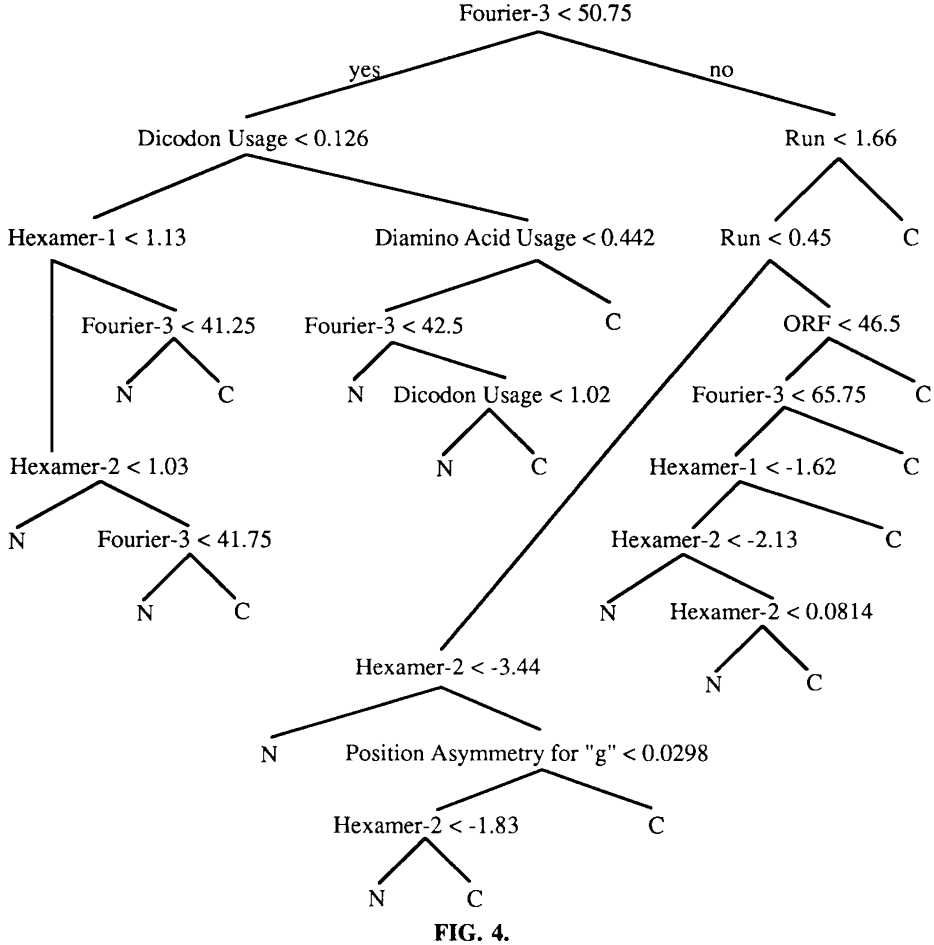

FIG. 4.

Equation 1: 7.27(Dicodon Usage) − 0.812(Hexamer-1) − 2.46(Hexamer-2) + 0.103(ORF) − 6.1(Fourier) + 71.1(Run) < 23.6

Equation 2: −1.69(Dicodon Usage) + 4.44(Hexamer-1) − 0.987(Hexamer-2) + 0.172(ORF) − 0.472(Fourier) + 2.53(Run) < 31.9

Equation 3: 0.101(Dicodon Usage) − 0.114(Hexamer-1) − 0.0166(Hexamer-2) + 0.00817(ORF) − 0.0249(Fourier) + 10.013(Run) < 7.91

Equation 4: 0.61(Dicodon Usage) + 0.116(Hexamer-1) − 9.45(Hexamer-2) − 0.191(ORF) + 8.77(Fourier) − 13.1(Run) < 6.68

Tree for classifying human 54-bp DNA

This is Tree 54-B, which uses just eight of the available 19 features. Those features used are dicodon usage, hexamer-1, hexamer-2, open reading frame (ORF), diamino acid usage, run, position asymmetry for "g," and the Fourier coefficient for positions 3 apart. The tree contains just 20 leaf nodes and 19 test nodes. By convention, a "yes" to any test means to proceed down the left branch, and a "no" indicates the right branch. For space constraints, the labels "C" and "N" are used in place of the terms "coding" and "noncoding."

ACKNOWLEDGMENTS

Thanks to S.K. Murthy for help making modifications to the OC1 source code, and thanks to Kenneth Fasman, Simon Kasif, and Eric Brill for their helpful comments and suggestions. This research is based upon work supported by the National Science foundation under Grants IRI-9116843, IRI-9223591, and IRI-9220960.

REFERENCES

Blaisdell, B.E. 1983. A prevalent persistent global nonrandomness that distinguishes coding and non-coding eucaryotic nuclear DNA sequences. *J. Mol. Evol.* 19, 122–133.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. 1984. *Classification and Regression Trees.* Wadsworth International Group, Belmont, MA.

Farber, R., Lapedes, A., and Sirotkin, K. 1992. Determination of eukaryotic protein coding regions using neural networks and information theory. *J. Mol. Biol.* 226, 471–479.

Fickett, J. 1982. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Res.* 10, 5303–5318.

Fickett, J., and Tung, C.-S. 1992. Assessment of protein coding measures. *Nucleic Acids Res.* 20, 6441–6450.

Heath, D., Kasif, S., and Salzberg, S. 1993. Learning oblique decision trees, 1002–1007. *In Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Chambery, France. Morgan Kaufmann, San Mateo, CA.

Hinds, P.W., and Blake, R.D. 1985. Delineation of coding areas in DNA sequences through assignment of codon probabilities. *J. Biomol. Struct. Dyn.* 3, 543–549.

Lapedes, A., Barnes, C., Burks, C., Farber, R., and Sirotkin, K. 1989. Application of neural networks and other machine learning algorithms to DNA sequence analysis, 157–182. *In* Bell, G., and Marr, T., eds., *Computers and DNA, SFI Studies in the Sciences of Complexity,* vol. 7, Addison-Wesley, Reading, MA.

Mathews, B.W. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta* 405, 442–451.

Murthy, S.K., Kasif, S., Salzberg, S., and Beigel, R. 1993. OC1: Randomized induction of oblique decision trees, 322–327. *In Proceedings of the Eleventh National Conference on Artificial Intelligence.* MIT Press, Washington, D.C.

Murthy, S.K., Kasif, S., and Salzberg, S. 1994. A system for induction of oblique decision trees. *J. Artif. Intelligence Res.* 2, 1–32.

Quinlan, J.R. 1986. Induction of decision trees. *Mach. Learn.* 1, 81–106.

Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Salzberg, S., Chandar, R., Ford, H., Murthy, S.K., and White, R. 1995. Decision trees for automated identification of cosmic rays in Hubble Space Telescopy images. *Publications Astron. Soc. Pacific* 107, 1–10.

Silverman, B.D., and Linsker, R. 1986. A measure of DNA periodicity. *J. Theor. Biol.* 118, 295–300.

Staden, R., and McLachlan, A.D. 1982. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Res.* 10, 141–156.

Uberbacher, E., and Mural, R. 1991. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proc. Natl. Acad. Sci. U.S.A.* 88, 11261–11265.

Address reprint requests to:
Steven Salzberg
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218

salzberg@cs.jhu.edu

Received for publication January 6, 1995; accepted as revised February 27, 1995.