Appears in the Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences (1/93)

Learning to Predict Reading Frames in *E. coli* DNA Sequences

Mark W. Craven
craven@cs.wisc.edu

Jude W. Shavlik
shavlik@cs.wisc.edu

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

Two fundamental problems in analyzing DNA sequences are (1) locating the regions of a DNA sequence that encode proteins, and (2) determining the reading frame for each region. We investigate using artificial neural networks (ANNs) to find coding regions, determine reading frames, and detect frameshift errors in E. coli *DNA sequences. We describe our adaptation of the approach used by Uberbacher and Mural to identify coding regions in human DNA, and we compare the performance of ANNs to several conventional methods for predicting reading frames. Our experiments demonstrate that ANNs can outperform these conventional approaches.*

1 Introduction

As part of the Human Genome Initiative, a project has been established at the University of Wisconsin to determine the sequence of the genome of the bacterium *E. coli* [2]. The laboratory of Prof. F. Blattner is now sequencing large stretches of *anonymous* DNA - that is, DNA whose function is not known. In order to understand this DNA, it is necessary to locate the genes in it, and to determine the reading frames of these genes. We are using artificial neural networks (ANNs) to locate the regions of DNA sequences that code for proteins, and to predict the reading frames of these coding regions. These ANNs, along with a protein similarity-search program [11], are being successfully used by the Wisconsin *E. coli* Genome Project to characterize sequenced DNA, and to detect *frameshift errors* that have occurred during the sequencing process.

In this paper we describe how we have adapted the approach of Uberbacher and Mural [13] that identifies coding regions in human DNA. Uberbacher and Mural's neural network is used in the Gene Recognition and Analysis Internet Link (GRAIL) system. We have modified their approach to apply to lower organisms, whose gene structure is fundamentally different from that of humans. The experiments we report compare the performance of ANNs using Uberbacher and Mural's approach to several other neural network approaches and to conventional reading-frame prediction algorithms. Our experiments indicate that neural networks can outperform the best conventional methods. Additionally, our experiments demonstrate the importance of choosing a good representation for the data that is used as input to the networks.

This paper is organized as follows: we first provide a brief description of the molecular biology underlying the problem that we are addressing. We then describe conventional approaches to reading-frame prediction. Section 4 describes how ANNs can be applied to the problem of predicting reading frames. The succeeding section describes various ways of representing DNA sequences to neural networks. Section 6 describes our experimental methodology, and the next two sections detail our experiments in applying ANNs to the reading-frame prediction task. The final section provides conclusions.

2 DNA and Protein Translation

This section provides a brief description of the biology that underlies reading-frame prediction; a more thorough treatment of this material can be found elsewhere [14]. A DNA strand is a linear sequence of *nucleotides* composed from the alphabet {A, G, T, C}. A DNA molecule comprises two strands organized as a double helix. Certain subsequences of a DNA strand, called *genes* or *coding regions*, encode *proteins*. Interspersed between the genes are areas, termed *noncoding regions* that do not encode proteins. Proteins

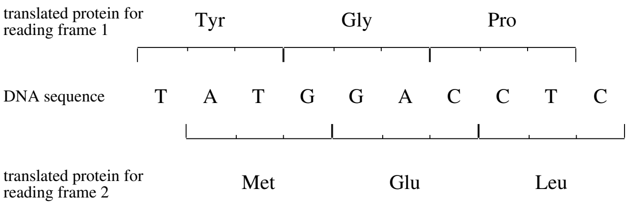

Figure 1: **The effect of reading frame on DNA translation.** The amino acid sequence that results from translation of the sequence in two different frames is depicted. The strand could also be translated in a third reading frame which is not shown here.

are also linear sequences; they are composed from the 20-character alphabet of *amino acids*. Proteins are important because they provide much of the structure and functionality of a cell.

Each consecutive string of three nucleotides in a gene encodes a single amino acid. The nucleotide triplets are called *codons* and the mapping from codons to amino acids is called the *genetic code*. The process of translating a gene into protein involves grouping nucleotides into triplets.[1] The *reading frame* refers to how the nucleotides of a DNA sequence are grouped into triplets as a gene is translated. Hence, each DNA strand has three possible reading frames. Since the reading frame defines the grouping, it is extremely important to the translation process. Figure 1 depicts the amino acids that would result from translation of the given DNA strand in two of the three possible reading frames.

There are three special codons, called *stop codons*, that signal the end of a gene. There are not any universal *start codons* that always signal the start of a gene, however. This means that finding genes in a DNA sequence is not a trivial task. Moreover, even if it is known that a particular region on a sequence encodes a protein, the amino acid sequence, and hence the protein, cannot be determined if the reading frame is not known. A further complication arises because laboratory errors often occur in the sequencing process. A *frameshift error* involves the mistaken insertion or deletion of a nucleotide. Because of the triplet nature of the genetic code, a frameshift error can have a devastating effect on the prediction of the amino acid sequence translated from a given gene. Once the translation process is out of frame, the predicted protein will bear no resemblance to the actual protein.

Our research involves using neural networks to:

1. locate the coding regions in a DNA sequence,
2. determine the reading frame for coding regions,
3. detect frameshift errors in coding regions.

3 Conventional Approaches

A number of methods have been developed that find genes and reading frames by exploiting statistical properties of coding regions [4, 5, 12]. Specifically, amino acid, codon, and nucleotide compositions can be measured and used to distinguish coding from noncoding regions. There are three factors which influence these compositions in sequences that encode a protein. First, some amino acids are used more frequently than others in proteins. Second, due to the degeneracy of the genetic code, there are unequal numbers of codons for different amino acids. Third, in any organism, for any given amino acid, the codons will not be equally used. This third factor is termed the *codon preference* of the organism.

The approaches to reading frame prediction that we discuss operate by generating predictions based upon a relatively small, fixed-length "window." By sliding the window along a DNA sequence, the algorithms are able to generate continuous signals that show the prediction for each reading frame along the length of the sequence. Figure 2 shows a set of reading frame signals that we generated using Staden's *codon usage method* [12]. Each signal corresponds to a particular reading frame. The presence of a coding region in a sequence is indicated by a relatively high signal in one of the frames. Noncoding regions are indicated by relatively low signals in all frames. A sharp transition in a signal may indicate the beginning of a gene, the end of a gene, or a frameshift error.

We have found Staden's codon usage method to be the most effective of the conventional approaches for finding reading frames. Given a window of nucleotides, this approach uses Bayes' formula to estimate the probability that each of the three reading frames of a strand encodes a protein. For a given sequence S occupying the window, the probability that frame i is coding (C_i) is calculated by:

$$P(C_i \mid S) = \frac{P(S \mid C_i) \times P(C_i)}{\sum_{f=1}^{3} P(S \mid C_f) \times P(C_f)} \qquad (1)$$

[1] DNA is actually *transcribed* into RNA which is then translated to protein. For the purposes of this paper, however, the transcription step can be ignored.

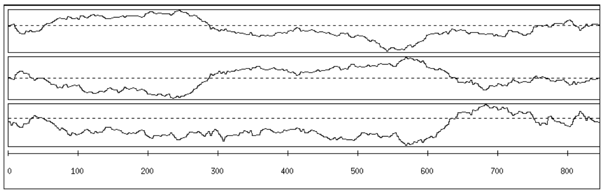

Figure 2: **Reading frame signals produced by the Staden algorithm.** The signals were generated on a first-pass, unedited sequence from F. Blattner's laboratory. The gene for *peripheral membrane protein U* starts at position 61 and ends at position 792. The sequence contains a frameshift error at position 283, one at position 637, and four located between 740 and 774. The vertical scale of each signal is $\log \frac{P}{(1-P)}$, where P is the probability for a frame calculated by the Staden algorithm.

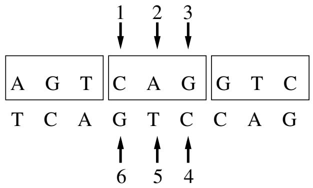

Figure 3: **Nucleotide positions in a codon.** The top strand is the coding strand and the boxes show how the nucleotides are grouped into codons. The numbers show the codon positions of six of the nucleotides in the sequence. Networks with six output units are trained to predict the codon position of the nucleotide in the center of the window.

The prior probability that each frame is coding, $P(C_i)$, is estimated as the number of triplets in frame i that can be formed in the window, divided by the number of triplets that can be formed in the window in all three frames. (As the window size is increased, $P(C_i)$ approaches $\frac{1}{3}$ for all three frames.) The conditional probabilities, $P(S \mid C_i)$, are estimated by compiling a table of the frequencies at which various codons occur in known genes. In machine learning terms, this table can be thought of as a training set for the Staden algorithm. The frequency value for each codon is an estimate of the conditional probability that the codon occupies each position in a sequence S, given that S encodes a protein. Staden makes the assumption that the codons that compose a gene are independent of each other, and thus makes the following estimation:

$$P(S \mid C_i) = \prod_{j=1}^{n} P(S_i(j) \mid C_i) \tag{2}$$

$S_i(j)$, in this equation, is the jth triplet in frame i in sequence S.

An additional assumption made by the Staden algorithm is that the given sequence encodes a protein in one of the three reading frames on the strand under consideration. Noncoding regions, consequently, are indicated by roughly equal probabilities for all three frames.

We hypothesized that an artificial neural network would be able to outperform the Staden algorithm because it would not be constrained by the independence assumption of neighboring codons. An ANN might be able to capture important information about the joint probabilities of certain codons occurring near each other. Section 7 addresses this hypothesis.

4 Using ANNs to Predict Reading Frames

As with the reading-frame determination method of Staden [12], we train artificial neural networks to make predictions given a fixed-length window of the DNA sequence of interest. Artificial neural networks provide an approach to machine learning characterized by simple processing units linked to each other by weighted connections. The state of a unit at any given time is represented by its *activation*, which is typically a real-valued number in the range [0, 1]. The *input* layer of a network contains units whose activations represent feature values of the problem domain to which the ANN is being applied. The units in the *output* layer of a network represent the decisions made by the network. Interposed between the input units and the output units, there can be a number of *hidden* layers of units. The activations of hidden and output units are computed by passing the weighted input to each unit through an *activation function* which is typically nonlinear. Comprehensive introductions to neural networks can found elsewhere [6, 8].

There are many ways in which the nucleotides occupying the window can be represented as activations on the input units. Section 5 describes the representations that we investigate in our experiments. The output representation that we use for our networks involves six output units. A network is trained to predict the position within a codon that the nucleotide in the center of the window occupies. As shown in Figure 3, if the part of the strand in the window is a coding region, then the network is trained to predict whether the nucleotide in the center of the window is the first, second, or third nucleotide in a codon. If the part of the strand in the window is not a coding region, but the corresponding part of the complementary strand is in a gene, then the network is trained to predict codon positions of four, five, or six. Thus, the networks have three units for the possible codon positions on the given DNA strand, and three units for the possible codon positions on the complementary strand.

Predicting codon positions is equivalent to predicting the reading frame *relative* to the position of the window. Consider, for example, a window size of seven nucleotides, where the first reading frame relative to the window starts grouping nucleotide triplets from the left edge of the window, the second reading frame starts a triplet with the second nucleotide in the window, and the third reading frame starts with the third nucleotide in the window. In this case, predicting codon position 1 for the center nucleotide is equivalent to predicting frame 1, predicting codon position 2 is equivalent to predicting frame 3, and predicting codon position 3 is equivalent to predicting frame 2.

In order to generate reading-frame *signals*, as in Figure 2, a network is scanned along a DNA sequence. Signals are formed by concatenating activation values from the output units. The input window is advanced by only one nucleotide each time it is moved; thus, activation values from three different output units must be interleaved to form each signal.

We have also investigated output representations using one and three output units. We have found, however, that six-output networks produce the best prediction accuracies. Networks with six output units are trained to recognize coding regions in all six frames relative to the window, whereas three-output networks are trained to recognize only three frames, and one-output networks learn to recognize only one frame relative to the window. For the problem of predicting reading frames, we have found that neural networks tend to make fewer prediction errors with increasing numbers of output classes.

5 ANN Input Representations

A fundamental issue involved in applying artificial neural networks to reading-frame prediction is how to represent the DNA sequence that is in the input window. We have investigated three approaches: (1) providing the network with only the nucleotides that are present in the window, (2) providing the network the codons that are present in the window, and (3) providing the network with statistical features derived from the contents of the window.

The first approach that we investigate uses a *local encoding* of the nucleotides that occupy the window.

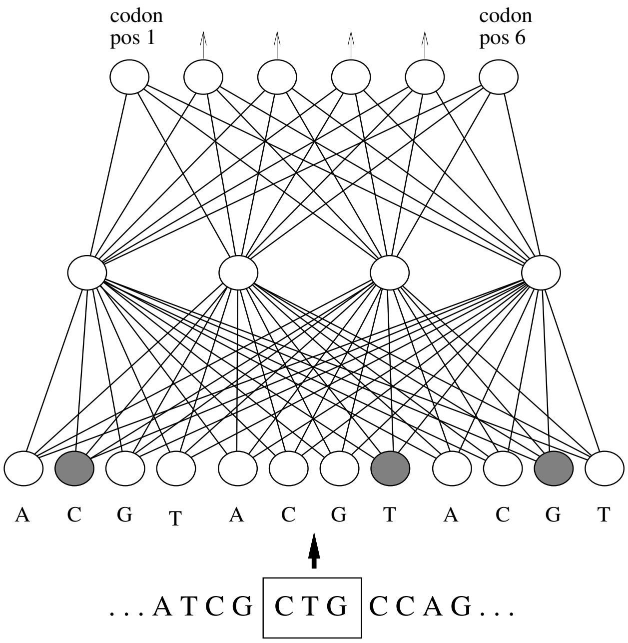

Figure 4: **Representing the input window as nucleotides using a local encoding.** The nucleotides in the window determine the activations of the input units. Shaded input units have activations of 1, the other input units have activations of 0. In this figure, the input window is only three nucleotides wide.

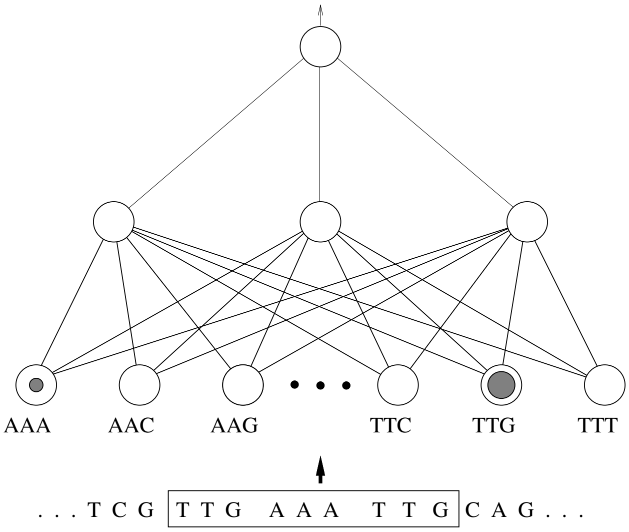

Figure 5: **Representing the input window as codons.** There are sixty-four input units for each frame. The activation of each input unit represents the number of occurrences of the corresponding codon in a particular frame in the window. In this figure, the input window is only nine nucleotides wide, and the units for only one frame are shown.

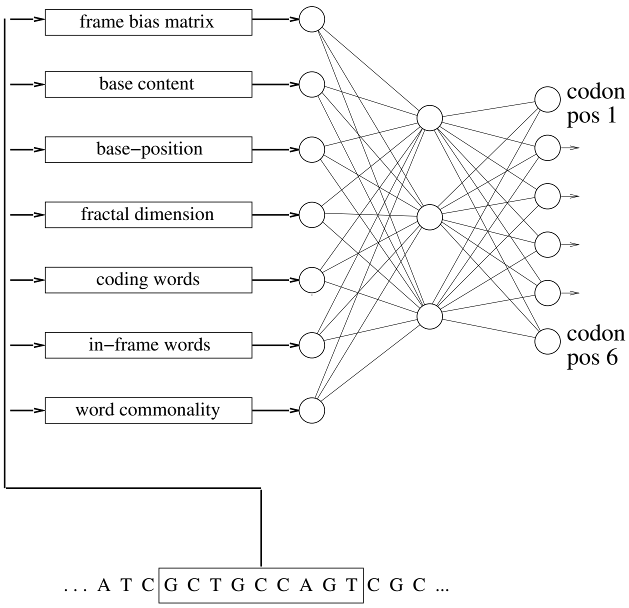

Figure 6: **Representing the input window as extracted features.** Statistics are derived from the nucleotides in the input window and these statistics are used as the input values to the ANN. The seven features in the figure are from Uberbacher and Mural. In this figure, each extracted feature is shown as involving only one value and hence one input unit. The actual networks that we use have 25 input units.

In a local encoding, one input unit is used for each possible value that a given feature can assume. As illustrated in Figure 4, four input units are used to represent each position in the input window. Each of the four input units represents one of the four nucleotides that could occupy the position.

Another approach to representing the input sequence is to extract some higher-level features from the raw sequence of nucleotides and to present these derived features to the network as input. This is essentially the approach employed by the Staden method, where the extracted features are the frequencies of the putative codons (nucleotide triplets) in the window. The second input representation that we investigate involves extracting the putative codons that are present in each frame in the window and representing the input window in terms of these codons. Thus, when the window is positioned over a gene, and it is positioned so that it is in-frame, the triplets in the window will actually be codons in a gene. This approach has been used by others to find protein coding regions in human DNA [9, 3]. As shown in Figure 5, this representation requires sixty-four input units for a network with one output unit. Each unit represents a particular codon and its activation represents the number of occurrences of that codon in the window. For a network with six output units, it is necessary to represent the triplets in all six frames; hence 384 input units are used.

Another input representation approach, depicted in Figure 6, is that of Uberbacher and Mural [13], who have described and investigated seven feature extractors that they use to locate coding regions in human DNA. We have adapted the approach of Uberbacher and Mural to *prokaryotes*, or organisms, such as *E. coli*, that lack nuclei.

In higher organisms, or *eukaryotes*, the DNA that encodes a gene may contain *intervening sequences*, or *introns*, that are spliced out before the gene is translated to a protein. In addition to discriminating genes from intergenic regions, identifying coding sequences in *eukaryotic* DNA also involves recognizing introns. Uberbacher and Mural have developed an ANN that distinguishes introns from the coding (*expressed*) parts of eukaryotic genes, called *exons*. We have adapted the Uberbacher-Mural approach to finding genes in prokaryotic DNA, which does not have introns. Instead of classifying exons and introns, we train networks to identify the strand and the reading frame that encodes a protein.

Our feature extraction approach employs somewhat modified versions of six of the seven features used by Uberbacher and Mural. One of our modifications is to use all six values for the features that calculate a value for each frame. A second modification is to calculate the Uberbacher-Mural features on both of the DNA strands for a given window position. Briefly described below are the features that we adapted from Uberbacher and Mural:

1. *Frame bias matrix*: Each element of a 4 x 6 frame bias matrix represents the difference in frequency, relative to the mean, for the occurrence of A, C, G, and T in each of the six codon positions. A matrix is constructed for each of the six possible reading frames in the input window, and the correlation coefficients of these matrices with a reference matrix are used as input features. The reference matrix is constructed from a set of known genes.

2. *Fickett values*: An algorithm developed by Fickett [4] measures the A, C, G, and T-content of a sequence, and the degree to which each nucleotide is favored in each codon position. The Fickett algorithm weights these eight values to form a sin-

gle score indicating the probability that the sequence encodes a protein. While Uberbacher and Mural use the single Fickett score, we use the content and codon position values as input features so that a network can develop its own weighting.

3. *Dinucleotide fractal dimension*: The transitions in the frequencies of sequential dinucleotides can be characterized by a fractal dimension [13, 7]. We calculate this dimension for the dinucleotides occurring on both strands in the input window and subtract from each value the fractal dimension of a set of noncoding reference strands.

4. *Coding 6-tuple word preferences*: For each six nucleotide-long word in the input window, the logarithmic ratio of its normalized frequency of occurrence in coding versus noncoding strands is calculated. For each strand, these values are summed to form input features. Word frequencies are compiled from a set of reference sequences.

5. *Coding 6-tuple in-frame preferences*: This set of features is similar to the previous one, except that the words considered for each score are all in the same frame, and the frequencies considered are tabulated only for in-frame words. These scores are calculated for all six frames in the input window.

6. *Word commonality*: For each 6-tuple word in the input window, the logarithmic ratio of its normalized frequency of occurrence divided by its expected random frequency is calculated. These values are summed for all words in all six frames and combined into a single score.

The Uberbacher-Mural feature that we do not use is one that calculates similarity to certain classes of repetitive DNA. Although this is a useful feature in analyzing eukaryotic DNA, it is not relevant to prokaryotic organisms such as *E. coli*.

6 Experimental Methodology

The data that we use to evaluate our networks is taken from thirty *E. coli* genes. The BLAST program [10] was run on all pairs of genes in the data set to ensure that the presence of homologous genes would not distort training and testing accuracies. The criterion used to define homology is 50% identity when aligned at the protein level. We have partitioned the set of sequences into four sets, and all of our experiments involve a four-fold cross-validation methodology. In four-fold cross-validation, networks are trained using examples from thee of the sets and tested on examples from the fourth set. This procedure is repeated four times so that each set is used as the testing set once.

The networks are not trained on noncoding DNA sequences because it is problematic to isolate noncoding sequences of significant length. The *E. coli* genome is dense with coding regions [1], and it is difficult to verify that what appears to be a noncoding region does not actually contain any genes. We have experimented with including some regions from the phage λ, which are believed to be noncoding, in our data sets. We have found, however, that including these sequences in our training sets did not significantly improve the prediction accuracies on the λ sequences in the test set. The handling of noncoding sequences for this prediction task is still an open problem.

For all of our training and testing examples we assume that *E. coli* DNA does not contain regions where the two complementary strands have coding regions that overlap. In other words, there is only one correct prediction for any given input window. We believe that this is not a harmful assumption because known cases of overlapping genes in *E. coli* are very rare.

Each network is trained using 10,000 window-sized DNA sequences randomly selected from three of the folds. An input window width of sixty-one nucleotides is used for all representations. There is a trade-off involved in selecting a window size for the task of predicting reading frames. Better prediction accuracies usually result from larger windows since they provide more information to networks. On the other hand, large windows result in worse resolution for detecting gene boundaries and frameshift errors. We have found that this window size provides an adequate compromise of accuracy and resolution.

The networks are trained for, at most, 100 epochs. A *tuning set* consisting of ten percent of each training set is used to determine when to stop training so that networks do not "overfit" the training data. Members of the tuning set are not presented to the network as ordinary training examples, but instead are used during learning to estimate the accuracy of the network on unseen examples. Each network is then tested on all of the examples in the left-out fold. The total number of testing examples is 65,116 window-sized sequences.

We use two different measures to assess the performance of networks on the testing data. The first measure is simply the percentage of windows for which the network generates the correct prediction. The output unit with the highest activation determines the predicted codon position for each example. The second measure is the *run-length* of the errors made by a network as it generates predictions for continuous sequences. The run-length of a sequence of errors is simply the number of contiguous nucleotides for which a network generates incorrect predictions. Run-length provides a performance metric that is complementary to simple window correctness because it reflects the fact that not all errors made by the network are equally significant. Isolated errors and short stretches of contiguous errors do not pose serious problems for the task of predicting reading frames, whereas errors with long run-lengths do.

7 Comparing Input Representations

The objective of our first experiment is to determine the effect of input representation on reading frame prediction accuracy. In this experiment we assess the performance of neural networks using the *nucleotides*, *codons*, and Uberbacher-Mural representations described previously. We also compare the performance of ANNs using these representations to two conventional approaches to reading frame prediction: the Staden and Gribskov algorithms.

The Gribskov [5] algorithm is similar to the Staden approach; the primary difference is that the Gribskov method tries to factor out the amino-acid composition of the translated protein. As with the Uberbacher-Mural features, we have extended the Staden and Gribskov methods to consider both strands for a given window position. Thus, both algorithms produce prediction values for six, instead of three reading frames. We have found that this extension results in better prediction accuracies.

This experiment also tests a network that uses an input representation consisting of the values generated by the Staden and Gribskov algorithms in addition to the Uberbacher-Mural features. All of the networks that we use are fully-connected between layers. In order to select the number of hidden units to be used for networks with a given input representation, we tested all representations using 40, 20, 10, 5 and no hidden units. We report the results for the *best* network architecture for each input representation (20 hidden units for the Uberbacher-Mural representation, 10 for the Uberbacher-Mural-Staden representation, 20 for the *bases* representation, and no hidden units for the *codons* representation).

The various reference statistics used to calculate the Uberbacher-Mural, Staden, and Gribskov features are extracted from a separate set of thirty-eight genes which are also taken from GenBank. The Staden and Gribskov algorithms are tested in the same manner as the neural networks. The training data for these algorithms consists of a set of genes used to calculate codon usage frequencies. As with the ANNs, the Staden and Gribskov algorithms are run on four separate testing sets, and in each case they used the same set of genes for training data as the networks.

Table 1: **Performance of various approaches.** The correctness percentages indicate the number of window-sized sequences for which the correct frame is predicted. The accuracy difference between the Uberbacher-Mural-Staden network and the Staden algorithm is statistically significant to the 99.5% confidence level using the Student t-distribution.

approach		% correct
neural nets	nucleotides	59.51
	Uberbacher-Mural	84.97
	codons	87.45
	Uberbacher-Mural-Staden	89.16
conventional	Gribskov	73.13
	Staden	87.82

The results of the experiment are reported in Table 1. The correctness percentages indicate the number of window-sized sequences in the test sets for which correct predictions are made. The results presented in Table 1 show that the input representation used by a neural network has a significant effect on the ability of the network to learn the task. The performance of the networks using the *nucleotides* representation falls far short of the other networks. Apparently, the hidden units used by these networks have failed to find representations for useful higher-level features such as codons.

The performance of the networks with the *codons* representation is about the same as the performance of Staden method. We had expected the *codons* networks to outperform the Staden approach by representing some information concerning the joint probabilities of codons. Farber *et al.* [3] have found that perceptrons are able to do this for a similar task.

Additionally, the results indicate that, for *E. coli*, the Staden algorithm is a better predictor of reading frames than a neural network trained using only the Uberbacher-Mural features. The best approach, however, is to use networks that are given the Staden and Gribskov signals in addition to the Uberbacher-Mural features. This result seems to indicate that there is complementarity in the features used by the Uberbacher-Mural and Staden methods.

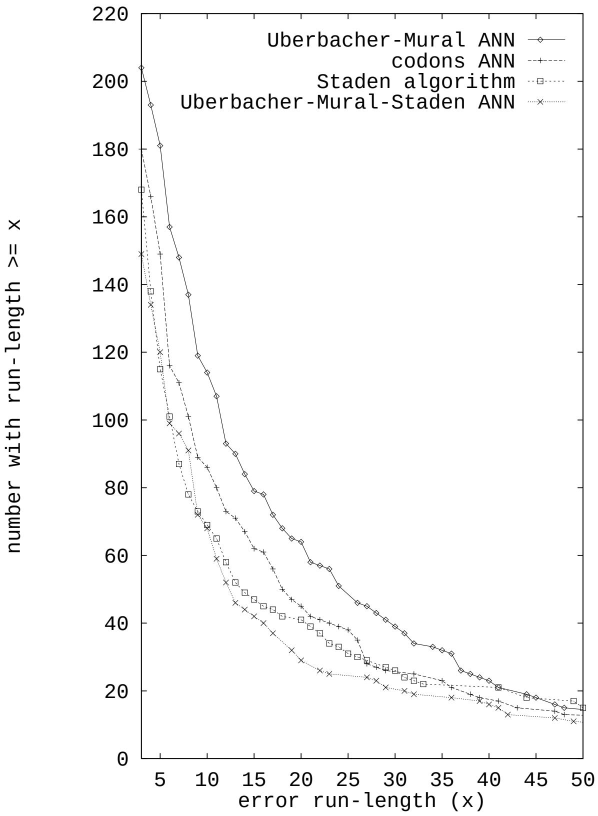

Figure 7: **Cumulative distribution of error run-lengths for four approaches.** The X-axis represents error run-length. The Y-axis represents the number of errors whose run-length is greater than or equal to a given point on the X-axis. Only errors with run-length of three nucleotides or greater are shown here.

Figure 7 shows the cumulative distributions of error run-lengths for four of the approaches. An error, for the purposes of this plot, is considered to be a successive sequence of incorrect predictions. The run-length of an error is the length, in nucleotides, of the sequence of wrong predictions. The x-axis in the plot represents error run-length. The y-axis represents the number of errors whose run-length is greater than or equal to a given point on the x-axis. For example, the networks using the Uberbacher-Mural representation made 114 errors with run-lengths of 10 nucleotides or longer. Only run-lengths of three nucleotides or greater are shown here.

Whereas Table 1 shows the number of incorrect predictions made by each approach, Figure 7 shows how these errors tend to cluster. It can be seen, for example, that the Uberbacher-Mural networks tend to make longer runs of incorrect predictions than the other approaches. While only about 6% of the errors made by the Staden algorithm have run lengths of 10 or more nucleotides, nearly 13% of the errors made by Uberbacher-Mural networks do. The networks using the *codons* representations also tend to make longer runs of incorrect predictions than the Staden algorithm and the Uberbacher-Mural-Staden networks. The distributions of these latter two approaches are very similar.

Finally, it should be pointed out that the networks with the *nucleotides* input representation effectively had less training data than the other networks. In addition to training examples, the other networks benefited from information present in the various reference tables provided to the feature extraction routines. In order to test whether this additional training data explained the difference in performance between the input representations, we constructed a *learning curve* (not shown here) for the *nucleotides* networks. A learning curve plots performance (the dependent variable) against the amount of training data (the independent variable). This analysis indicated that the performance of the *nucleotides* networks sharply levels off at around 2000 training examples and seems to have an asymptote around 60%. Furthermore, since ANN training is quite slow, the feature-extraction process is an efficient way to exploit information available in the data.

8 Evaluating the Uberbacher-Mural Features

The previous experiment demonstrated the value of using an input representation of extracted statistics. We conducted a second experiment to determine how much information is contributed by each individual feature in the Uberbacher-Mural representation. Those features that provide a value for each reading frame can be used by themselves to classify sequences. Similarly, those features which provide a value for each strand of a sequence can be used to classify sequences according to which strand is coding. In both cases, the classification is done by choosing the frame/strand with the greatest value. The results of applying individual features to the classification task are shown in Table 2. The Fickett features and the word commonality features are left out of this experiment. Since there are eight Fickett values and only one word commonality value, we cannot use these features to classify

Table 2: **Predictiveness of Uberbacher-Mural features.** These percentages indicate the ability of individual features to identify the correct reading frame and/or the correct strand.

feature	frame % correct	strand % correct
fractal dimension	-	61.42
frame bias matrix	63.10	68.40
Gribskov scores	73.13	79.00
coding word preferences	-	81.43
in-frame word preferences	85.05	87.97
Staden scores	87.82	90.28

Table 3: **Leaving one Uberbacher-Mural feature out.** The difference in test set correctness between networks using the Uberbacher-Mural features and networks with one feature left out of the input representation. Differences are reported in percentage points.

feature left out	difference in % correct
coding word preferences	-0.11
word commonality	-0.24
fractal dimension	-0.25
frame bias matrix	-0.36
fickett values	-0.62
in-frame word preferences	-9.29

sequences by simply selecting the largest from a set of values.

In addition to testing features individually, we also performed the converse experiment: leaving individual features out of the input representation. As before, networks are trained and tested using four-fold cross-validation. In each training run, however, one of the six Uberbacher-Mural features is left out of the input representation. The results of this experiment are presented in Table 3. The numbers in this table represent the percentage-point difference in test-set correctness between networks using the Uberbacher-Mural input representation and networks with each feature left out.

The results in Table 2 indicate that the Staden scores and in-frame word preferences are the best reading-frame predictors among the input features. All of the Uberbacher-Mural features, however, seem to provide some information useful in classifying sequences.

The results in Table 3 indicate that there is a fair amount of redundancy in the Uberbacher-Mural features. Except for in-frame word preference, leaving any Uberbacher-Mural feature out of the input representation does not significantly affect the prediction accuracy of the network. Leaving the in-frame word preference feature out, however, does have a significant negative effect. This result, as well as the gain in prediction accuracy resulting from including Staden signals in the Uberbacher-Mural networks, seems to indicate the significance of codon preference in *E. coli* for the sequences that are presently in the GenBank database.

9 Conclusions

We have applied artificial neural networks to the task of locating coding regions in DNA sequences and determining the reading frames for these coding regions. We have extended the Uberbacher-Mural approach to apply to prokaryotic DNA and to predict reading frames in addition to coding regions. We have shown that the input representation used for a neural network makes a significant difference in the performance of the network. We have also compared the performance of our ANNs to two commonly-used conventional algorithms for reading frame prediction and found that the networks are able to outperform both of them.

Our ANNs are currently being used in the University of Wisconsin *E. coli* Genome Project to assist in the location of genes in uncharacterized DNA sequences, and to detect frameshift errors that have occurred in the sequencing process. We believe that the application of our networks in this laboratory will provide further insight into the problem of identifying genes in DNA.

10 Acknowledgements

This work was partially supported by Department of Energy Grant DE-FG02-91ER61129, National Science Foundation Grant IRI-9002413, and Office of Naval Research Grant N00014-90-J-1941. Prof. Frederick Blattner of the University of Wisconsin Genetics Department provided many helpful comments at various stages of this research. Dr. Michael Cherry at Massachusetts General Hospital kindly provided the list of *E. coli* genes used in constructing his codon usage table.

References

[1] Fred Blattner, 1992. Personal communication.

[2] Donna L. Daniels, Guy Plunkett III, Valerie D. Burland, and Frederick R. Blattner. Analysis of the Escherichia coli genome: DNA sequence of the region from 84.5 to 86.5 minutes. *Science*, 357:771–778, 1992.

[3] R. Farber, A. Lapedes, and K. Sirotkin. Determination of eucaryotic protein coding regions using neural networks and information theory. Technical Report LA-UR-90-4014, Los Alamos National Laboratory, Los Alamos, NM, 1990.

[4] James W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 10(17):5303–5318, 1982.

[5] M. Gribskov, J. Devereux, and R. Burgess. The codon preference plot: Graphic analysis of protein coding sequences and prediction of gene expression. *Nucleic Acids Research*, 12(1):539–549, 1984.

[6] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, Redwood City, CA, 1991.

[7] Kenneth J. Hsu and Andreas J. Hsu. Fractal geometry of music. *Proceedings of the National Academy of Sciences*, 87:938–941, February 1990.

[8] Kevin Knight. Connectionist ideas and algorithms. *Communications of the ACM*, 33(11):59–74, 1990.

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

[10] E. W. Myers, W. Miller, S. F. Altschul, W. Gish, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 214, 1990.

[11] J. W. Shavlik. Case-based reasoning with noisy case boundaries: An application in molecular biology. Technical Report 988, University of Wisconsin, Madison, WI, 1990.

[12] R. Staden. Finding protein coding regions in genomic sequences. *Methods in Enzymology*, 183:163–180, 1990.

[13] Edward C. Uberbacher and Richard J. Mural. Locating protein coding regions in human DNA sequences by a multiple sensor – neural network approach. *Proceedings of the National Academy of Sciences*, 88:11261–11265, 1991.

[14] James D. Watson, Nancy H. Hopkins, Jeffrey W. Roberts, Joan Argetsinger Steitz, and Alan M. Weiner. *Molecular Biology of the Gene*, volume I. Benjamin/Cummings, Menlo Park, CA, fourth edition, 1987.