Appears in the Proceedings of the 13th International Joint Conference on Artificial Intelligence (8/93)

Learning to Represent Codons: A Challenge Problem for Constructive Induction*

Mark W. Craven and Jude W. Shavlik
Computer Sciences Department
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706, U. S. A.
{craven, shavlik}@cs.wisc.edu

Abstract

The ability of an inductive learning system to find a good solution to a given problem is dependent upon the representation used for the features of the problem. Systems that perform constructive induction are able to change their representation by constructing new features. We describe an important, real-world problem – finding genes in DNA – that we believe offers an interesting challenge to constructive-induction researchers. We report experiments that demonstrate that: (1) two different input representations for this task result in significantly different generalization performance for both neural networks and decision trees; and (2) both neural and symbolic methods for constructive induction fail to bridge the gap between these two representations. We believe that this real-world domain provides an interesting challenge problem for constructive induction because the relationship between the two representations is well known, and because the representational shift involved in constructing the better representation is not imposing.

1 Introduction

The ability of an inductive learning system to find a good solution to a given problem is dependent upon the representation used for the features of the problem. Work in *constructive induction* [Michalski, 1983] has focused on methods for constructing new features, thereby changing the problem representation [Matheus and Rendell, 1989; Matheus, 1990; Pagallo and Haussler, 1990]. Similarly, one of the touted virtues of multi-layer artificial neural networks is that their hidden units are able to construct new features from the given input features. In this paper we describe a difficult real-world problem that we believe provides an interesting challenge for inductive learning systems that adjust their input representations. We also present a set of experiments that support our claim that this task poses a challenge to existing systems that perform constructive induction.

The real-world task that we discuss is the problem of identifying genes in DNA sequences. As part of the Human Genome Project, many biological laboratories are now in the process of ascertaining the DNA sequence of humans and other organisms. One of the primary challenges of these efforts is to distinguish the more interesting parts of DNA sequences from the parts with little or no functionality. Several researchers have addressed this problem using neural networks and have found that the input representation has a significant impact on the generalization performance of the trained networks [Craven and Shavlik, 1993; Farber *et al.*, 1992; Uberbacher and Mural, 1991]. In this paper we further investigate two particular input representations. The performance difference between these two representations is especially intriguing because the relationship between the representations is well known, and furthermore, the representational shift involved in going from the weaker representation to the better one involves only forming an appropriate set of conjunctions.

In this paper we present two sets of experiments. We first demonstrate the effect of input representation on generalization performance for this problem domain, using both a symbolic learning method (Quinlan's C4) and a neural learning technique (Rosenblatt's perceptrons). We then evaluate two approaches to constructive induction on this problem and show that neither of them seem to be able to construct the features needed for good generalization. The first approach to constructive induction involves simply adding hidden units to the networks used in the first experiment. The second approach that we investigate is CITRE [Matheus and Rendell, 1989], which performs constructive induction on decision trees.

2 The Problem Domain

This section provides a brief description of the problem that serves as a testbed for our experiments. A more thorough treatment of the biology underlying the problem can be found in a genetics textbook.

A DNA strand is a linear sequence of *nucleotides* composed from the alphabet {A, G, T, C}. Certain subsequences of a DNA strand, called *genes* serve as blueprints for *proteins*. Proteins are important because they pro-

*This research was partially supported by DOE Grant DE-FG02-91ER61129, NSF Grant IRI-9002413, and ONR Grant N00014-90-J-1941.

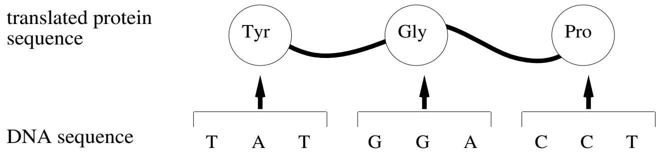

Figure 1: **DNA and protein translation.** Depicted here is a DNA sequence and the amino acid sequence (protein) that results from its translation. Each bracket delineates a codon; collectively, the brackets show the reading frame for this sequence.

vide most of the structure, function, and regulatory mechanisms of cells. Interspersed between the genes are areas, termed *noncoding regions*, that do not encode proteins. An important problem in biology is to be able to distinguish the coding from the noncoding regions of DNA sequences.

Proteins are also linear sequences; they are composed from the 20-character alphabet of *amino acids*. As illustrated in Figure 1, each consecutive string of three nucleotides in a gene encodes a single amino acid (e.g., "GGA" encodes glycine). The nucleotide triplets are called *codons* and the mapping from codons to amino acids is called the *genetic code*. The genetic code is almost universal across species and is well known. The process of translating a gene into protein involves grouping nucleotides into codons and inserting the amino acid encoded by each codon into the protein chain being synthesized.

In order to determine the amino-acid sequence encoded by a given DNA sequence, it is necessary to know the *reading frame* of the sequence. The *reading frame* refers to how the nucleotides of a DNA sequence are grouped into codons as a gene is translated. As an analogy, consider trying to decode a bit stream that contains a message encoded in ASCII. Unless the bits are correctly grouped into bytes, the decoded message will be nonsense.

The problem that we address in this paper is the following: given a relatively small, fixed-length "window" on a DNA sequence, predict whether or not the sequence is part of a gene that is "in-frame " with the window. The window is considered to be in-frame with a gene when the leftmost nucleotide in the window is the first nucleotide in a codon of the translated gene. Thus the problem involves classifying input sequences into two classes: *coding* and *noncoding*. Other researchers have also investigated neural network approaches [Farber *et al.*, 1992; Uberbacher and Mural, 1991] to the problem of finding genes in DNA sequences.

3 The Effect of Input Representation

In this section we describe two different representations that can be used for DNA sequences and compare the resultant performance for these representations using both perceptrons and decision trees. The first approach is to use a binary encoding of the nucleotides that are present in the input window. This input representation requires four features for every nucleotide position in the input window, where each feature represents one of the four nucleotides that could occupy the position. We will refer to this as the *nucleotides* representation. The second approach involves representing the codons that are present in the window and are in-frame with respect to the window. This input representation involves sixty-four binary features for every codon position in the window, where each feature represents one of the codons that could occupy the position. We will refer to this as the *codons* representation. Figure 2 and Figure 3 depict artificial neural nets using the *nucleotides* and *codons* representations respectively.

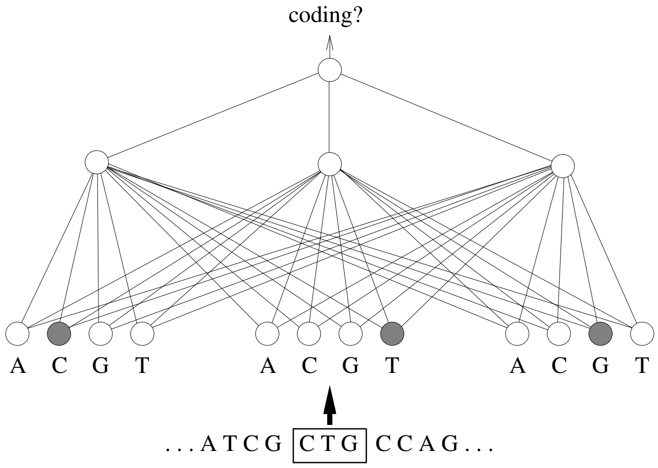

Figure 2: **Representing the input window as nucleotides.** The nucleotides in the window determine the activations of the input units. Shaded input units have activations of 1, the other input units have activations of 0. In this figure, the input window is only three nucleotides wide.

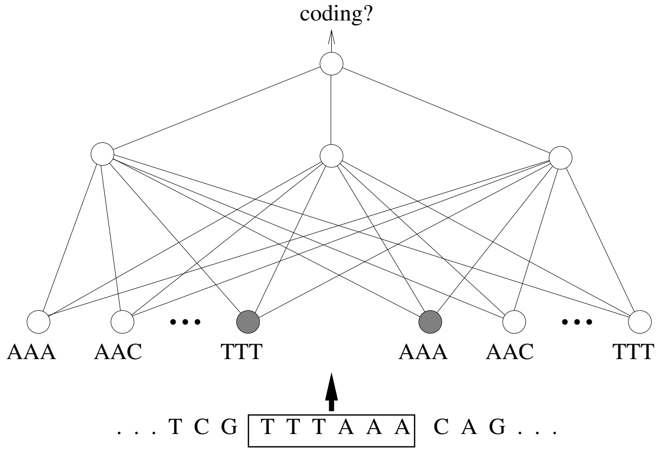

Figure 3: **Representing the input window as codons.** There are sixty-four input units for each codon in the window. In this figure, the input window is only two codons wide.

In order to evaluate the effect of input representation for the problem of recognizing genes in DNA, we construct generalization curves (described below) for both representations using the C4.5 decision tree algorithm [Quinlan, 1993], and perceptrons [Rosenblatt, 1958]. A

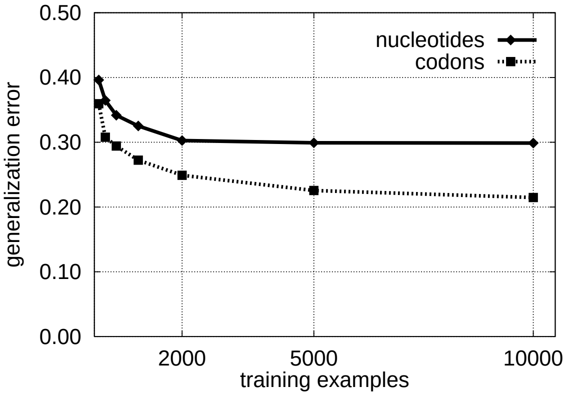

Figure 4: **Perceptron generalization curves.** The solid line shows the observed generalization curve for the *nucleotides* representation. The dashed line shows the generalization curve for the *codons* representation. *Generalization error* is the percentage of misclassified test-set examples.

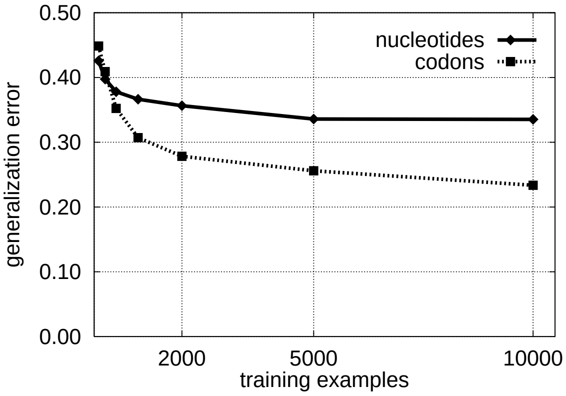

Figure 5: **C4.5 generalization curves.** The solid line shows the observed generalization curve for the *nucleotides* representation. The dashed line shows the observed generalization curve for the *codons* representation. *Generalization error* is the percentage of misclassified test-set examples.

perceptron is a neural network with no hidden units. A window size of 15 nucleotides is used to determine the input features for both representations. We partition the set of sequences into four sets, and all of our experiments involve a four-fold cross-validation methodology.[1] For both representations, classifiers are trained on example sets that range from 100 to 10,000 examples. For a given run, each successive training set is a superset of the previous training set. The classifiers are tested on a disjoint set of 5,000 examples after learning each training set, and the accuracies on this set are plotted. All training and testing sets contain approximately 50% *coding* sequences and 50% *noncoding* sequences.

The perceptrons are trained until convergence using a conjugate-gradient algorithm.[2] A *tuning set* consisting of ten percent of each training set is used to determine when the network weights are saved 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. Pessimistic pruning [Quinlan, 1993] is used on the trained decision trees to help avoid overfitting.

Figure 4 shows the observed generalization curves for perceptrons using the *nucleotides* and *codons* input representations. Figure 5 shows the observed generalization curves for decision trees using both representations. A generalization curve plots test-set error on the y-axis against training set size on the x-axis. It can be seen that perceptrons achieve better generalization performance than decision trees for both input representations. This result indicates that the inductive bias of neural networks is better suited to this task than the bias of decision trees. Furthermore, for both learning methods, using the *codons* representation results in classifiers that generalize substantially better than those using the *nucleotides* representation.

4 Gene Recognition and Constructive Induction

Because the defining characteristics of genes are more readily apparent at the codon level than the nucleotide level, the *codons* representation of DNA sequences is obviously a better representation for the task of recognizing genes. However, what if we did not know *a priori* that the *codons* representation was a reasonable input representation to try for this task? Would this representation be discovered by state-of-the-art learning systems that perform constructive induction?

There are several reasons why the problem of finding genes provides an interesting testbed for constructive induction research. First, it is a real-world problem that can be "reverse-engineered." We know that codons provide a good representation for the problem, and the mapping between nucleotides and codons is known. Second, the representational shift involved in going from nucleotides to codons is neither trivial nor overly-complex. The *codons* representation does not contain any information that is not present in the *nucleotides* representation, and the features of the *codons* representation, namely the codons themselves, are simply ternary conjunctions of adjacent nucleotides. Thus we would expect that a fairly general feature-construction algorithm would be able to construct codon features from the nucleotide features. Third, the process of finding the *codons* representation is, to some extent, a problem of scientific discovery. The process of "cracking" the genetic code in the early 1960's involved discovering the mapping between sequences of nucleotides and sequences of amino acids.

[1] In four-fold cross-validation, classifiers are trained using examples drawn from three 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.

[2] A conjugate-gradient algorithm obviates the need for learning-rate and momentum parameters.

Two parts of this discovery, in particular, are manifested in the process of learning the *codons* representation: (1) determining that each consecutive nucleotide triplet encodes a single amino acid; and (2) determining that the code is non-overlapping (i.e., codons do not overlap each other).

In the following sections we describe experiments that involve applying both neural and symbolic constructive induction approaches to the problem of recognizing genes. The question that motivates these experiments is whether or not general methods for constructive induction are able to discover the "right" representation (i.e., the *codons* representation) for this problem when given only the *nucleotides* representation?

5 Constructing Features in Networks

The artificial neural networks used in the previous experiment were perceptrons, meaning that they did not have any hidden units. When used for classification tasks, such as the gene-recognition problem, perceptrons are able only to make linear discriminations in their input space. The role of hidden units in a neural network is to transform the input space into a different representation – one in which two classes may be separated by a single linear boundary. Thus, the concepts represented by hidden units can be thought of as constructed features. The question that we address in this section is whether or not neural networks using the *nucleotides* representation, when given a sufficient number of hidden units, are able to construct features that enable them to approach the generalization performance of networks using the *codons* representation. It is important to note that we are not particularly concerned with exactly what features the networks learn to encode with their hidden units. It is possible that the hidden units could learn a local encoding of codons, or they may learn a distributed representation of codons, or some entirely different, but useful, set of features.

In order to investigate this question, we construct generalization curves using networks with different numbers of hidden units. The methodology and the data sets used to train and test the multi-layer networks are the same as in the previous experiment. Networks are trained using a conjugate-gradient algorithm until a local minimum is reached. A tuning set is used to determine when the weights are saved. The networks that we test have 5, 10, 20 and 40 hidden units. Each hidden unit is fully-connected to the set of input units. The number of free parameters (weights + biases) in a neural network gives a rough indication of the *capacity* of the network [Baum and Haussler, 1989]. The 40-hidden-unit networks used in this experiment have 2481 parameters. In contrast, it would take only 1601 parameters to encode by hand a network that had a layer of 320 hidden units hard-wired to represent each of the 64 codons in each of the five codon positions in the window. Thus, although the networks in this experiment do not provide a topology that enables them to form a local representation of the codons in the window, they should have sufficient complexity to form an alternative, distributed encoding of similar features. We do not allow the networks to form a local representation of codons, because to do so would require either hard-wiring codon features into the networks, or using networks with an extremely large number of parameters. For example, a network with 320 fully-connected hidden units has 19,521 parameters. Using a rule-of-thumb that recommends two training examples per parameter [Hinton, 1989], such a network would require training with more examples than is practicable.

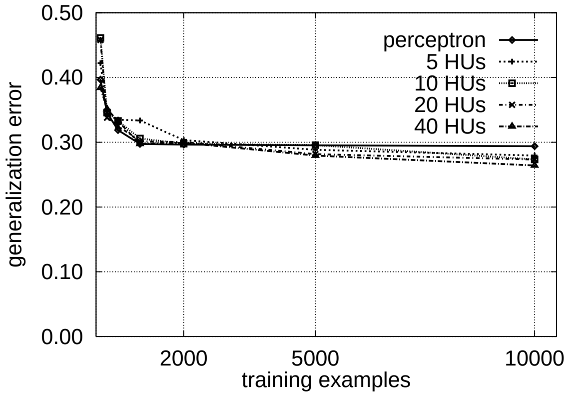

Figure 6: **Neural network generalization curves for the *nucleotides* representation.** The curves show observed generalization for networks with 5, 10, 20 and 40 hidden units. In contrast, perceptrons using the *codons* representation result in generalization error of about 0.21 for 10,000 training examples.

Figure 6 shows the generalization curves for networks with 5, 10, 20 and 40 hidden units, as well as the *nucleotides* perceptrons used in the first experiment. Although networks with hidden units offer an improvement over the generalization performance of perceptrons using the *nucleotides* representation, this improvement is not enough to match the performance of the perceptrons using the *codons* representation. This result indicates that the fully-connected networks are failing to represent codons with their hidden units. Although the networks with 40 hidden units should have sufficient capacity to form a representation of codons, it is possible that networks with more hidden units would learn such a representation.

6 Constructing Features in Trees

A number of algorithms have been developed to perform feature construction using decision trees as the concept description language [Matheus and Rendell, 1989; Pagallo and Haussler, 1990]. The CITRE system [Matheus and Rendell, 1989] provides a general approach for constructive induction on decision trees. In this section we describe an experiment in which we train decision trees using the *nucleotides* representation, and then use CITRE to construct features on these decision trees. The motivation for this experiment is to see if a symbolic system for constructive induction, such as CITRE, is able to bridge the generalization gap between the *nucleotides* representation and the *codons* representation.

The CITRE approach to feature construction involves an iterative cycle of learning a decision tree, constructing new features, and then learning a new tree using the constructed and original features. CITRE uses a learned decision tree to suggest constituent features to be used in the constructed features. As outlined by Matheus, the CITRE approach has four aspects:

1. the *detection* of when new features are required
2. the *selection* of relationships used to define new features
3. the *generalization* of new features
4. the global *evaluation* of constructed features

We will now discuss how we address these four aspects in our experiment.

Matheus' criterion for *detection* is that feature construction should be performed whenever disjunctive regions, as evidenced by the presence of more than one positively-labelled leaf, are detected in a decision tree (for our purposes, a positively-labelled leaf is one labelled *coding*). We believe that for many real-world tasks this is probably too stringent of a criterion because it requires the concept to be represented as a conjunction. In our experiment, we finesse the issue of defining a good detection criterion; instead we just try to establish a lower bound on the generalization error over a fixed number of iterations of tree building and feature construction.

The *selection* process involves forming conjunctions of pairs of Boolean features. Matheus describes a number of ways in which pairs of features can be selected [Matheus, 1990]. In this experiment we use the *adjacent* method which selects all adjacent pairs of tests that occur on decision-tree branches that lead to positively-labelled leaves. The selection process can also exploit domain knowledge to narrow the set of candidate constructed features. We do not use any domain knowledge in the selection process because we are interested in determining how well we can do without using such information.

The *generalization* step enables CITRE to generalize the features constructed in the selection process. For example, constants may be replaced by variables. In this experiment we do not perform any generalization of constructed features.

The *evaluation* process serves to limit the number of new features constructed. The evaluation criterion we use is the information gained when an individual feature is used to partition the entire training set. For this experiment we limit the number of constructed features to 320 at any given time. Since there are 60 original features, the total number of features never exceeds 380. When the number of features is restricted in this manner, CITRE can be thought of as performing a beam search through the space of constructed features. Note that 320 features is the number necessary to represent each codon in each position in the input window.

In order to evaluate the ability of CITRE to construct useful features, we test the feature-construction process starting only with nucleotides as features. In this experiment we use fixed-size training sets of 5,000 examples.

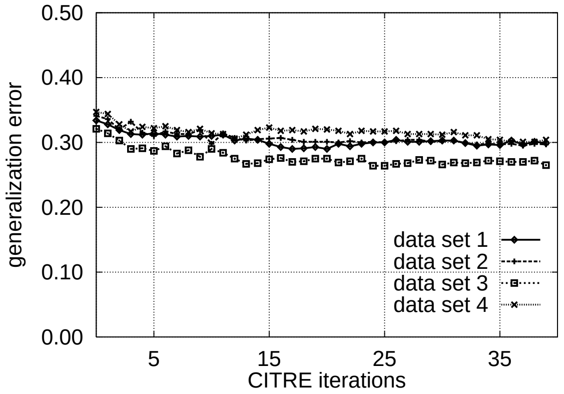

Figure 7: **Test set error for the CITRE trees.** The initial set of features is simply the *nucleotides* representation. The x-axis represents the number of feature-construction iterations, and the y-axis represents the test set error for a decision tree induced using the features of a given iteration. Fixed-size training sets of 5,000 examples are used to generate the trees.

The training and test sets are the same 5,000-example sets used in the previous experiment. We run CITRE for 40 iterations on each training set.

Figure 7 shows the test set error for each of the four training sets on each iteration of CITRE. From this figure it can be seen that the generalization performance fluctuates slightly as the feature set changes. Over the course of the feature-construction process, however, test set accuracies do improve somewhat. Table 1 shows the generalization performance of the CITRE-induced trees relative to the performance of C4.5 trees using both representations. The numbers in this table represent averages over all four folds of data. The CITRE error rate was determined by taking the tree from the iteration with the *best* performance for each data set. Thus, the error rate for the CITRE approach is a lower bound on what it would be if we used, say, a tuning set to decide when to stop iterating. All differences in this table (except CITRE vs. the *nucleotides* perceptrons, and CITRE vs. 40-hidden unit networks) are significant to at least the 0.025 level using a paired, 1-tailed t-test.

The results of this experiment indicate that CITRE, when given no domain knowledge, is able to improve upon the performance of a decision tree using only the *nucleotides* representation, but that this improvement is not enough to match the performance of trees using the *codons* representation. We have also conducted this experiment using the *fringe* feature selection method [Pagallo and Haussler, 1990], and *competitive evaluation* of features [Matheus, 1990]. However, we found that the *adjacent* selection method and information-based evaluation provided the best results. Further experimentation needs to be conducted to explore the effects of using domain knowledge, feature-generalization operators, and more iterations of feature construction.

Table 1: **Generalization error for decision tree and neural network approaches.** Reported values are averages for four training sets of 5,000 examples each. The CITRE result is an average of the values from the iteration with the *best* performance for each data set.

approach	% error
C4.5 with *nucleotides*	33.59
CITRE starting with *nucleotides*	28.75
C4.5 with *codons*	25.60
perceptron with *nucleotides*	29.53
40 HU network with *nucleotides*	27.95
perceptron with *codons*	22.56

7 Conclusions

We have described a real-world machine learning task that we believe serves as an interesting challenge problem for researchers investigating constructive induction. The problem is an interesting one for several reasons:

- it embodies the complexity inherent in an important real-world problem,
- two different, yet natural, input representations result in significantly different generalization performance for neural networks and decision trees trained using them,
- the relationship between the two representations is well known,
- the representational shift involved in constructing the better representation is not too imposing.

We have presented experiments that demonstrate that the *nucleotides* and *codons* representations result in significantly different test set accuracies for both perceptrons and decision trees. Additionally, we have presented experiments in which we tested the ability of both neural and symbolic constructive induction methods to bridge the gap between the two representations. These experiments indicate that the two approaches do not easily find the constructed features necessary to approach the performance of the *codons* representation.

The next question to be addressed in our research is how much domain knowledge is required in order to match the performance of the *codons* representation? One obvious piece of domain knowledge that could be exploited is information about the sequential nature of DNA. For example, in a neural network we could connect hidden units so that each one is connected only to a spatially-local part of the input window. Similarly, we could bias CITRE so that conjunctions of neighboring nucleotides are preferred. Although we know the mapping between nucleotides and codons, it is important to find feature-construction methods which are as general as possible so that they can be extended to problems in which a good representation is not known.

The challenge that we issue to the machine learning community is to develop an approach that is able to achieve the performance of the *codons* representation using the *nucleotides* representation, and as little domain knowledge as possible.[3] We believe that this problem can help guide research in constructive induction towards its goal of finding effective domain-independent biases for feature construction.

Acknowledgements

Insightful comments by Chris Matheus contributed to this research and its presentation. Dave Opitz provided helpful comments on an earlier version of this paper.

References

[Baum and Haussler, 1989] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[Craven and Shavlik, 1993] M. W. Craven and J. W. Shavlik. Learning to predict reading frames in *E. coli* DNA sequences. In *Proc. of the 26th Hawaii International Conf. on System Sciences*, pages 773–782, Wailea, HI, 1993. IEEE Press.

[Farber *et al.*, 1992] R. Farber, A. Lapedes, and K. Sirotkin. Determination of eucaryotic protein coding regions using neural networks and information theory. *Journal of Molecular Biology*, 226:471–479, 1992.

[Hinton, 1989] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[Matheus and Rendell, 1989] C. J. Matheus and L. A. Rendell. Constructive induction on decision trees. In *Proc. of the Eleventh International Joint Conf. on Artificial Intelligence*, pages 645–650, Detroit, MI, August 1989.

[Matheus, 1990] C. J. Matheus. *Feature Construction: An Analytic Framework and an Application to Decision Trees.* PhD thesis, University of Illinois at Urbana-Champaign, 1990.

[Michalski, 1983] R. S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[Pagallo and Haussler, 1990] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.

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

[Rosenblatt, 1958] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 1958.

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

[3] We plan to make this data set publicly available through the UC-Irvine Repository of Machine Learning Databases and Domain Theories. This database may be accessed by doing an anonymous ftp to ftp.ics.uci.edu.