A Generalized Hidden Markov Model for the Recognition of Human Genes in DNA

David Kulp David Haussler
Baskin Center for Computer Engineering and Information Sciences
University of California
Santa Cruz, CA 95064
{dkulp,haussler}@cse.ucsc.edu

Martin G. Reese Frank H. Eeckman
Genome Informatics Group
Lawrence Berkeley National Laboratory
1 Cyclotron Road
Berkeley, CA 94720
{martinr,eeckman}@genome.lbl.gov

Abstract

We present a statistical model of genes in DNA. A Generalized Hidden Markov Model (GHMM) provides the framework for describing the grammar of a legal parse of a DNA sequence (Stormo & Haussler 1994). Probabilities are assigned to transitions between states in the GHMM and to the generation of each nucleotide base given a particular state. Machine learning techniques are applied to optimize these probabilities using a standardized training set. Given a new candidate sequence, the best parse is deduced from the model using a dynamic programming algorithm to identify the path through the model with maximum probability. The GHMM is flexible and modular, so new sensors and additional states can be inserted easily. In addition, it provides simple solutions for integrating cardinality constraints, reading frame constraints, "indels", and homology searching.

The description and results of an implementation of such a gene-finding model, called Genie, is presented. The exon sensor is a codon frequency model conditioned on windowed nucleotide frequency and the preceding codon. Two neural networks are used, as in (Brunak, Engelbrecht, & Knudsen 1991), for splice site prediction. We show that this simple model performs quite well. For a cross-validated standard test set of 304 genes [ftp://www-hgc.lbl.gov/pub/genesets] in human DNA, our gene-finding system identified up to 85% of protein-coding bases correctly with a specificity of 80%. 58% of exons were exactly identified with a specificity of 51%. Genie is shown to perform favorably compared with several other gene-finding systems.

Introduction

Genomic DNA from human and model organisms is being sequenced at an exponentially increasing rate, making it all the more important to have the right tools for analyzing and annotating such sequences. It is particularly useful to identify coding regions from which one can deduce the structure of genes and the resulting proteins. Over the past decade, a large body of research has accumulated that deals with the recognition of translational and transcriptional features. Functional sites and regions include promoter, start codon, splice sites, stop codon, 3' and 5' untranslated regions, introns, and initial, internal and terminal exons. Research historically could be categorized as either statistical or homology based, and most research until recently aimed to characterize a single feature. Fickett(Fickett & Tung 1992) provides an overview and evaluation of many statistical measures for signal and content sensors. Recently, gene-finding systems have been developed that employ many of the known recognition techniques in concert. Current state-of-the-art gene-finding methods combine multiple statistical measures with database homology searching to identify gene features (see, for example, FGENEH (Solovyev, A., & Lawrence 1994), GRAILII (Xu *et al.* 1994), GenLang (Dong & Searls 1994), GENMARK (Borodovsky & McIninch 1993), and GeneID (Guigo *et al.* 1992)). The development of gene-finding systems raises research questions regarding the effective and efficient implementation of the system separate from the efficacy of its components. In this paper, we present the results of the implementation of a gene-finding system as a Generalized Hidden Markov Model. Our system is similar in design to GeneParser (Snyder & Stormo 1993), but is based on a rigorous probabilistic framework. We show how a GHMM offers a simple elegant model of genes in eukaryotic DNA. The probabilistic framework provides meaningful answers (in a probabilistic sense) to the problem of predicting a complete gene structure or individual components. We present an implementation that is efficient in both time and space, and is general and flexible enough so as to facilitate a modular approach to the use of sensors. We show that an implementation with fairly simple sensors performs as well as the better published gene-finding systems when compared against a standard test set.

Methods

System Framework

Hidden Markov Models have been used for decades in pattern recognition (Rabiner & Juang 1986). More recently, their applicability to computational biology has gained recognition, see e.g. (Krogh *et al.* 1994). In (Krogh, Mian, & Haussler 1994), an HMM was built for identifying gene structure in *E. coli*. HMMs have been generalized to allow one state in the model to generate more than one symbol (Stormo & Haussler 1994). This generalized framework separates the overall structure of the HMM from the embedded component submodels. Generalized HMMs provide an intuitive framework for representing genes with their various functional features, and efficient algorithms can be built to use such models to recognize genes.

Figure 1 shows a simplified GHMM for multiple exon genes. Arcs correspond to states in the state machine and the nodes represent transitions between states. The labeled states are `J5'` - 5' untranslated region; `J3'` - 3' non-coding, `EI` - Initial Exon; `E` - Internal Exon; `I` - Intron; `EF` - Final Exon; `ES` - Single Exon. Nodes correspond to signals: `D` - Donor site, `A` – Acceptor site, `S` – Start Translation; `T` – Terminate Translation. `B` (Begin) and `F` (Finish) are special source and sink nodes, respectively, for the entire graph. We conceptualize the GHMM as a machine in which each state generates zero or more symbols. Given a candidate DNA sequence, X, we define the predicted gene structure as the ordered set of states, ϕ, called the parse, such that the probability of generating X according to ϕ is maximal over all possible parses.

To formalize these concepts, we first define a standard hidden Markov model in which each state generates a single symbol. We then generalize this model to accommodate multiple symbols per state. Let

$$M = \text{model} \tag{1}$$

$$X = \{X[1], \ldots, X[n]\} \tag{2}$$

$$\phi = \{q_1, q_2, \ldots, q_n\} \tag{3}$$

where $X[i]$ is the ith base in the sequence X of length n and q_i is the ith state in the parse ϕ. We require q_1 to be an arc leaving the Begin node `B` and and q_n to be an arc leading to the Finish node `F`. The parameters of M specify for each node a probability distribution over the arcs leaving that node, and for each arc (state), a probability distribution over strings that are generated by that state.

To parse X we find ϕ to maximize $P(X, \phi|M)$. In a standard hidden Markov model, we can write this probability as the independent joint probabilities of transitioning to each state and generating the base $X[i]$ in state q_i. It is implicitly conditional on M. So, we have (see Rabiner (Rabiner & Juang 1986))

$$P(X,\phi) = P(q_1|\mathtt{B}) \left(\prod_{i=1}^{n} P(X[i]|q_i) \right) \left(\prod_{i=1}^{n-1} P(q_{i+1}|\mathrm{n}ode(q_i)) \right), \tag{4}$$

where $\mathrm{n}ode(q_i)$ is the node that the arc q_i leads to. The generalization of HMMs to accommodate the generation of multiple symbols per state just introduces an ordered set of subsequences of X, $\{x_1, x_2, \ldots, x_k\}$, such that

$$X = x_1 x_2 \ldots x_k \text{ (the concatenation of subsequences)}$$

and a redefinition of the parse as an ordered set of state/sequence pairs:

$$\phi = \{(q_1, x_1), (q_2, x_2), \ldots, (q_k, x_k)\}.$$

Then equation 4 can be generalized (Fong 1995; Auger & Lawrence 1989; Sankoff 1992; Bengio 1996) as

$$P(X,\phi) = P(q_1|\mathtt{B}) \left(\prod_{i=1}^{k} P(x_i|q_i) \right) \left(\prod_{i=1}^{k-1} P(q_{i+1}|\mathrm{n}ode(q_i)) \right). \tag{5}$$

Each term $P(x_i|q_i)$ can be further decomposed using

$$P(x_i|q_i) = P(x_i|l(x_i), q_i) P(l(x_i)|q_i)$$

where $l(x_i)$ is the length of the subsequence x_i.

Each term, $P(x_i|l(x_i), q_i)P(l(x_i)|q_i)$, can be described conceptually as a "content sensor" that returns a probability of generating x_i according to the model of state q_i. Note that a state's model can be arbitrarily complex and might be, itself, an HMM or GHMM. Our framework allows us to define an abstract system-level relationship independent of sensor implementation details. The "optimal" parse of X is defined as the parse ϕ that maximizes $P(X, \phi)$. GHMMs are strictly more powerful than standard HMMs. For example, in a GHMM the length distribution $P(l(x_i)|q_i)$ can be defined by arbitrary histograms, whereas in standard HMMs they are simple geometric distributions.

Gene Structure Constraints

Cardinality Constraints By generalizing HMMs as described in the previous section, we are able to replace the implicit geometric distribution on the lengths of features with an arbitrary distribution. We wish to describe the distribution of *occurrences* of a feature in a similar manner. For example, given the simplified GHMM in figure 1, and a fixed probability for $P(\mathtt{E}|\mathtt{A})$ – i.e., the transition probability from an acceptor to an internal exon – then the distribution of the number of internal exons is geometric over $P(\mathtt{E}|\mathtt{A})$. Experimental evidence indicates that the number of exons in a gene is not geometric (Hawkins 1988)(Smith 1988). Hence we would like to impose an arbitrary distribution constraint on the "cardinality" of exons (Wu 1995). The solution requires the removal of all cycles in the GHMM by virtually "unspooling" the graph. Figure 2 shows the unspooled version of figure 1. The transition probabilities $P(\mathtt{E}_{i+1}|\mathtt{A}_i)$ can be arbitrarily assigned to each state transition i either through a learning process or from experimental evidence such as frequency counts.

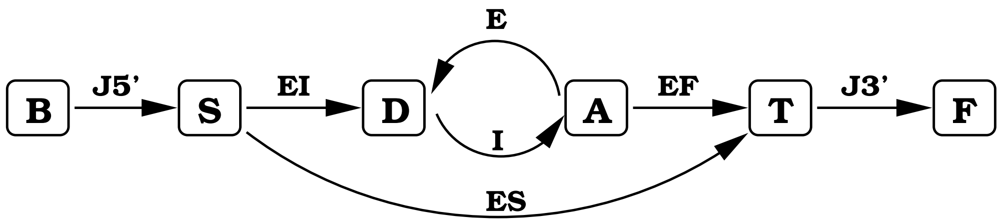

Figure 1: A simple GHMM for a sequence containing a multiple exon gene. The arcs represent multi-symbol states and nodes represent transitions between states. The arrows imply a generation of bases from 5' to 3'.

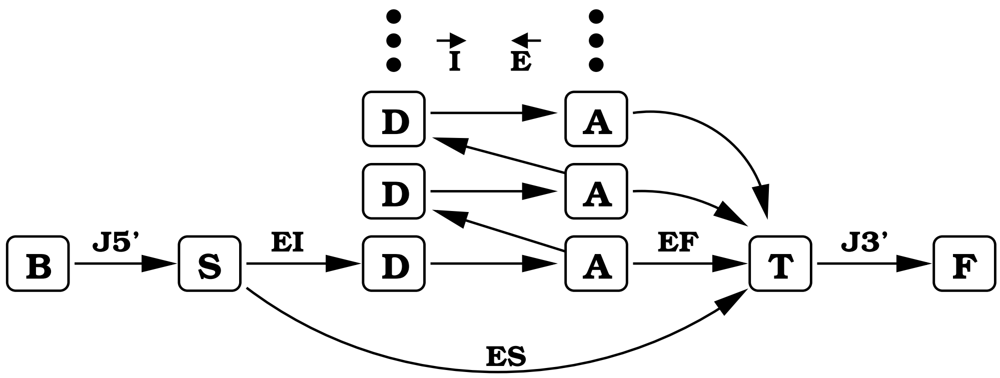

Figure 2: Shows the virtual model of an unspooled GHMM. Transition probabilities can be assigned at each transition node. An arbitrary number of internal exon and intron states can be added.

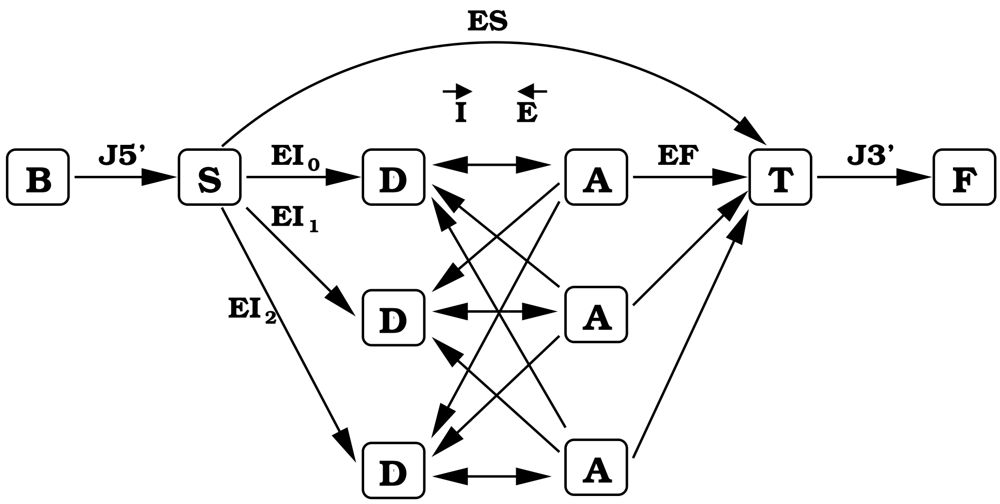

Figure 3: A GHMM including frame constraints. The additional acceptor and donor transition nodes ensure that only syntactically correct parses are considered.

Frame Constraints The problem of maintaining a correct reading frame can be solved by adding additional states to the GHMM. Thirteen states are added to the state machine such that no legal parse is allowed that does not maintain a correct reading frame. Figure 3 shows a modified version of figure 1 that ensures correct reading frame. Three donor sites and three acceptor sites represent the nine possible ORFs. We can say that introns retain frame, and so there is only one intron state exiting from each of the three donor site nodes. But, an exon can change frame depending on the length of the exon, and so there are three possible exon states exiting from each of the three acceptor sites. Note, however, that the first base of the initial exon and the last base of the final exon must always be in the same reading frame. Therefore, there is an effective source and sink at the `S` (Start Translation) and `F` (Terminate Translation) transitions. By placing reading frame length restrictions on each exon state, we ensure that only valid parses can be generated. For example, the states EI_0, EI_1, and EI_2 require that the subsequence must be of a length equal to 0, 1, and 2, modulo 3, respectively. That is, the content sensor associated with each state would return a non-zero probability only if this condition is met. In this way, frame constraint is built into the system framework. No additional mechanism is needed to selectively eliminate parses with incorrect reading frames.

Signal Sensors and Consensus Constraints A signal sensor is usually implemented as a statistical discriminant function or a neural network that returns a posterior probability of a functional site given a fixed-length subsequence at or surrounding the site. Within a parse, for each x_i and x_{i+1} we define a transition t_i between the two states q_i and q_{i+1}. The transition t_i represents a signal, e.g. a donor site between an exon x_i and an intron x_{i+1}. The location of the fixed-length functional site x'_i partially overlaps zero or more bases of x_i and x_{i+1} as shown in figure 4. A signal sensor returns a posterior probability of the form $P(t_i|x'_i)$. A consensus constraint is a restriction imposed by the model on the allowable symbols at relative positions with respect to a particular state. The dinucleotide consensus found at acceptor and donor sites is an example of a consensus constraint. Such constraints are often part of the model of the signal sensors of functional sites. These constraints are implemented within the probabilistic framework by ensuring that non-zero posterior probabilities are returned only for those sites that agree with consensus constraints. The simplest signal sensor returns the frequency of the signal in the training set for all sites that agree with the consensus.

Integrating Signal Sensors Figure 4 shows the regions and sensors corresponding to two adjacent subsequences x_i and x_{i+1}. In order to correctly compute the value of equation 5, it is necessary to convert the posterior probability $P(t_i|x'_i)$ returned by the signal sensor into the likelihood $P(x'_i|t_i)$. By Bayes Rule, we have

$$P(x'_i|t_i) = P(t_i|x'_i)P(x'_i)/P(t_i) \tag{6}$$

Let $\neg t_i$ be the "local null model" for a transition site used when the signal sensor was trained to discriminate true signals from non-signals. Using equation 6 twice we have the ratio

$$\frac{P(x'_i|t_i)}{P(x'_i|\neg t_i)} = \frac{P(t_i|x'_i)P(\neg t_i)}{P(\neg t_i|x'_i)P(t_i)} = \frac{P(t_i|x'_i)(1-P(t_i))}{(1-P(t_i|x'_i))P(t_i)} \tag{7}$$

Hence,

$$P(x'_i|t_i) = \frac{P(t_i|x'_i)(1-P(t_i))}{(1-P(t_i|x'_i))P(t_i)}P(x'_i|\neg t_i) \tag{8}$$

Here, $P(t_i|x'_i)$ is the posterior probability output from a signal sensor, and $P(t_i)$ can be interpreted as the observed frequency of t_i in the training set used to train the signal sensor. The term $P(x'_i|\neg t_i)$ is sometimes problematic, since it is often not clear what null model is being implicitly used in many discriminative training methods, such as neural network methods. In these cases it must be estimated. For example, for donor sites we use a simple model in which all letters of x'_i are independent and distributed according to the frequencies of nucleotides in a local window, except the consensus pattern 'GT' which is required. This is because the neural network we use to recognize donor sites was trained with negative examples with the consensus 'GT', but were otherwise random non-donor sites.

Once we have computed the likelihood $P(x'_i|t_i)$, we need to integrate this value into the calculation of the overall joint likelihood $P(X, \phi)$. Referring to figure 4, we see that in the absence of the signal sensor for t_i, the likelihood for this part of the parse would contain the term $P(x_i|q_i)P(x_{i+1}|q_{i+1})$. With the output $P(x'_i|t_i)$ from the signal sensor, this part of the likelihood can be refined to

$$P(x_{ab}|q_i)P(x'_i|t_i)P(x_{de}|q_{i+1})$$

where x_{ab} and x_{de} are the parts of x_i and x_{i+1}, respectively, not overlapped by x_i.

Note, in the extreme case the signal sensor returns probability 0 for the transition t_i from state q_i to q_{i+1}, $P(x'_i|t_i) = 0$, and hence the refined likelihood of the parse drops to zero. This is how consensus constraints are enforced by the probabilistic mechanism.

Correcting for Insertions and Deletions

Insertion and deletion of nucleotides ("indels") introduced by sequencing errors need to be corrected before applying the frame constraint. The system described in this paper does not explicitly address these errors at the GHMM level; rather this is left to the exon content sensors, where the problem can be dealt with using varying degrees of sophistication. As discussed in the previous section, the frame constraint places absolute

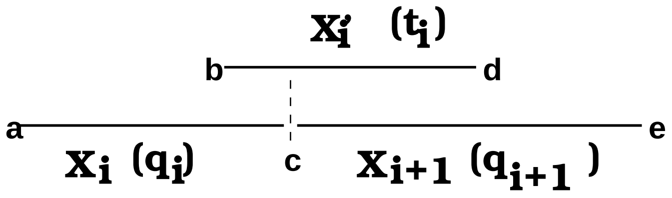

Figure 4: The location of two hypothetical content sensor regions, x_i and x_{i+1}, and the signal sensor region x_i' representing the transition t_i. The transition occurs at position c with overlap from position b to d.

length restrictions on exons. Alternatively, probabilities can be assigned to subsequences whose length, modulo 3, do not match the required frame for a particular state. These probabilities can be easily derived from statistical estimation of indel frequencies. A more complicated approach models an exon as a GHMM. This model includes an insertion and/or deletion state with a small probability of transitioning into this state, as in the codon models of Krogh et al (Krogh *et al.* 1994).

Dynamic Program for Optimizing Parse

The Viterbi algorithm is used to maximize equation 5 for ϕ. This approach is well described elsewhere including (Snyder & Stormo 1993; Gelfand & Roytberg 1993; Auger & Lawrence 1989; Sankoff 1992; Gelfand & Roytberg 1995; Bengio 1996). Notable differences from the standard dynamic programming algorithm relate to accommodating the GHMM framework. Specifically, a first pass through the sequence establishes candidate transition sites and constructs a graph of the syntactically legal parses. With the addition of multi-symbol states, the DP algorithm must iterate through all transitions t_i, for $1 \leq i \leq k$, considering all legal preceding states from each possible transition t_j, for $j < i$. This implies that the running time is $O(k^2)$ where k is the number of possible transitions. Experimental evidence shows that $k \approx \frac{n}{100}$, where n is the total number of bases. The running time can be further reduced by imposing maximum length restrictions on certain states. For example, no exon region longer than 3,000 bases is considered in our implementation. If all states include maximum length restrictions, then the asymptotic running time becomes linear in k for large n.

The graph can be stored such that each transition node requires a number of pointers equal to the (constant) number of possible states that can legally precede it. As a result, the space required to store the graph is also linear in k – the number of nodes in the graph. The algorithm will scale well to accommodate large sequences as contiguous DNA on the order of 100Kb become available.

An advantage of the GHMM gene model is the ability to calculate the probability of a particular feature by using a dynamic program to sum over all possible parses with that feature. Suppose, for example, we wish to determine the probability that some subsequence x is an exon given the context of the full sequence X. As described under "System Framework", let $\mathtt{E}$ be the exon state, then formally, we wish to find $P((x, \mathtt{E}) \in \phi | X, M)$. This requires that we sum $P(\phi|X, M)$ over all possible parses, ϕ, that contain the pair $(x, \mathtt{E})$. To efficiently calculate this probability, a forward-backward algorithm is employed (Stormo & Haussler 1994). Additionally, the best parse given the feature, $(x, \mathtt{E})$, i.e. $\arg\max \phi P(\phi | M, X, (x, \mathtt{E}) \in \phi)$, can be simply deduced by applying a variation of the Viterbi algorithm which processes the two half-sequences on either side of x independently.

Implementation

A working system built according to the model and design described here was implemented and experimentally validated. Genie depends to a large extent on the quality of its individual content and signal sensors. Each component is designed and trained independently, and then combined into a modular system. More sophisticated training methods, e.g. like those used with hidden Markov models, can also be employed (Rabiner & Juang 1986; Stormo & Haussler 1994; Bengio 1996). We describe briefly the key points in the current implementation of Genie.

Length Distributions In a GHMM individual states can generate multi-symbol strings based on arbitrary length distributions. In our implementation, the state-specific length distributions were found by generating a length histogram for each state. Figure 5 shows the smoothed and normalized distributions derived from the first training set for introns and internal exons.

Splice Site Model Two neural network recognizers were developed as described in (Brunak, Engelbrecht, & Knudsen 1991). We trained a backpropagation feed-forward neural network with one layer of hidden units to recognize donor and acceptor sites, respectively. Different from Brunak et al., we only consider genes that had constraint consensus splice sites, i.e., 'GT' for the donor and 'AG' for the acceptor site. Hence, the neural network distinguishes between GT donor sites (AG ac-

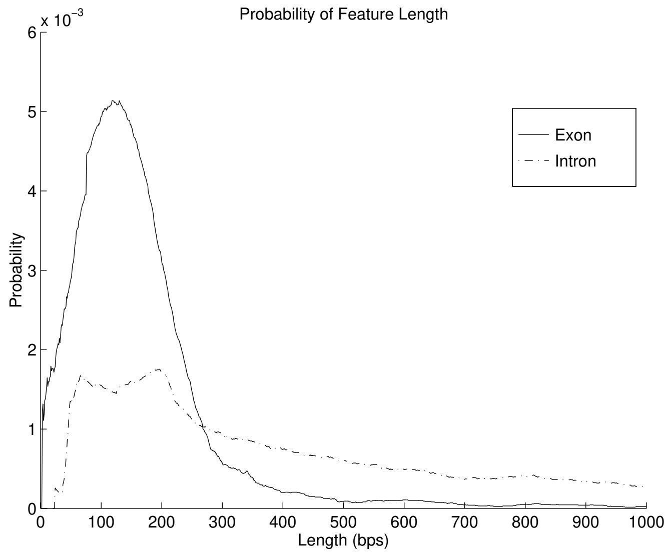

Figure 5: Probability distributions of the length of introns and internal exons.

ceptor sites) occurring in the DNA sequence that function as splice sites and those that do not. To achieve that goal, the sequence is coded using 4 input units for each nucleotide and one unit as the output. Empirical experiments similar to (Brunak, Engelbrecht, & Knudsen 1991) show that sequence window sizes of 15bp for donor sites (-7..+8) and 41bp for acceptor sites (-21..+20) are optimal. In addition the number of hidden units was experimentally optimized. The best results are achieved with 50 hidden units for donor and 40 hidden units for acceptor sites. Additional hidden layers do not improve the results. It is interesting to note that the number of hidden units do not seem to play an important role. For example, the Correlation Coefficient for donor site prediction in a network with 50 hidden units is 0.855 whereas in a network with no hidden units it is 0.81. The output of the two networks are interpreted as the posterior probabilities for donor and acceptor sites at a given position in the sequence.

Intron Model The intron model is essentially a windowed null model. For any base b at position i, the frequency of nucleotides in a window of 300 bases, from $i-150$ to $i+150$ excluding position i, is computed. The probability of b is assigned according to the computed frequencies. The current implementation does not include any sophisticated knowledge of introns, such as repeat detection. Intuition suggests that those features peculiar to introns, such as repeats, do not have high coding potential, so a good exon model will be unlikely to favor such regions.

Exon Model The exon model uses only two coding statistics to determine coding potential. First, GC-content and any other local frequency bias is considered by computing the frequency of the four nucleotides within a window of 300 bases, similar to the intron model. The size of the window was chosen experimentally. For larger window sizes, local variation in base composition was less evident. Second, a first-order Markov chain is used to condition the distribution over the 61 possible codons. These criteria are combined as feature input into a 2-layer neural network with 17 hidden units, trained using standard backpropagation. (The number of hidden units were experimentally optimized, and hidden units were found to have only a marginal effect.) Hence, the GC-content, codon usage, and previous codon are simply integrated in a single discriminator.

Results

For our studies we built a representative human gene data set using Genbank, release 89, 1995. The human gene set was selected from all known human genes in Genbank. To obtain a representative set we preprocessed the data using several filters. We required a correct species label, i.e. “Homo sapiens”, and at least one intron in the sequence. A valid CDS annotation must exist; coding must begin with “atg” and finish with one of three consensus stop codons, and splice sites must conform to the consensus dinucleotides. Sequences with alternative splicings and in-frame stop codons were discarded. Additionally, sequences were discarded if the sequence identity of the translated protein was greater than 50% using BLAST. The resultant data set of 304 genes was divided into seven groups to be used in cross-validation – one seventh of the data is used for testing. This data set (in Genbank flatfile format) is publicly available via anonymous FTP from www-hgc.lbl.gov in directory /pub/genesets/.

For comparison with other gene-finding systems, we also tested Genie against a second data set, provided by Burset and Guigo (Burset & Guigo 1996). This data set of 570 genes from many different organisms was used in (Burset & Guigo 1996) to compare the effectiveness of many different gene-finders. Our system, like most of those tested in (Burset & Guigo 1996), was trained on human genes only, but it is still interesting to compare the relative predictive ability among the systems.

Table 1 shows statistical results from tests of the gene-finder against two (arbitrarily chosen) of the seven test sets using the 304-gene data set. We also tested Genie against the Burset/Guigo data set; results comparing our gene-finder with other gene-finding systems is shown in Table 2. In accordance with the testing scheme established by Burset and Guigo, we report sensitivity and specificity with respect to per-base prediction of coding/non-coding and with respect to exact prediction of exons. The per-base sensitivity is the fraction of true coding bases predicted as coding, and the specificity is the fraction of all predicted coding bases that were correct. Similarly, the exon sensitivity is the fraction of true exons predicted exactly, and the specificity is the fraction of predicted exons that were correct. In these tests, correct exon prediction requires identification of the exact position of splice sites. Fully or partially overlapping predictions are not accepted. The approximate coefficient (AC) is described by (Burset & Guigo 1996) as a preferred alternative over the correlation coefficient and defined by

$$AC = \frac{1}{2}\left(\frac{\mathtt{TP}}{\mathtt{TP+FN}} + \frac{\mathtt{TP}}{\mathtt{TP+FP}} + \frac{\mathtt{TN}}{\mathtt{TN+FP}} + \frac{\mathtt{TN}}{\mathtt{TN+FN}}\right) - 1$$

where TP, FP, TN, and FN are true positives, false positives, true negatives, and false negatives.

In addition, we also report the fraction of true exons that were not identified either exactly or overlapping (Missing Exons) and the fraction of predicted exons that did not overlap any true exon (Wrong Exons).

Discussion

The predictive ability of our gene-finder is shown to be as good as other gene-finding systems. In particular, in comparisons using the Burset/Guigo data set, Genie's performance is comparable to that of GenLang (Dong & Searls 1994), which was the second best program for predicting exact exons among those tested. This is encouraging, since Genie is based on a rather simple probabilistic framework. However, a short-coming of the current implementation seems to be the proclivity to predict extraneous exons. Although up to 93% of true exons are identified, at least 29% of the total predictions do not overlap any known coding region. Observations suggest that the length of these predicted regions were often relatively small. Attempts to improve the specificity of exon prediction by artificially adjusting model parameters have not yet shown good results. In this regard, there is still much room for improvement.

Our research is currently focused on integrating homology-based searching into our GHMM gene model. We consider one of the most important advantages of homology-based discrimination to be the ability to identify exon pairs, thus implying the exact location of splice sites. Therefore, a key feature in our proposed database model is the introduction of a "splice junction" sensor – a fixed-length sensor that identifies database matches from a putative splice. The second component is a new exon sensor as a linear HMM. The HMM is built on-the-fly for each candidate exon and includes states for each database match. A database is interpreted very generally and includes protein motifs and collections of cDNA, DNA, and amino acid sequences.

Adding homology searching complicates the probabilistic interpretation of the parse. We consider a database match in an information theoretic sense as a bit cost for encoding the unique identification of the match. The probability of a match can then be derived from the encoding cost and integrated into the joint probability of the complete parse.

Additional current work includes designing a graphical interface for use by biologists at large-scale sequencing centers such as Lawrence Berkeley National Laboratory, incorporating a promoter signal sensor (Reese 1995), and providing multiple gene recognition capability. We hope to report results regarding these enhancements by the time of the conference.

Acknowledgments

We wish to thank Gary Stormo for his supportive advice and comments throughout the design and development of this work. Thanks is also extended to Gregg Helt for early work on the gene data set and Nomi Harris for her graphical interface, which proved helpful to study individual predictions. We gratefully acknowledge Saira Mian, Michael Brown, Leslie Grate, Richard Hughey and Kevin Karplus for helpful comments and discussions on the topic.

This work was supported in part by DOE grant no. DE-FG03-95ER62112 and DE-AC03-76SF00098. D. Haussler acknowledges support of the Aspen Center for Physics, Biosequence Analysis Workshop.

Data Set	*Per Base*			*Exact Exon*				
	Sn	Sp	AC	Sn	Sp	Avg	ME	WE
Part 1	0.85	0.80	0.80	0.58	0.51	0.54	0.07	0.29
Part 2	0.70	0.69	0.64	0.49	0.41	0.45	0.23	0.39

Table 1: Sensitivity (Sn), Specificity (Sp), Approximate Coefficient (AC), Average of Sensitivity and Specificity (Avg), Missing Exons (ME), and Wrong Exons (WE) as measured for two parts of a cross-validated test set from a data set of 304 human genes in DNA.

Gene-finder	*Per Base*			*Exact Exon*				
	Sn	Sp	AC	Sn	Sp	Avg	ME	WE
Genie	0.76	0.77	0.72	0.55	0.48	0.51	0.17	0.33
FGENEH	0.77	0.85	0.78	0.61	0.61	0.61	0.15	0.11
GeneID	0.63	0.81	0.67	0.44	0.45	0.45	0.28	0.24
GeneParser2	0.66	0.79	0.66	0.35	0.39	0.37	0.29	0.17
GenLang	0.72	0.75	0.69	0.50	0.49	0.50	0.21	0.21
GRAILII	0.72	0.84	0.75	0.36	0.41	0.38	0.25	0.10
SORFIND	0.71	0.85	0.73	0.42	0.47	0.45	0.24	0.14
Xpound	0.61	0.82	0.68	0.15	0.17	0.16	0.32	0.13

Table 2: A comparison of Genie with other gene-finding systems. Tests were run on a set of 570 annotated sequence from different organisms.

References

Auger, I. E., and Lawrence, C. E. 1989. Algorithms for the optimal identification of segment neighborhoods. *Bull. Math. Biol.* 51:39–54.

Bengio, Y. 1996. *Neural Networks for Speech and Sequence Recognition.* Thomson.

Borodovsky, M., and McIninch, J. 1993. Genmark: Parallel gene recognition for both DNA strands. *Computers and Chemistry* 17(2):123–133.

Brunak, S.; Engelbrecht, J.; and Knudsen, S. 1991. Prediction of human mRNA donor and acceptor sites from the dna sequence. *JMB* 220:49–65.

Burset, M., and Guigo, R. 1996. Evaluation of gene structure prediction programs. *Genomics (to appear).* Data set and evaluation results can be found at http://www.imim.es/GeneIdentification-/Evaluation/Index.html.

Dong, S., and Searls, D. B. 1994. Gene structure prediction by linguistic methods. *Genomics* 162:705–708.

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

Fong, D. 1995. Parsing of DNA sequences using dynamic programming with a state machine model. unpublished manuscript.

Gelfand, M. S., and Roytberg, M. A. 1993. Prediction of the exon-intron structure by a dynamic programming approach. *BioSystems* 30:173–182.

Gelfand, M. S., and Roytberg, M. A. 1995. Dynamic programming for gene recognition. In *Gene-Finding and Gene Structure Prediction Workshop.*

Guigo, R.; Knudsen, S.; Drake, N.; and Smith, T. 1992. Prediction of gene structure. *J. Mol. Biol.* 226:141–157.

Hawkins, J. D. 1988. A survey on intron and exon lengths. *Nucl. Acids Res.* 16:9893–9908.

Krogh, A.; Brown, M.; Mian, I. S.; Sjölander, K.; and Haussler, D. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *JMB* 235:1501–1531.

Krogh, A.; Mian, I. S.; and Haussler, D. 1994. A Hidden Markov Model that finds genes in *E. coli* DNA. *NAR* 22:4768–4778.

Rabiner, L. R., and Juang, B. H. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3(1):4–16.

Reese, M. G. 1995. Novel neural network prediction systems for human promoters and splice sites. In *Gene-Finding and Gene Structure Prediction Workshop.*

Sankoff, D. 1992. Efficient optimal decomposition of a sequence into disjoint regions, each matched to some template in an inventory. *Math. Biosci.* 111:279–293.

Smith, M. W. 1988. Structure of vertebrate genes: a statistical analysis implicating selection. *J. Mol. Evol.* 27:45–55.

Snyder, E. E., and Stormo, G. D. 1993. Identification of coding regions in genomic DNA sequences:

an application of dynamic programming and neural networks. *Nucl. Acids Res.* 21:607–613.

Solovyev, V.; A., S.; and Lawrence, C. 1994. Predicting internal exons by oligonucleotide composition and discriminant analysis of splicable open reading frames. *Nucl. Acids Res.* 22:5156–5163.

Stormo, G. D., and Haussler, D. 1994. Optimally parsing a sequence into different classes based on multiple types of information. In *ISMB-94*. Menlo Park, CA: AAAI/MIT Press.

Wu, T. 1995. A phase-specific dynamic programming algorithm for parsing gene structure. In *Gene-Finding and Gene Structure Prediction Workshop*.

Xu, Y.; Einstein, J. R.; Shah, M.; and Uberbacher, E. C. 1994. An improved system for exon recognition and gene modeling in human dna sequences. In *ISMB-94*. Menlo Park, CA: AAAI/MIT Press.