Finding Genes in DNA with a Hidden Markov Model

John Henderson* Steven Salzberg† Kenneth H. Fasman‡

Abstract

This study describes a new Hidden Markov Model (HMM) system for segmenting uncharacterized genomic DNA sequences into exons, introns, and intergenic regions. Separate HMM modules were designed and trained for specific regions of DNA: exons, introns, intergenic regions, and splice sites. The models were then tied together to form a biologically feasible topology. The integrated HMM was trained further on a set of eukaryotic DNA sequences, and tested by using it to segment a separate set of sequences. The resulting HMM system, which is called VEIL (Viterbi Exon-Intron Locator), obtains an overall accuracy on test data of 92% of total bases correctly labelled, with a correlation coefficient of 0.73. Using the more stringent test of exact exon prediction, VEIL correctly located both ends of 53% of the coding exons, and 49% of the exons it predicts are exactly correct. These results compare favorably to the best previous results for gene structure prediction, and demonstrate the benefits of using HMMs for this problem.

1 Introduction

Robust computational solutions to the gene-finding problem are a valuable resource for the Human Genome Program and for the molecular biology community at large. Software that can reliably identify putative genes in DNA sequence can significantly speed up their discovery in the age of high-throughput genomic sequencing. A number of gene-finding systems have been developed in recent years, with varying degrees of success, but the problem still does not have a satisfactory solution. Many of these systems are still under development, and improvements continue to appear. Some of the leading systems are GRAIL [26], GeneID [10], GeneParser [23], SORFIND [11], and FGENEH [25].

*Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218. Email: jhndrsn@cs.jhu.edu.

†Corresponding author. Department of Computer Science and Division of Biomedical Information Sciences, Johns Hopkins University, Baltimore, MD 21218. Email: salzberg@cs.jhu.edu.

‡Genome Database, c/o Whitehead/MIT Center for Genome Research, One Kendall Square, Bldg. 300, Cambridge, MA 02139. Email: ken@gdb.org.

These systems use a variety of sophisticated computational techniques, including neural network algorithms, dynamic programming, rule-based methods, decision trees, and probabilistic reasoning. Although none of the systems perform perfectly, the information they provide is valuable enough that some of them are widely used to identify putative genes in uncharacterized DNA.

Hidden Markov Models (HMMs) provide a precise probabilistic method for modelling sequences of discrete data, and therefore seem to be a natural solution for the problem of finding new genes in DNA sequences. The VEIL (Viterbi Exon-Intron Locator) system is an HMM that incorporates several distinct, small HMMs that model information within exons, introns, intergenic regions, and splice junctions. We have developed a training regimen that allows us to tie together these smaller HMMs into a single gene-finding system.[a] VEIL uses the well-known EM algorithm [18] to train all its models, and once the training is complete, it uses the Viterbi algorithm [27] to parse a new sequence into its component exons and introns. Details of the models and the algorithms are explained below.

VEIL has been tested on a database of 570 vertebrate sequences that was collected by Burset and Guigo [5] specifically to test gene-finding systems. Our cross-validated results show that VEIL obtains a summary accuracy of 92% of bases correctly labelled, with a sensitivity of 74% and specificity of 72% for exonic regions, and a correlation coefficient of 0.68. Using the more stringent test of exact exon prediction, VEIL correctly located both ends of 46% of the exons. These results compare favorably with all the major gene structure prediction programs featured in previous studies. In addition, our results are carefully cross-validated, which makes it easy for future studies to make direct comparisons to this one. More details of the comparisons are given in Section 3.[b]

2 HMMs for gene finding

Hidden Markov Models have been remarkably successful in the field of speech recognition [1, 16], where they are used in most state-of-the-art systems. Biological sequences, like speech, can be modelled as the output of a process that progresses through a series of discrete states, some of which are "hidden" to the observer. HMMs excel at this type of modelling, and as a result, researchers in computational biology have recently begun to use them for analysis of DNA and protein sequences. HMMs have been used for

[a] To be precise, our system does not simply "find genes." Rather, it finds coding regions beginning with a start codon and ending with a stop codon. It does not find the beginning or end of transcription. All of the major gene-finding systems perform this same task. A more accurate description of this task might be "coding region finding," but "gene finding" has become standard usage.

[b] The VEIL system is available at ftp://ftp.cs.jhu.edu/pub/veil. This directory contains the executables and the database used in this study.

finding periodicities in DNA [2], for exploring structural similarities of families of genes [6], for producing multiple sequence alignments [13, 3], for finding palindromic repeats [12], and for protein secondary structure prediction [7, 4]. Krogh et al. [14] have used HMMs to find genes in *E. coli*, where the problem of introns does not arise. The VEIL system described herein demonstrates the use of HMMs to find complex gene structures in eukaryotic DNA sequences. Kulp et al. [15, 19] have also developed an HMM system for this task (Genie), using a generalized HMM architecture that is very different from VEIL's. Genie is a "generalized HMM" in which an edge can be a complete HMM in itself, and a state can be an arbitrary program. In their GHMM, states use neural networks for splice site recognition, making their system a neural network/HMM hybrid. Most of the HMMs produced in these computational biology projects have been relatively small in comparison to speech recognition systems, in part because of the limited amount of data available. Larger HMMs have many more free parameters and therefore require much more data for accurate training. These small models have nonetheless produced impressive results.

2.1 HMM basics

Although HMMs cannot be covered in detail here, a brief introduction will be useful. For more a detailed description, see Lee [16] or Rabiner [18]. An HMM models a process in which some of the details are unknown, or *hidden*. Typically this process is stochastic in nature. Most commonly, HMMs are used to model sequential data or processes, which could be a sequence of nucleotides, sounds (for speech processing), or any other discrete sequence. For this discussion, it is useful to think of an HMM as *generating* a sequence as output; however, it is just as easy to treat a sequence as input to an HMM. An important assumption behind any modeling with HMMs is the *Markov* assumption: the states that follow any state v in the model depend *only* on v, and are independent of all states preceding v. This independence assumption is essential for the computations used for training HMMs. We realize that this assumption does not hold for genomic DNA sequences, and will describe later how the architecture of the model can help reduce its impact.

An HMM is defined by a set of states and transitions, usually represented as a graph where states correspond to vertices and transitions to edges. Each state v is associated with a discrete output probability distribution, $P(b)$; for DNA, this output distribution is simply the probability that the HMM will generate each base $b \in \{A, C, G, T\}$. Similarly, each transition has a probability, which represents the probability that a generating process makes that transition. Thus the probabilities of all the transitions *leaving* a given state v must sum to 1.

Throughout this paper we will be using the terms *chain* and *stage* to indicate certain graph topologies. A chain is a set of n states (numbered $1..n$) such that there is an edge

from state i to state $i+1$ for each $i \in \{1, 2, \ldots, (n-1)\}$. Furthermore, no edges enter or leave any state in the chain other than the first and last states. (An example of a chain is shown in Figure 4.) A stage is a set of states that are aligned in the graph. Stages are indicated in Figure 3. Intuitively, the states that participate in a stage compete for the basepair that is aligned with the stage; i.e., a stage is a set of states that corresponds to a single position in a DNA sequence. A series of stages with no loops as shown in Figure 3 is also a chain.

The Expectation-Maximization (E-M)[c] algorithm can be used to train HMMs to recognize new sequences. The E-M algorithm automatically adjusts all the probabilities, both for the transitions and for the output symbols. The training data is treated as if it were generated by the model, and the goal is to adjust all the probabilities until the probability of the data is maximized for the given model topology. The E-M learning procedure is guaranteed to find a locally optimal setting of these probabilities. Once we have our trained model, we use a dynamic programming procedure known as the Viterbi algorithm to parse a new input sequence. Parsing a new sequence aligns each base in the sequence to a particular state in the model. If the states represent exons and introns, then the parsed input predicts the locations of the exons and introns in the sequence.

2.2 The Design of HMMs in VEIL

VEIL contains a number of separate HMM modules that are intended to capture properties of eukaryotic DNA sequence and are specifically designed so that VEIL could predict gene structure. The most important feature in the design of an HMM is its topology, which we illustrate here. The transition and output probabilities are learned using E-M, as explained in Section 2.3.

The exon HMM module was designed so that it could capture regularities that appear in exons, such as codon usage (certain codons and dicodons appear more frequently in coding than in non-coding DNA) and periodicity (because of preferential codon usage, some bases show a tendency to appear every third position within coding regions) [8]. Another goal in the design of this model was to rule out in-frame stop codons. The model was initially trained on whole exons only, before being tied together with the other models as explained below. We designed a similar HMM to capture introns. Compared to the other modules, the exon and intron HMMs are quite small. We found that the information in the splice sites, upstream, and downstream models was much more useful for performing the coding/noncoding segmentation. Note that because HMM methods do not impose any constraints on the shape of these models, virtually unlimited variation in the design of the exon and intron models is possible, and it is likely that other designs can be found that will improve VEIL. The modular design of the system will allow us

[c] E-M for HMMs is also commonly referred to as the Forward-backward or Baum-Welch algorithm.

to experiment with alternative models easily. The exon model topology currently used in VEIL is shown in Figure 1.[d]

2.2.1 A walk through the exon model

Each arrow in Figure 1 is a transition, which has an associated probability. Each node, or state, outputs one symbol (one DNA base) when it is processed. (Alternatively, the whole system can be interpreted as reading symbols instead of writing them.) A node with a symbol written inside it can only output that symbol. The other nodes output all four symbols (bases) in proportion to probability distributions stored at those nodes. A walk through this model starts with a transition from another model, which in VEIL must be either the intron model or the "upstream" model representing intergenic DNA and the 5' UTR region. Once the exon model has been entered, the walk proceeds by visiting a series of states, making sure that the symbol output by each state matches the symbol from the sequence. The walk leaves a state by following a transition out of that state, visiting the next state, and so on, outputing one symbol in each state until the end of the sequence is reached. At the end of the sequence, VEIL must be in a designated final state; if so the path is acceptable. The product of the transition probabilities on the edges in that path and the output probabilities from the nodes in that path is an estimate of the probability that the model generated the sequence. Intuitively, this can be thought of as the confidence we have that the model might generate the sequence by that path.

As shown in Figure 1, there are four ways to walk through the exon model, corresponding to the four possible pairs of exon boundaries. First, the exon can start with the start codon, generate codons by visiting a node in each of the stages of the model and returning to the first stage by a backedge to generate the next codon, and then exit the model by one of the six transitions that lead to the donor (5' splice site) model. This path would correspond to a typical initial exon. An alternative walk through the model starts with the start codon and exits to the downstream model through the states representing a stop codon. This walk represents a single exon gene. An internal exon would take a path starting at the acceptor (3' splice site) model and end up at the donor model. Finally, the last exon in a gene consisting of multiple exons would start at the acceptor model and continue until it reaches a stop codon, which would force it to exit to the downstream model.

[d] At the present time, there is no publicly available HMM-building software that can handle the size and topological complexity of the models in VEIL. Systems that allow the construction of chain models such as HMMer and SAM are available, but we needed to construct arbitrary models with any topology (e.g., not just chains) and to process sequences of tens of kilobases reasonably quickly. Therefore we developed our own HMM system to build, train, and test these models. Our code for constructing novel HMMs will be made available in the near future.

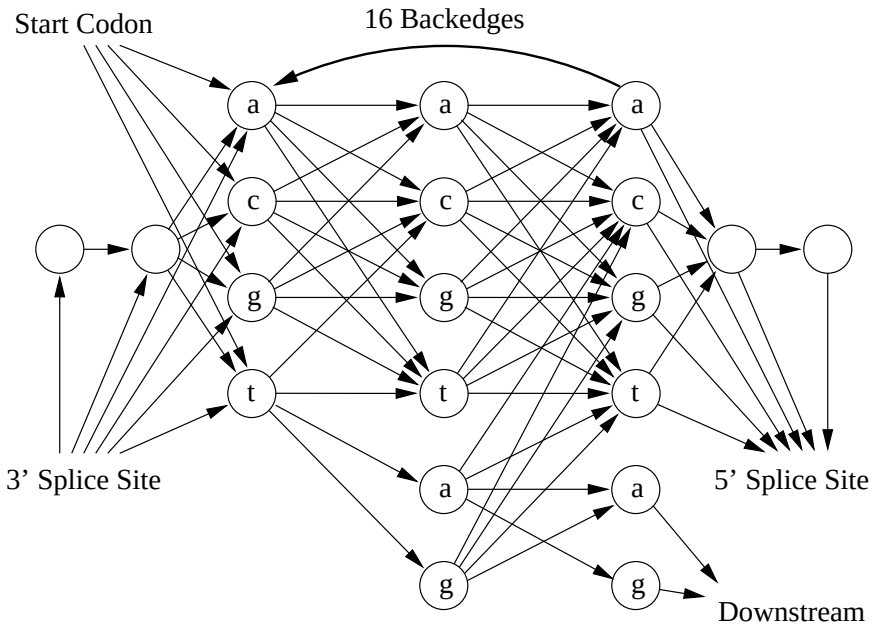

Figure 1: The exon and stop codon models in VEIL. This model can be entered in two ways: either just after outputting a start codon, or upon leaving the 3' splice site model, which follows the intron model. The three central columns of states correspond to the three codon positions. Each of these 12 states is labeled with the base that it can output. The system outputs bases three at a time, looping back after each codon. Note that the paths corresponding to a stop codon (TAA, TAG, and TGA) all force the system to exit from the model (four states at lower right of figure). Alternatively, the system can exit through the 5' splice site, in which case an intron must follow the exon. The two blank states on either end of the model can output any base; these "absorbing states" allow the model to align itself to the proper reading frame, as splice junctions need not respect codon boundaries.

Note that the exon model consists primarily of three stages, meaning that bases will be read by the model in groups of three, corresponding to codons. Thus if certain codons appear with characteristic frequencies in exons, the probability distributions on the corresponding sets of three states can be adjusted accordingly. The backedges allow the exon model to loop through any number of codons. Of course, exon-intron boundaries need not correspond to codon boundaries, so the model needs to be able to adjust itself. This is accomplished by the two states on either side of the model, which can absorb up to two extra bases before entering or leaving a coding region. Once the model is inside a coding region, it must read bases in groups of three, but it can shift at either or both ends. (Note that this model cannot recognize a 1- or 2-base exon.)

The four states at the bottom of Figure 1 represent the stop codon model. Note that any *tag*, *tga*, or *taa* that starts at the *t* node in the first stage is forced to follow a path to the downstream model; i.e., it leaves the exon model. Other codons that start with a *t* lead back into the third stage of the main model. By this method we avoid erroneously allowing in-frame stop codons that are not the last codon of the last exon.

2.2.2 The intron and splice site models

The intron model of Figure 2 is very similar to, though a bit simpler than, the exon model. There are still three main stages which are traversed in order and cyclically. This is because, based on previous research [8], there are differences in the frequency of codon usage between exons and introns; VEIL is trying to capture those differences here. All paths through the intron model start at the donor site model and exit through the acceptor site model. The machinery for handling stop codons is, naturally, unnecessary in this model.

The splice site models are very simple chains with multiple stages. Like a pipeline, every path through them starts at the first stage, ends at the last stage and has exactly the same length. These models have more edges and states than the exon or intron models, yielding a more precise probability estimate for each path through them. The donor model has 9 stages, and the acceptor model has 15 stages. The lengths of these models were based on the lengths of the consensus sequences for donor and acceptor sites as given in Mount et al. [17] and Senapathy et al. [22].

While the exon model will mark every state traversed in it with an *e*, and the intron model marks every state traversed in it with an *i*, the splice site models have some states indicating they are part of an exon and others that are part of an intron. In the donor site model shown in Figure 3, stages 1-3 are part of the exon and stages 4-9 are part of the intron, which indicates that the bases that align with them should be marked accordingly. In the acceptor site model (not shown), where the exon-intron boundary occurs at positions 14-15, stages 1-14 are part of the intron while stage 15 part of the exon.

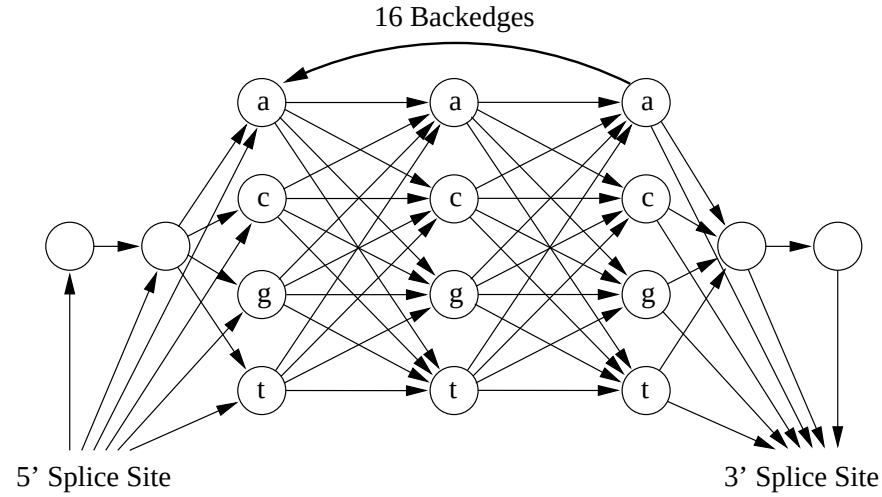

Figure 2: The intron model in VEIL. This model functions similarly to the exon model, with a few important differences. It reads bases three at a time in order to capture differences in the frequency of codon usage between coding and noncoding regions. Unlike the exon model, stop codons do not lead out of this model. The intron model must be entered and exited via splice junctions, which enforces the constraint that exons must appear on both sides of each intron.

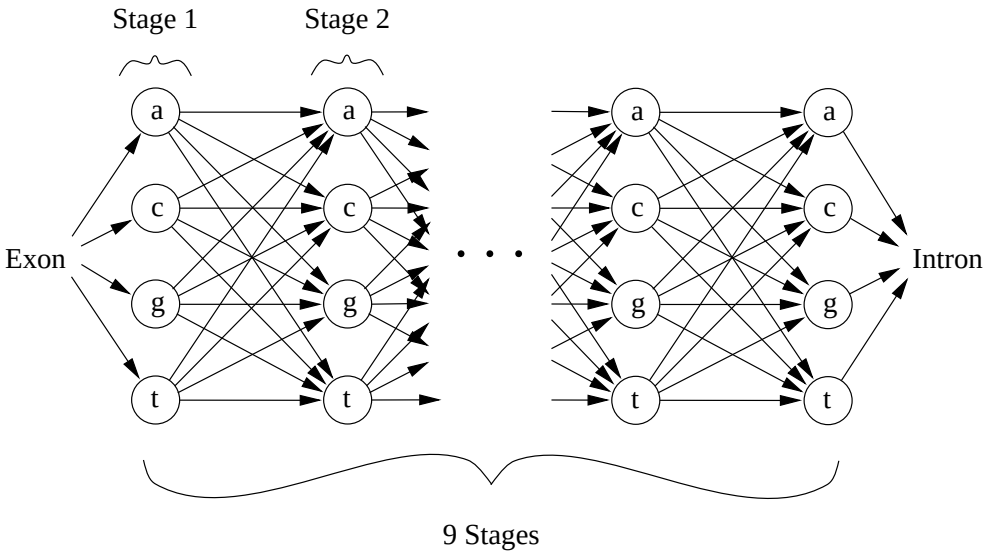

Figure 3: The donor site (5' splice site) model. Sequences must pass through this model to get from the exon model to the intron model. The exon-intron boundary occurs between stages 3 and 4; therefore stages 1-3 are part of the exon and stages 4-9 are part of the intron. Each state can output only one base, as indicated by the labels. Each edge between two states here contains the conditional probability of outputting a base in the latter state given the base shown in the previous state.

2.2.3 Other HMM modules

The start codon model, shown in Figure 4, is an extremely simple chain whose mechanism is obvious from the diagram. The path through the start codon model starts in the upstream model and leads to the exon model.

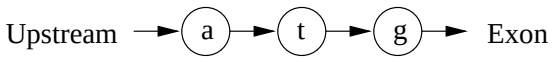

Figure 4: Start Codon Model

We also created modules to capture the intergenic regions (upstream of the start of translation, and downstream of the stop codon). The intergenic model contains two simple chains of stages (representing the upstream and downstream intergenic pieces) with loops on the ends to absorb extra bases. Those loops allow the model to align at any place within the input sequence. The upstream chain has 15 stages, and the downstream chain has 10. The absolute lengths of these chains may not be optimal; however, we attempted to make the upstream chain long enough to capture some of the sequence regularities known to exist 5' of the start codon.

Finally, we created a small module to capture the 3' polyadenylation binding site, whose consensus sequence is AAATAA. This hexamer, which is the site where binding occurs before the poly-A tail is added to mRNA, occurs in 70% of the 3' noncoding regions in our training data. It occurs at widely varying distances from the stop codon, but this poses no problem for the VEIL architecture. We simply created a chain of six states that recognizes only AAATAA, analogous to the 3-state chain for the start codon. This chain is looped into the final stage of the downstream model, so that VEIL is "encouraged" to put an AAATAA after the stop codon if possible. An important note here is that we added this chain to the model *after* training, and no additional training was necessary. The overall performance improved by 1-2% as a result. This demonstrates how easily biological information can be incorporated into the VEIL HMM.

2.2.4 The combined HMM

The figures shown so far omit some edges for clarity, but in reality the separate modules of VEIL are connected with numerous edges. Figure 1 shows six edges leaving the exon model and entering the 5' splice site (donor) model, and Figure 3 shows four edges from the exon model into the 5' splice site model. The true picture would show 24 edges leading from the exon model to the 5' splice site model, one for each of the pairings (x, y), where x is one of the states in the exon model depicted with a transition to the donor site model, and y is a state in the donor site model depicted with a transition from the Exon model.

Tying all the smaller models together produces the complete VEIL model, shown schematically in Figure 5. This diagram is even more compressed, in order to give an overview of the VEIL architecture. Edges in this schematic represent multiple edges in the true model. Hence, in this schematic, the single edge leading from the exon model to the donor site model represents 24 edges. Other edges in the schematic are similar.

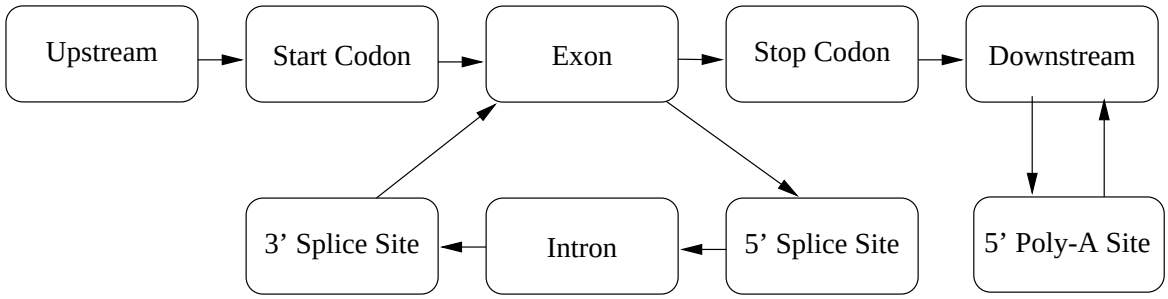

Figure 5: Combined Model Schematic

Note that to combine the models, a total of 112 edges had to be added. These are the edges that benefitted from the final round of E-M training, where the complete model was trained on whole sequences (described below). Many of these edges are critical, especially those that cross into and out of the exon model. These transitions must be made correctly at both ends of an exon in order to correctly predict the exon, which is one of the main criteria we used for judging the accuracy of the system. In total, the VEIL HMM has 1003 edges and 241 states.

The VEIL model contains many states that are labeled with specific bases; this points out a nice feature of HMMs that we exploited in our system. Namely, during training we can specify that certain states and transitions are fixed, i.e., not subject to re-estimation by the learning algorithm. In this way, certain parts of the model were hard-wired with specific biological information. For example, the start and stop codons were explicitly coded as sequences of three states each, and the 5' and 3' splice junctions were coded as chains of states that captured the consensus sequence probabilities for those regions.

After the splice sites are fully trained, one can read off conditional probability matrices from the edge and state probabilities in those models (this use of conditional probabilities was suggested by Simon Kasif). These edges contain, after training, the conditional probability of observing base y in position i given that base x appears in position $i-1$. Unlike neural networks, the weights on the edges in HMMs have a precise probabilistic interpretation. Furthermore, the ability to interpret the edges allows us to validate the training algorithm, because we can compare the values in the splice site models to sequence profiles for splice sites that appear in the literature. For example, we can look at the probabilities on outgoing edges in our 5' splice site model, where the

edge from the G state in stage 6 to the A state in stage 7 has a transition probability of 0.82. This means that if an intron starts with GTG (the G and T edges in the appropriate stages have probability 1.0 in the 5' splice site model), then the next base will be A with probability 0.82. This example illustrates how the HMM's strict probabilistic framework helps provide insight into what the model "learned."

2.3 Training an HMM: the E-M algorithm

Three efficient algorithms, the *Forward*, *Viterbi*, and *Expectation Maximization (E-M)* algorithms, are used in computing with HMMs. For our experiments, we only needed the Viterbi and E-M (sometimes called the Forward-Backward) algorithms, so we will only describe those algorithms here. This description is merely a summary; for details the reader should see Lee [16].

The E-M algorithm is used to solve what is known as the learning problem in HMMs; i.e., determining reasonable values for all the probabilities in an HMM. The model topology is fixed, and all of the output probabilities and transition probabilities are initialized to random values. (If any prior estimates of these probabilities are available, it is usually beneficial to initialize them to these estimates instead.) Upon being presented with a set of DNA sequences, the E-M algorithm re-estimates all of these probabilities. Briefly, it runs each training sequence through the model and computes the posterior probability $P(s|M)$ for each sequence s. These values are then multiplied together to produce $P(S|M)$, where S represents all the sequences. The re-estimation procedure then adjusts all of the probabilities in order to increase $P(S|M)$. The data are then run through the model again and the probabilities are further refined. This process is iterated until the $P(S|M)$ is maximized. The E-M algorithm is guaranteed to converge to a locally optimal estimate of all the probabilities in the model.

We utilized a variant of the E-M algorithm that we customized for DNA sequence analysis. Prior to training the combined model, the DNA alphabet of $\{A, C, G, T\}$ was combined with the alphabet we used to label our sequences, $\{N, E, I\}$ (intergenic, coding/exon and noncoding/intron). This produced an alphabet of twelve symbols, $\{NA,NC,NG,NT,EA,EC,EG,ET,IA,IC,IG,IT\}$, that described the DNA and the proper labelling associated with each base.

Prior to processing a training sequence, the symbols are replaced using the 12-symbol alphabet. Thus portions of the sequence are labelled according to whether they fall in exons, introns, or noncoding regions. Then when VEIL processes the sequence, those portions labeled with, for example, $\{EA,EC,EG,ET\}$, are forced to align to the exon model. Likewise, those portions of the sequence that fall in introns and intergenic regions are forced to align to the correct parts of the HMM. Once the model is trained, the states are labeled not with the 12 symbols, but with the original set of four. This allows VEIL to process new sequences, for which the exon and intron regions are unknown, in order

to produce the optimal alignment of those sequences to the model.

Each iteration of the E-M algorithm runs in time $O(Ne)$, where N is the total length of all the sequences in the training database and e is the number of edges in the model. In theory, the number of iterations can be very large, which might seem to make this an extremely time-consuming procedure. In practice, though, E-M usually converges quickly, and in our experiments it always converged in less than a dozen iterations for each of the modules.

2.4 Parsing a new sequence: the Viterbi algorithm

After training, the model is ready to be used to interpret new sequences. For this purpose, we use the Viterbi algorithm [27], a dynamic programming algorithm that efficiently aligns any sequence to an HMM. Given a sequence and a trained HMM, the Viterbi algorithm will find the *most likely* sequence of states through the model for that particular sequence. (In addition, the Viterbi algorithm computes the probability of the model producing the sequence via that path.) Although there are an exponential number of distinct paths through the model, the Viterbi algorithm finds the best one in $O(ne)$ time, where n is the length of the aligned sequence. (The Markov assumption makes it possible to follow an efficient dynamic programming strategy.) Since VEIL contains explicit states representing the start codons, splice junctions, and stop codons, this alignment tells us directly where the first exon begins and where each of the subsequent exon-intron transitions occurs.

In order to search efficiently for the best path, the Viterbi algorithm builds a data structure called a *trellis*, which requires $O(ne)$ storage space. (For details of how the trellis is constructed, see [16].) The trellis consists of a sequence of stages, one for each successive base in the input. Each stage contains one node for each state in the model. Every edge (u, v) in the model appears once between every successive pair of stages in the trellis, connecting node u to node v. Thus the trellis stages themselves can be quite large and densely connected to each other. The algorithm builds the trellis from left to right, beginning with the first character (the first base) of the input sequence. The space requirement is clearly very large for long sequences and large models. Fortunately, the trellis can be *pruned* to reduce space while the algorithm is being run, using the following observation: as the trellis is built, we can keep track of which nodes in each stage appear along any optimal paths. Most nodes will fail this test, and are therefore not needed any longer. After the trellis is extended from stage $i-1$ to stage i, all vertices in stage $i-1$ that are not used in an optimal path can be removed from the computation as later stages are computed. All vertices in stages $i-2$ through 1 that are not referenced by any later stages can be recursively removed as well. In our experiments, this pruning

technique yielded more than a 95% space savings. [e]

2.5 Assumptions inherent in VEIL

There are a number of assumptions that VEIL makes when it parses a new sequence. These assumptions are shared by most gene-finding systems, and therefore do not invalidate the comparisons we present. However, some of them still need to be relaxed to produce a completely general gene finder.

1. The first "exon" of every gene is defined to begin with a start codon; i.e., any 5' untranslated sequence is processed by the upstream model.

2. Noncoding regions and introns are flanked by exons.

3. Each piece of DNA presented for analysis will start and end with a noncoding region and contain a single gene.

4. The minimum length of a coding exon is 7 bases. (The acceptor site model includes one coding base, the exon model absorbs at least three bases, and the donor site model absorbs three more coding bases.)

These assumptions have a number of shortcomings, most of which are minor. There are genes in which the stop codon appears in the middle of an exon (sometimes not even the last one). On the 5' end, the start codon may appear somewhere in the middle of an exon (possibly not even the first).[f] The parsing process in VEIL produces a list of exons that by definition begins with a start and ends with a stop codon; therefore it is more accurate to refer to this process as locating the coding regions, not the entire gene. More critically, VEIL's output leads directly to the protein sequence produced by the gene.

Assumption number 3 is the only difference between a truly general gene-finder and VEIL; we assume that exactly one gene exists in the data, which of course is not necessarily true. Other benchmark experiments to date have also relied on this assumption [5]. Extending VEIL to relax this assumption is discussed below.

Finally, many gene finding systems take advantage of the fact that exons are in consistent reading frames; e.g., if the first exon ends in the middle of a codon, the

[e] This increased the program's speed substantially by reducing the number of page faults.

[f] There is some inconsistency in the definition of "gene" and "exon" that is reflected in the annotations in the DNA sequence database entries themselves. For example, some entries use the stop codon to *define* the end of the final exon, while others use the location of the end of the spliced mRNA transcript. Likewise with the start of transcription and translation – some entries use the start codon to define the start of the first exon, even though this may not be technically correct when compared with the transcript.

second exon will take up in the same frame where the first one left off. This assumption is important to other dynamic programming systems such as GRAIL [28] and MORGAN [21], because it significantly reduces the number of alternative parses. It leads to a problem, though, since it makes the segmentation algorithm very sensitive to frame shift errors realized as indels in the exon regions.[g]

Because an HMM uses only local information, it is difficult to adjust it to keep track of frame shifts; essentially, one would have to duplicate or tie together the output probabilities of large pieces of the model to represent different frames and phases. Although this could be accomplished, we made a conscious decision to ignore frame information, which makes the HMM simpler and avoids the frameshift problem. On the downside, relaxing this assumption opens up the possibility that the VEIL will erroneously label certain regions as exons: it is quite possible that subsequences that are not true exons might have strong coding characteristics. If such subsequences are out of frame with respect to a gene being assembled, a frame-sensitive method can exclude them, while VEIL cannot. As a result, VEIL might erroneously include exons that do not match the coding frame of the rest of the gene. On other hand, VEIL's splice junction modules should effectively exclude most spurious exons, since noncoding regions are unlikely to match well with the splice site models.

2.6 Finding more than one parse

By default, the Viterbi algorithm gives us the single best alignment of a sequence to our model, and this is what the current version of VEIL outputs. It is straightforward to modify this to determine the k best parses by keeping the best k in each state in the Viterbi trellis. This requires a factor of k additional storage space, and a factor of $O(k \log k)$ additional time. It is unclear whether using the k best parses would provide much benefit, however, because the top ranking parses all look highly similar to one another, a phenomenon that has been observed in other systems as well [24].

3 Methods and Results

3.1 The database of DNA sequences

The final accuracy of the VEIL system, and the accuracy of the probabilities it learns, depend to a large extent on the quality and quantity of data used to train them. In order to compare VEIL to other gene finding systems, we used the database collected in the recent comprehensive study by Burset and Guigo [5], comparing a number of

[g]The authors of GRAIL have proposed an iterative algorithm for removing apparent frame shifts[29, 9].

the major gene-finding systems using a single database. This data consists of sequences from GenBank release 85.0 since January 1993, so it included relatively recent entries. Burset and Guigo first collected every complete vertebrate protein coding sequence, and then discarded entries in a series of quality control steps. They removed all pseudogenes, all entries with in-frame stop codons, all entries with no introns (many of which were cDNAs), and all entries with non-standard splice junctions (i.e., the sequence contained an intron that did not begin with GT and end with AG). They also removed immunoglobulins and histocompatibility antigens. The resulting data set contains 570 complete sequences, each containing exactly one gene with at least one intron.

Among these 570 vertebrate sequences, there are a total of 2649 exons and 2079 introns. Six of the sequences contain no 5' untranslated region (i.e., they begin with a start codon), but the remaining sequences contain at least a few bases upstream of the initiation codon.

Note that including different organisms in the data implies that the splice site models will contain "average" statistics on the consensus sequence for these sites. Insofar as the splice site machinery differs among vertebrates, this will cause VEIL to overgeneralize somewhat. The only remedy to this is to collect enough data on a single organism to create a species-specific version of VEIL. As part of our future efforts, these more specific models will be constructed (a human-specific VEIL is already under development) as data becomes available.

3.2 Experimental design

We performed a 5-fold cross validation experiment to estimate how well the system would perform when tested on data that was not in the training set. A cross validation (CV) is a standard experimental technique for determining how well a classifier system will perform on unseen data. The data is randomly divided into 5 roughly equal-sized partitions (for 5-fold CV). For each of these partitions, the system is trained on the other four partitions and then tested on the held-out partition. Combining the results from the five test sets gives an accurate estimate of how well the system will perform on unseen data. This is because each test set was new data with respect to the partitions the system was trained on. In our experiments, the 570 sequences were randomly partitioned into five sets of 114 sequences each. Each training set in the experiment contained roughly 2100 exons and each test set contained about 500 exons. Each of five copies of the VEIL HMM was trained on four of the five sets and tested on the remaining one.[h]

Although the 5-fold CV is an excellent design for estimating accuracy on unseen data, one confounding issue with this data set is the existence of highly homologous

[h]The complete data set, broken up into the five partitions used in these experiments, is available on the VEIL ftp/web site, at ftp://ftp.cs.jhu.edu/pub/veil.

sequences in the data. For a new sequence with close homology to an existing GenBank sequence, the database annotation will in most cases be superior to the results of any gene finder. To estimate the accuracy of VEIL on non-homologous sequences, therefore, we repeated the 5-fold CV above after removing homologous sequences from the Burset and Guigo data set. Using the same partitioning of the data, we removed from each test set all sequences that were at least 80% identical to any sequence in the training set. This procedure removed an average of 18 sequences from each test partition. These results are reported as the "non-homologous" test results below.

It should be noted that the objective function that is maximized by the E-M algorithm is not the same as the functions we used to evaluate VEIL's performance. The objective function is the metric on which optimization is taking place. The E-M algorithm maximizes the probability of the training sequences given the model; on the other hand, the tables given below describe the percentage accuracy in terms of total base pairs correctly labeled and total exons correctly found. The local maximum that the E-M algorithm finds may not be a local maximum in the metrics we use to evaluate the performance of our system. For example, there is no reason to believe that maximizing the probability of the training sequences will be the same as maximizing the number of exons that have both edges correct.

We therefore developed a method to avoid "overtraining" and to focus the training on the accuracy at finding exons: we sampled the desired metrics on the training set after each iteration of training. The four metrics we used in this step were (1) the percentage of true coding bases correctly predicted (nucleotide sensitivity), (2) the percentage of predicted coding bases that were correct (nucleotide specificity), (3) the percentage of true exons predicted exactly, both on the 5' and the 3' ends (exon sensitivity), and (4) the percentage of predicted whole exons that exactly matched a true exon (exon specificity). We used the arithmetic mean of the four metrics as an indicator of the goodness of a particular model. We chose to stop training the HMM as soon as we find that the next iteration of training decreases this metric more than a threshold amount. This is illustrated in Figure 6. The rapid convergence of these graphs (in some cases the system is close to its optimum performance in just one iteration) is due to the pre-training of the submodels, which initializes the probabilities of a large majority of edges and nodes to locally optimal values. Many of these pre-trained values, such as those in the splice junction models, were fixed and were not adjusted by the E-M procedure.

A final observation about the training procedure is that in its current state, most of the VEIL model can be trained "directly"; i.e., the E-M algorithm is not strictly necessary. Because most of the states only allow one output, there is only one possible alignment of each training sequence to the model. Using this property, the output and transition probabilities can be computed directly in a single pass through the training data, without any need for an iterative procedure (Simon Kasif, personal communication). This would result in a much faster training algorithm. However, we plan to add

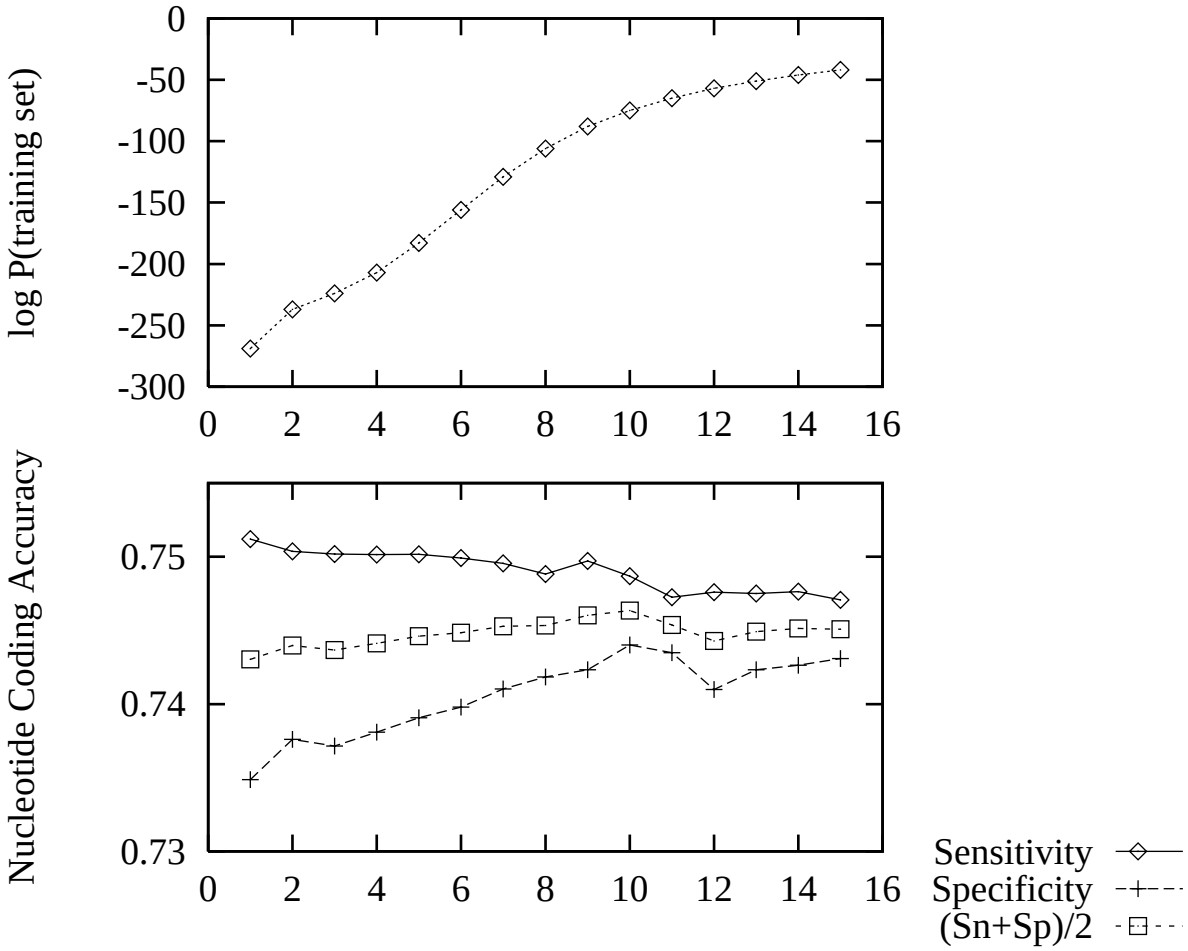

Figure 6: Overtraining avoidance graphs. The top graph shows how the probability of the training data given the model increases steadily with each training iteration. The vertical axis is the log base 4 of the probability of data, plus a normalization constant. A training iteration is one pass of the data through the E-M algorithm. The lower graph shows that, although the probability of the data increases throughout training, the accuracy does not increase uniformly, but seems to peak after approximately 10 iterations. The graph shows accuracy measured in terms of the number of coding nucleotides correctly predicted.

more states to VEIL in the future, and we expect to continue using E-M because it gives us the greatest flexibility to extend and re-train the model.

3.3 Results

The results are summarized in Table 1. The results for the training and test set for each of the five partitions of the cross validation are shown, as well as the combined results for all of the five test sets. The union of the test sets is the entire set, so the combined result shows the overall accuracy on the data. Note that every example appeared exactly once as a member of a test set. The table also shows the accuracy of VEIL on the non-homologous test data, for which no test sequence is 80% or more identical to any training sequence. This led to a total of 93 sequences being removed from all partitions.

In the table, sensitivity (Sn) for nucleotides is the percentage of coding bases that are labeled correctly, and for exons it is the percentage of whole exons which are predicted exactly. Specificity (Sp) for nucleotides is the percentage of exon bases that are predicted correctly, and for exons it is the percentage of exons predicted that are exactly correct. **CC** is the correlation coefficient, defined as

$$CC = \frac{(TP)(TN) - (FP)(FN)}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}}$$

where TP is true positives (number of coding bases correctly predicted), TN is true negatives (number of noncoding bases correctly predicted, FP is false positives, and FN is false negatives. **1ME** stands for "1 matching edge" and is the percentage of exons for which VEIL predicted at least one of the edges exactly; and **Ov** is the percentage of exons for which VEIL's prediction overlaps the true exon (including exact matches). **P(All)** is the probability that VEIL will mark any base correctly.

Overall, VEIL obtained 83% sensitivity and 72% specificity for coding bases on the full data set, which dropped to 80% and 72% on the non-homologous data. The correlation coefficient was 0.73 on the full set, and 0.71 using non-homologous data. Exact exon prediction is a more stringent criterion which counts only those coding exons for which VEIL gets both the 5' and 3' ends exactly right. On this measure, VEIL achieved 53% sensitivity and 49% specificity, which dropped slightly to 50% and 47% on the non-homologous data.

A comparison of these and other results in Table 1 to eight other major gene-finding systems is given in Table 2. Most of these results are from Burset and Guigo's comprehensive study [5], with the exception of Genie [19] and MORGAN [20]. This table shows that VEIL's performances on nucleotide prediction and especially on exact exon prediction are competitive with the best results reported previously. Only three systems (Genie, MORGAN, and FGENEH) show better overall accuracy as measured by the

Partition	Nucleotides				Whole Coding Exons			
	Sn	Sp	CC	P(All)	Sn	Sp	1ME	Ov
Full Vertebrate Data Set								
Train-1	0.82	0.75	0.75	0.93	0.54	0.53	0.73	0.80
Test-1	0.80	0.76	0.74	0.93	0.51	0.49	0.71	0.80
Train-2	0.82	0.74	0.74	0.93	0.53	0.50	0.73	0.80
Test-2	0.80	0.75	0.73	0.93	0.52	0.52	0.70	0.78
Train-3	0.82	0.75	0.74	0.93	0.55	0.52	0.73	0.81
Test-3	0.75	0.70	0.68	0.92	0.45	0.44	0.64	0.72
Train-4	0.82	0.75	0.74	0.93	0.54	0.51	0.73	0.81
Test-4	0.79	0.72	0.71	0.93	0.50	0.46	0.69	0.76
Train-5	0.82	0.73	0.73	0.93	0.54	0.50	0.72	0.80
Test-5	0.87	0.70	0.74	0.92	0.53	0.47	0.77	0.86
Test-All	0.83	0.72	0.73	0.92	0.53	0.49	0.73	0.81
Non-Homologous Test Data								
Test-1	0.79	0.73	0.72	0.92	0.50	0.47	0.71	0.79
Test-2	0.80	0.74	0.73	0.93	0.50	0.51	0.69	0.77
Test-3	0.76	0.70	0.69	0.92	0.46	0.44	0.66	0.74
Test-4	0.79	0.71	0.71	0.93	0.51	0.46	0.70	0.77
Test-5	0.87	0.69	0.73	0.92	0.51	0.45	0.76	0.85
Test-All	0.80	0.72	0.71	0.92	0.50	0.47	0.70	0.78

Table 1: Results of 5-fold cross validation for the full 570 gene vertebrate data set and for the non-homologous cross validation. In the non-homologous data, all sequences that had 80% or more homology to any training sequence were removed from each test set. Sensitivity (Sn) for nucleotides is the percentage of coding nucleotides correctly labeled as coding. Specificity (Sp) is the percentage of nucleotides labeled as coding that were actually coding. P(All) is the overall probability of predicting any base correctly. The right half of the table contains the corresponding values for whole exons; i.e., the accuracy at predicting the coding regions exactly. 1ME is the percentage of exons for which one or both edges was correct, and Ov is the percentage of true exons that overlapped a predicted exon. The Test-All line contains the combined results for all test data.

System	Nucleotides			Whole Coding Exons		
	Sn	Sp	AC	Sn	Sp	Ov
VEIL	0.83	0.72	0.73	0.53	0.49	0.81
MORGAN	0.82	0.80	0.78	0.58	0.54	0.84
Genie	0.78	0.84	0.77	0.61	0.64	0.85
FGENEH	0.77	0.85	0.78	0.61	0.61	0.85
GeneID	0.63	0.81	0.67	0.44	0.46	0.72
GeneParser2	0.66	0.79	0.67	0.35	0.40	0.71
GRAIL 2	0.72	0.87	0.75	0.36	0.43	0.75
GenLang	0.72	0.79	0.69	0.51	0.52	0.79
SORFIND	0.71	0.85	0.73	0.42	0.47	0.76
Xpound	0.61	0.87	0.68	0.15	0.18	0.67

Table 2: A comparison of VEIL's performance on 570 vertebrate sequences with other gene-finding systems. Sources for all results are cited in the text. AC is the approximate correlation proposed by Burset and Guigo [5], as a more stable overall indicator than the correlation coefficient. VEIL's AC and CC values were identical for this data.

approximate correlation [5], but this is confounded by the fact that the training data for each of these systems overlapped the test set here. (For Genie, 120 of the 570 sequences were included in Genie's all-human training set. For MORGAN, a separate test set comprising 20% of the sequences was used. For the other systems, overlap exists but is not known precisely.) A direct comparison is difficult to make because of this mingling of training and test data. In order to make such a comparison reliably, the other systems would have to be re-trained and tested using a methodology such as cross validation, which would require access to the source code.

4 Conclusions

The design of VEIL is still changing. Hidden Markov Models can be constructed in an infinite variety of shapes and sizes, and their effectiveness varies widely depending on the design. The VEIL system demonstrates, in its current form, that HMMs can be trained to find coding regions in DNA as accurately as, if not more than, the best known gene-finding systems. With further development, it is quite likely that VEIL's performance will improve. One direction for improvement is to design and integrate further small HMMs into VEIL that model additional signals associated with genes. For example, we can model the promoter sequences upstream (5') of the start of transcription. The 3' polyadenylation site has already been added, resulting in an across-the-board

improvement that is included in the results presented above.

As more data becomes available on individual organisms, it will be possible to train HMMs on data from a single species. We are in the process of building a human-only VEIL, which we expect to release during 1997. In the near future we expect to be able to create specialized versions of VEIL for mouse, arabidopsis, and other organisms, as the genome projects for those organisms proceed.

With a simple modification, VEIL can be used to perform whole genome and arbitrary sequence parsing. As stated in section 2.5, our current implementation requires the sequence to begin and end its alignment in the intergenic region. By making all states be valid initial and final states, and connecting the two intergenic region models together, we can permit VEIL to find multiple genes (or no genes at all) in an input sequence. This would allow VEIL to find genes in larger stretches of uncharacterized DNA, and it would allow us to remove the assumption in the current system that the input sequence contains exactly one gene.

Naturally, if one can find homologous sequences by searching the major sequence databases, and if the homologous entries include annotations showing where the exons are located, then the task of finding a gene in a new sequence becomes much easier. A number of systems have already reported major improvements by tying together a purely computational approach with a database search method. Burset and Guigo [5], for example, found that systems that did this achieved a substantial increase in the accuracy of exact exon prediction, from a typical level of 40% on average to more than 60%. A related idea is to use the new and rapidly growing expressed sequence tag (EST) databases to improve the accuracy of coding region prediction. If one finds a hit between a new sequence and an entry in an EST database, then the corresponding subsequence can be labelled as part of an exon.

We have already implemented in VEIL a mechanism for using information from ESTs or homologous sequences; we use the alphabet-shift mechanism described earlier. Simply stated, the alphabet is augmented by adding four new symbols: (EA, EC, EG, ET). These symbols indicate the corresponding base must align inside the exon model. Any subsequence that is found to hit an EST would be replaced with this augmented version of the sequence. Because these special symbols appear only within the exon model, the subsequences that hit the EST database would be forced by VEIL to align to coding regions. If an EST spanned two or more exons, VEIL would detect this as well; in fact, in such cases VEIL would be guaranteed to find the correct exon-intron boundaries. An open problem that remains to be solved is the presence of large amounts of untranslated sequence in the EST databases, particular 3' UTRs, which should align to the downstream model rather than to the exon model.

Although database lookup procedures are somewhat distinct from the problem of modelling genes *de novo* using computational methods, they clearly represent an important opportunity for overall improvement. And as long as sequences with no clear

homology to known database entries are being discovered, computational modelling methods such as HMMs will serve an important function.

Acknowledgements

Thanks to Radu State for help implementing the 3' polyadenylation site module. This material is based upon work supported by the National Science Foundation under Grant Nos. IRI-9223591 and IRI-9530462, and by the National Center for Human Genome Research at the National Institutes of Health under Grant No. K01-HG00022-1. KHF is supported by the Department of Energy under award number DE-FC02-91ER61230.

References

[1] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pages 308–319, 1983.

[2] P. Baldi, S. Brunak, Y. Chauvin, J. Engelbrecht, and A. Krogh. Hidden Markov models of human genes. In *Advances in Neural Information Processing Systems*, volume 6, pages 761–768. Morgan Kaufmann, 1994.

[3] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure. Hidden Markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, 91:1059–1063, February 1994.

[4] M. Brown, R. Hughey, A. Krogh, I. Mian, K. Sjolander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proc. First Internatl. Conf. on Intelligent Systems for Molecular Biology (ISMB-93)*, pages 47–55, Menlo Park, CA, 1993. AAAI Press.

[5] M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, 1996.

[6] G. Churchill. Hidden Markov chains and the analysis of genome structure. *Computers and Chemistry*, 16(2):107–115, 1992.

[7] A. L. Delcher, S. Kasif, H. R. Goldberg, and B. Hsu. Probabilistic prediction of protein secondary structure using causal networks. In *Proc. of the Eleventh National Conf. on Artificial Intelligence*, pages 316–321, 1993.

[8] J. Fickett and C.-S. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20(24):6441–6450, 1992.

[9] X. Guan and E. Uberbacher. Alignments of DNA and protein sequences containing frameshift errors. *Computer Applications in the Biosciences (CABIOS)*, 12(1):31–40, 1996.

[10] R. Guigo, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *J. Molecular Biology*, 226:141–157, 1992.

[11] G. Hutchinson and M. Hayden. The prediction of exons through an analysis of spliceable open reading frames. *Nucleic Acids Research*, 20:3453–3462, 1992.

[12] K. Karplus. Using Markov models and Hidden Markov Models to find repetitive extragenic palindromic sequences in Escherichia coli. Technical Report UCSC-CRL-94-24, University of California, Santa Cruz, CA, July 1994.

[13] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *J. of Molecular Biology*, 235:1501–1531, 1994.

[14] A. Krogh, I. Mian, and D. Haussler. A hidden Markov model that finds genes in E. Coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[15] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In *ISMB-96: Proc. Fourth Internatl. Conf. Intelligent Systems for Molecular Biology*, pages 134–141, Menlo Park, CA, 1996. AAAI Press.

[16] K.-F. Lee. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic, Boston, MA, 1989.

[17] S. Mount, X. Peng, and E. Meier. Some nasty little facts to bear in mind when predicting splice sites. In *Gene-Finding and Gene Structure Prediction Workshop*, Philadelphia, PA, October 1995.

[18] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, pages 257–286, 1989.

[19] M. Reese, F. Eeckman, D. Kulp, and D. Haussler. Improved splice site detection in genie. In *RECOMB 97*, pages 232–240. ACM Press, 1997.

[20] S. Salzberg, X. Chen, A. Delcher, J. Henderson, and K. Fasman. Morgan: A decision tree system for finding genes in dna. Technical Report JHU-97/02, Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218, 1997.

[21] S. Salzberg, X. Chen, J. Henderson, and K. Fasman. Finding genes in DNA using decision trees and dynamic programming. In *ISMB-96: Proc. Fourth Internatl. Conf. Intelligent Systems for Molec. Bio.*, pages 201–210, Menlo Park, CA, 1996. AAAI Press.

[22] P. Senapathy, M. Shapiro, and N. Harris. Splice junctions, branch points, and exons: sequence statistics, identification, and applications to genome project. *Methods in Enzymology*, 183:252–278, 1990.

[23] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA sequences: An application of dynamic programming and neural networks. *Nucleic Acids Research*, 21(3):607–613, 1993.

[24] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA. *Journal of Molecular Biology*, 248:1–18, 1995.

[25] V. Solovyev, A. Salamov, and C. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Res.*, 22:5156–5163, 1994.

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

[27] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, IT-13:260–269, April 1967.

[28] Y. Xu, R. Mural, and E. Uberbacher. Constructing gene models from accurately predicted exons: an application of dynamic programming. *Computer Applications in the Biosciences (CABIOS)*, 10(6):613–623, 1994.

[29] Y. Xu, R. Mural, and E. Uberbacher. Correcting sequencing errors in DNA coding regions using a dynamic programming approach. *CABIOS*, 11(2):117–124, 1995.