Finding Genes in Human DNA with a Hidden Markov Model

John Henderson* Steven Salzberg* Kenneth Fasman†

January 31, 1996

Abstract

This study describes a new Hidden Markov Model (HMM) system for segmenting uncharacterized human genomic DNA into exons, introns, and intergenic regions. Three separate models were designed for each of the three types of human DNA (exons, introns, and intergenic), and training was performed on a corpus collected specifically for this project. The model was then augmented using biological knowledge about splice junction consensus sites, which were used to tie together the separately trained models. The resulting integrated model was then used to segment a test set of human DNA sequences that were not used during training. The initial results are highly encouraging and indicate that an HMM can form the basis of an effective gene-finding system.

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 discovery in the age of high throughput genomic sequencing. A number of gene-finding systems have been developed in the past few years, with varying degrees of success, but the problem still does not have a satisfactory solution. Most of these systems are still under development, though, and improvements continue to appear. Some of the leading systems are GRAIL [21], GeneID [9], GeneParser [17], and FGENEH [18], among others. These systems use a variety of comptuational techniques including neural network algorithms, dynamic programming, rule-based methods, and probabilistic reasoning. Even

*Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218. Email: (jhndrsn,salzberg)@cs.jhu.edu. Telephone: 410-516-8438. Fax: 410-516-6134.

†Genome Data Base, Johns Hopkins University School of Medicine, 2024 E. Monument St., Baltimore MD 21205. Email: ken@gdb.org. Phone: 410-955-9705.

though they are not perfect, the information they provide is valuable enough that some of these systems, which are available via the Internet, are already being widely used.

One technique that seems a natural fit to the gene-finding problem is that of Hidden Markov Models, a probabilistic method designed for sequences of discrete data. HMMs have been used to find genes in *E. coli* [14], but no one has previously used them to find genes in eukaryotic DNA, which presents a more difficult problem. HMMs provide an elegant mechanism to capture information about splice junctions and other signals, as we shall illustrate below.

2 The design of an HMM for finding genes

Hidden Markov Models are noted for their success in the field of speech recognition [1, 15]. Because they are designed to process sequences of information, researchers in computational biology have recently begun to use them for analysis of DNA and protein sequences. For example, they have been used for 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 [11], and for protein secondary structure prediction [8, 4]. Most of the models produced have been relatively small (in comparison to speech recognition systems), in part because of the limited amount of data available but also to reduce the number of free parameters of the system. In an HMM, larger models tend to have many more free parameters and therefore require much more data for accurate training.

At the present time, there is no publicly available HMM system (either commercial or public domain) that can handle the size and topological complexity of the model we have built for gene finding. Therefore we implemented our own HMM system, and developed our models using it. We needed a system in which we could implement HMMs that could handle arbitrary models (any topology is permitted, not just chains) and that could process sequences of tens of thousands of bases reasonably fast. We plan to release our code and data in the near future.

2.1 HMM basics

Although HMMs cannot be covered in detail here, a brief introduction will be useful. 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 a sequence of events, which could be a sequence of nucleotides (for DNA), sounds (for speech processing), or any other sequence. We will speak of the HMM as producing a sequence as output; however, it is just as easy to treat a sequence as input to an HMM. One main assumption is required for an HMM: the events that follow any state v in the model depend *only* on v, not on any state preceding v. This independence assumption

is essential for the computations that we use.

An HMM is defined by a set of states and transitions, often 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 of producing each base $b \in \{A, C, G, T\}$. These probabilities must sum to 1 for each state. Similarly, each transition has a probability, which represents the probability that a generating process makes that transition. Thus the probabilities of all the transitions *out* of a given state v must also sum to 1. As a very simple example, consider the "coin flipping" model in Figure 1. This model only has one state, which

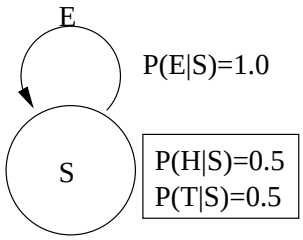

Figure 1: A Markov model for flipping an unbiased coin.

outputs either heads (H) or tails (T) with equal probability. After each output, it makes a transition back to itself with probability 1. If we just "run" this model, it will generate a sequence of characters from the alphabet (H,T).

2.2 HMM algorithms

What makes HMMs useful is the existence of three distinct efficient algorithms for computing with them. These algorithms are called the *forward*, *Viterbi*, and *Expectation Maximization (E-M)* algorithms. For our experiments, we only needed Viterbi and E-M (sometimes called the Forward-Backward algorithm), so we will briefly describe these algorithms here. Space constraints prevent us from explaining the algorithms fully; the interested reader should see Lee [15] for details.

The E-M algorithm is used to solve the learning problem, i.e., to learn good values for all the probabilities in an HMM. The model topology must be fixed by the developer, and all of the output probabilities and transition probabilities are initialized to random values. By presenting the model with a set of DNA sequences, the E-M algorithm can re-estimate all of these probabilities. The data are then run through the model again and the probabilities are further refined. The process is iterated until the probability of the data given the model is maximized. In our experiments, E-M always converged in under six iterations. Each iteration of E-M algorithm runs in $O(ne)$ where n is the

length of the dataset and e is the number of edges in the model. Our code allows smaller models to be constructed using *null states*, which are states that output no characters.

2.3 Parsing with the Viterbi algorithm

After training, the model is ready to be used in a gene-finding system. For this purpose, we use the Viterbi algorithm, a dynamic programming algorithm that efficiently aligns any sequence to an HMM. The idea is that, 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. Although there are an exponential number of such paths through the model, the Viterbi algorithm finds the best one in time that is proportional to ne, where n is the length of the sequence and e is the number of edges (transitions) in the model.

Essentially, then, the Viterbi algorithm *aligns* the sequence to the model. Since our model 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.

2.4 HMM implementation difficulties

Actually, the Viterbi algorithm computes something more precise than the most likely path through the model. It also computes the probability of the sequence given that it took a particular path. With long sequences, however, this number can be vanishingly small, since every transition in the path has a probability of less than 1.

Real numbers with magnitudes larger than 10^{300} or smaller than 10^{-300} cannot be used in computations even on today's modern workstation without special numerical packages. Consider the simple Markov model in Figure 1 evaluating the Viterbi algorithm on any sequence in $\{H, T\}^{1000}$. The probability of any such sequence being produced by the model is $P(s|M) = 2^{-1000}$. Running this example with standard math packages causes underflow. Replacing the probabilities by their logs to some small base is an obvious answer to this problem. The problem one confronts in implementing this trick is how to compute $\log(a+b)$ given $\log a$ and $\log b$. We use the method described by Lee [15]. That method replaces additions on logarithmic representations of the rational numbers greater than zero by a comparison, two additions and a table lookup.

2.5 A monolithic model

A macroscopic view of our model's topology is shown in Figure 2. This model contains some states that are labeled with specific nucleotides; 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. In this way

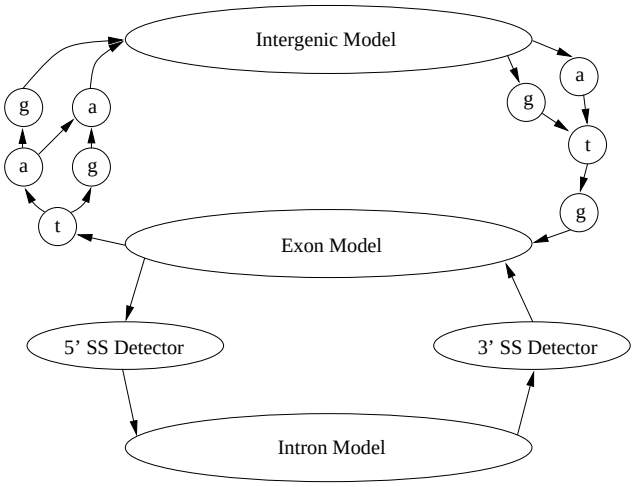

Figure 2: A monolithic model of human genomic DNA. Noncircular nodes indicate regions of more detail which have been omitted. `GTG` is an extremely rare start codon.

we loaded certain parts of the model with biological knowledge. In particular, 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 for those regions. The probabilities in the splice junctions were based on the consensus splice site summaries given in [16], except that a provision was made for nonconsensus splice sites by changing zero probability base-site occurances to a very small probability. Even though the splice junctions were encoded explicitly, it is possible for *any* region to be labeled as a splice site if the sequence fits the rest of the model well enough.

To train our HMM, we separated out the exons, introns, and other non-coding regions in the training set, and trained three smaller models separately. Two of these three models are represented schematically in Figure 2, which only shows them as large ovals, omitting details. The intergenic model was very simple, containing two disconnected 10 state chains (representing the upstream and downstream intergenic pieces) with loops on the ends to absorb extra bases. The exon model contained 89 states and the intron model contained only 10. A more detailed view of the exon and intron models is provided in Fig. 3. Triangles provide a collapsed view of subtrees. Nodes that are labelled can output only the symbol shown in the label, except ϕ which is a null state. Outputs of other nodes are trained. The E-M algorithm was only run on these sub-models, not on the entire combined model.

When we combined the models, we had to add 18 edges. The probabilities on six of those edges were estimated by hand. Another round of training on the combined model would probably help to estimate those edges even better, and this is one of our next

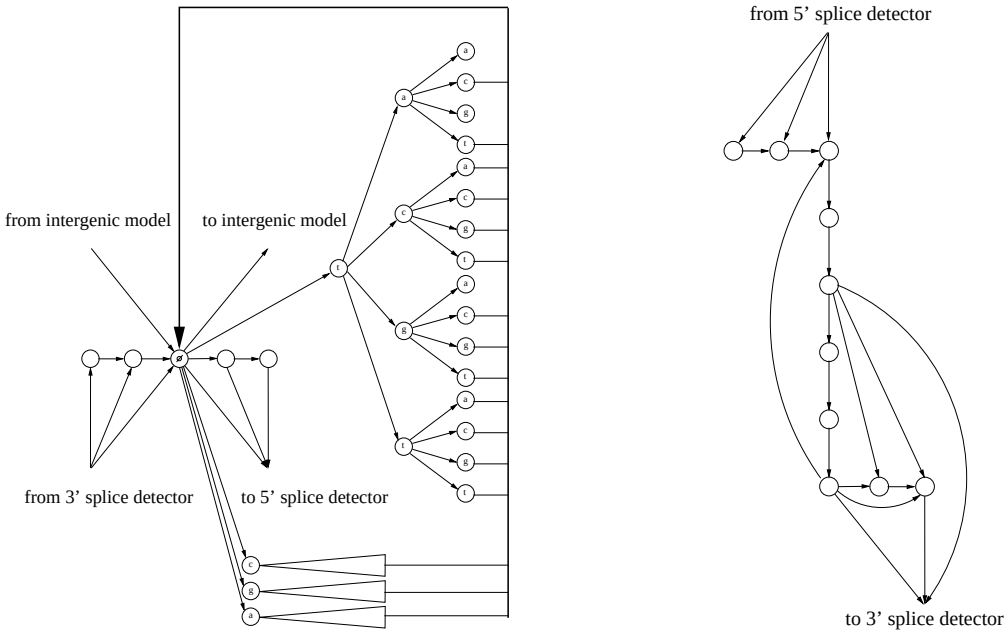

Figure 3: Topologies of the exon (left) and intron models.

steps in this project. The hand-estimated edges are critical, since they are used every time the model switches between the various segment submodels.

2.6 Assumptions used in gene parsing

As stated above, we designed separate HMMs and trained them on exons, introns, and intergenic DNA respectively. Each of these models can then give the probability that a given sequence is entirely of that type (using the Viterbi algorithm). To process anonymous DNA, these models must be glued together using additional states that detect segment boundaries.

There are a number of inter-region constraints and assumptions that can be made for the gene-finding problem. Some of the ones we considered were as follows.

1. The first exon of every gene must begin with a start codon.

2. A gene can have only one stop codon in the same reading frame as the start codon, and that stop codon must appear as the last codon in the gene.

3. Each non-initial exon must be in phase (in a compatible reading frame) with the previous exon. Given that the last base in the exon e_x is at position i, the reading frame for e_x is f ($f \in \{0, 1, 2\}$), and the first base in the exon e_y following e_x is at

position j, reading frame $(j - i - 1 + f)$ mod 3 must be an open reading frame in the current exon [21]. Note that our HMM system does *not* use this assumption.

4. Noncoding regions and introns are flanked by exons.

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

6. There are a number of signals which denote boundaries between exons and introns[18, 16]. We can find all boundary candidates, giving us a superset of the true boundaries, by looking for these signals.

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 the last exon. In other genes, the start codon appears after the start of transcription, and thus occurs in the middle of the first exon. There is some inconsistency in the definition of "gene" and "exon" that has been reflected in the DNA sequence database (GenBank, GSDB) entries; 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. Assumption number 5 is the only difference between the general gene-finding problem and our problem; we assume that exactly one gene exists in the data, which of course is not necessarily true. However, other benchmark experiments to date have also relied on this assumption [5].

Finally, the assumption that exons are in consistent reading frames is important to other dynamic programming systems such as GRAIL [21], because it significantly reduces the number of alternative parses. It leads to two problems, however: (1) it explicitly prohibits the correct characterization of genes that are alternatively spliced, and (2) it makes an algorithm very sensitive to frame shift errors, which occur wherever there are indels in the exon regions. Because an HMM only uses local information, it is difficult to adjust it to keep track of frame shifts; essentially, one would have to duplicate large pieces of the model to represent different frames and phases. We made a conscious decision to ignore frame information, which makes the HMM simpler and avoids problems (1) and (2) above.

2.7 Finding more than one parse

By default, the Viterbi algorithm gives us the single best alignment of a sequence to our model. It is straightforward to modify this to determine the top k best parses by keeping the best k in each state in the Viterbi trellis. (For details of how the trellis is constructed, see [15].) This requires an additional factor of k space, and a factor of $O(k \log k)$ additional time. We have implemented these modifications in our system; however, the results below only include the single best alignment. A more detailed discussion comparing the correct parse to the k best will appear in a forthcoming paper.

3 Assembling a collection of DNA sequences

The training process involves the adjustment of model parameters and adjustment of the model topologies. The final accuracy of these methods depends to a large extent on the quality and quantity of data used to train them. We therefore placed considerable emphasis on the development of a reliable training set. Ad hoc query access to scientific databases via a standard query mechanism is essential for a study of this kind [19]. The Genome Sequence Data Base [7] was therefore the clear choice as our primary source of human DNA sequences. Among the major public nucleotide sequence databases (including Genbank, EMBL Data Library, and DDBJ), GSDB is the only one that supports publicly accessible SQL queries. This feature is of critical importance in obtaining a well-defined set of sequences from the database, and easily updating that set as new data becomes available.

The defining features of all sequences in our data set are as follows:

- obtained from human genomic DNA
- greater than 500 nucleotides in length
- contain a complete coding sequence (CDS), including both 5′ and 3′ ends
- contain at least one exon

Because of GSDB's implementation in a relational database management system (Sybase), pointers to the basic data set can be obtained with a single query. The sequences used in our study were retrieved in August 1995.

In our results below, we always assume that "truth" is given by the annotation from the original GSDB entry. Of course, it is well known that the quality of the entries in the DNA sequence databases varies dramatically (e.g, [12, 20]). Poor quality sequence data can be caused by everything from vector contamination to erroneous annotation of sequence features. Although each of the contributing databases has its own quality control checks, they are not uniformly applied at all sites.

Once our initial data set was obtained, a number of quality controls had to be applied to remove entries which did not meet our standards. We applied filters to the raw data set to check for overlapping exon and intron ranges as well as exons defined out of the range of the coding region. We also manually removed the entry for the human germline T-cell receptor beta chain gene ([10]; accession L36092) from our data set. Although this entry is the largest single human sequence currently in the database, and it is a model for careful, dense annotation, it represents a somewhat atypical sequence because of the V, D, and J segment structure. This, combined with its very large size made it very difficult to process, and ultimately easier to leave out of our data set.

After these quality checks, we were left with 435 complete coding sequences from the GSDB database[7]. A set of 100 sequences was randomly selected and held out for testing. The remaining 335 sequences became our training set, and were used for all training and tuning of the HMM parameters. When all parameters and system

Data set	Num. of sequences	Length in base pairs	Total exon length
Training	335	2440619	543535
Test	100	656803	135258

Table 1: Characterization of the human DNA data.

architecture issues were fixed, the system was used to parse the test set. We report below our results on both the training sequences and the test set.

The final data set is described in Table 1, and a complete list of the sequences used in this study can be found in appendix A.

4 Results

We built our HMM parser as described in section 2.5, and trained it on the 335 human DNA sequences in our training set. As explained above, the training was conducted separately for the exon, intron, and intergenic DNA models, then the models were combined. We plan to train the full model on complete sequences in a followup study, which should give improved results. After training, we applied the HMM to 100 test sequences that we had reserved as a test set from the beginning. The results are summarized in Tables 2 and 3.

Set	Size	TrCDS	PrCDS	TP	TN	TEx	PEx	OvEx	1MEx	Exact
Train	2440954	543566	554068	274183	1617503	2564	2491	1519	1010	419
Test	656903	135267	179985	78398	420049	633	778	426	292	111

Table 2: Overall results of gene parsing.

In Table 2, CDS refers to the coding regions of the data; i.e., the exons. **TrCDS** is the sum of the true CDS lengths, **PrCDS** is the sum of the predicted CDS lengths, **TP** is the total length (in base pairs) of correctly predicted coding (exonic) regions, **TN** is the tototal length (in base pairs) of correctly predicted noncoding (intronic or intergenic) regions, **TEx** is the total number of true exons, **PEx** is the total number of predicted exons, **OvEx** is the number of true exons that are overlapped by a predicted exon, **1MEx** is the number of exons for which we predicted one of the edges exactly, and **Exact** is the number of exons that were exactly predicted.

In Table 3, we summarize the accuracy of the HMM system on the training set and the test set. These numbers are all based on the number of bases correctly predicted.

Set	Sn	Sp	CC	P(I)	P(All)
Train	0.57	0.55	0.36	0.91	0.85
Test	0.66	0.45	0.35	0.84	0.81

Table 3: Per-basepair results of gene parsing.

In Table 3, **Sn** is the sensitivity of the parse: the number of true exon bases in the truth that were correctly predicted divided by the length of the true CDS (computed over the entire set), **Sp** is the sensitivity of the parse: the number of true exon bases that were correctly predicted divided by the length of the predicted CDS, **CC** is the correlation coefficient, **P(I)** is the probability that if a given base is truly an intron we will mark it correctly, **P(All)** is the probability that we will mark a base correctly.

The choice of evaluation measures reported in these tables is based on the excellent comparative study of Burset and Guigo [5], who reported similar measures in a comparison of gene-finding systems on vertebrate DNA sequences.

4.1 A detailed example

We consider a complex gene from the test set, HUMG6PDG (human glucose-6-phosphate dehydrogenase). It is 3262 basepairs long with 2224 coding bases spread across 13 exons. The HMM system parsed the gene into 14 exons, 7 of which matched the true exons and 6 of which overlapped the correct exons. Below is how our system's output.

```
Truth        System      Comment
 473  534     265  534   3' end correct
 605  732     605  732   Exact
 776  813     776  813   Exact
 847  955     847  969   5' end correct
 999 1216     999 1216   Exact
1260 1418    1260 1418   Exact
1462 1587    1462 ...
1618 1711    ...  1711   Missed an Intron
1755 1941    1755 1933   5' end correct
1985 2220    1985 2220   Exact
2264 2340    2264 2340   Exact
2385 2477    2385 2477   Exact
2522 ...     2522 2587
...  ...     2877 2933   Split one exon into three
...  3218    3152 3204
```

5 Conclusions

The overall goal of this project is to provide a new, HMM-based tool for finding genes in eukaryotic DNA. This initial study shows that on a large set of human DNA, HMMs perform well. The HMM system is most accurate at finding regions that substantially overlap true exons, and it also does very well at pinpointing at least one end of each exon. It definitely needs improvement in finding both ends more precisely, and we noticed in some cases (which become apparent when one looks at the details of all 100 test sequences) that some genes are just completely missed. We need to characterize those complete misses (which bring down the overall accuracies substantially) and find ways to avoid them. We have already begun work on several steps in this direction.

There are other signals associated with genes that we have not attempted to utilize. Upstream of the 5' end of the gene there are various promoter sequences whose positions are highly conserved. We can use these signals to locate the start of transcription better. Our current intergenic model is too simple to capture all of this information.

An obvious improvement in an entirely computational method for finding genes is to use the megabases of genes that have already been sequenced. High-scoring homologies to DNA in databases can help pin down the exons in a gene. We can run a local alignment search algorithm against known genes and use the results to restrict the resulting segments from aligning with the intronic or intergenic parts of the monolithic model.

With a simple modification, the HMM architecture can be used to perform whole genome and arbitrary sequence parsing. As stated in section 2.6, our current implementation requires the sequence to begin and end its alignment in the intergenic region. By allowing any state to be a valid initial and final state, and connecting the intergenic region, we can permit the sequence to align to multiple copies of the HMM. This is equivalent to allowing the model to find as many (or as few) genes as are needed.

Finally, there are many technical issues which need to be dealt with when using larger HMMs, which tend to take a very long time to train and which are difficult to modify. We plan to work on parallel implementations to speed up training, and to work on better user interfaces so that the models can be built and modified easily.

Acknowledgements

Thanks to Simon Kasif for many helpful suggestions. This material is based upon work supported by the National Science foundation under Grant Nos. IRI-9116843 and IRI-9223591, and by a Young Faculty Research Initiative grant from the G.W.C. Whiting School of Engineering at Johns Hopkins University.

References

[1] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions 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. In *Proc. Natl. Acad. Sci. USA*, volume 91, pages 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*, 1996. accepted.

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

[7] M. Cinkosky, J. Fickett, and G. Keen. A new design for the genome sequence data base. *IEEE Engineering in Medicine and Biology*, in press 1995.

[8] A. Delcher, S. Kasif, H. Goldberg, and W. Hsu. Probabilistic prediction of protein secondary structure using causal networks. In *Proc. of the Eleventh National Conf. on Artificial Intelligence (AAAI-93)*, pages 316–321. AAAI Press, July 1993.

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

[10] L. Hood, B. Koop, L. Rowen, and K. Wang. Human and mouse t-cell-receptor loci: the importance of comparative large-scale DNA sequence analyses. *Cold Spring Harb. Symp. Quant. Biol.*, 58:339–348, 1993.

[11] 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.

[12] T. Kristensen, R. Lopez, and H. Prydz. An estimate of the sequencing error frequency in the DNA sequence databases. *DNA Seq.*, 2(6):343–346, 1992.

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

[16] 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.

[17] 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.

[18] 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.

[19] M. Waterman, E. Uberbacher, S. Spengler, F. Smith, T. Slezak, R. Robbins, T. Marr, D. Kingsbury, P. Gilna, C. Fields, K. Fasman, D. Davison, M. Cinkosky, P. Cartwright, E. Branscomb, and H. Berman. Genome informatics I: Community databases. report of the invitational DOE workshop on genome informatics. *J. Computational Biology*, 1:173–190, 1994.

[20] O. White, T. Dunning, G. Sutton, M. Adams, J. Venter, and C. Fields. A quality control algorithm for DNA sequencing projects. *Nucleic Acids Res*, 21(16):3829–3838, 1993.

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

A Genes Used in this study

The sequences are referenced by accession number, except when the CDS spanned multiple entries. For those sequences we list the segment name which is shared by the entries.

The following sequences were in the training set for this study:
HUM2C18X, HUM2C9X, HUM3BHSD, HUM4F2HG, HUMA1AR, HUMADH, HUMADH2S, HUMADPRT, HUMADRDO, HUMALAD, HUMALDC, HUMALIDN, HUMALRED, HUMAMYB, HUMANFZ, HUMAPB, HUMAPOAI, HUMAS, HUMATCT, HUMATPK, HUMATPSY, HUMB7AN, HUMBAT2B, HUMBAT3B, HUMBHA, HUMBHSD, HUMBNSP, HUMBPGM, HUMBTF, HUMBTKB, HUMC, HUMCA, HUMCALCR, HUMCANP, HUMCAVII, HUMCCK, HUMCD19W, HUMCD3E, HUMCETP, HUMCFCGR, HUMCFTRA, HUMCFXII, HUMCLG4Q, HUMCR1SF, HUMCRABP, HUMCRP, HUMCSF1M, HUMCYAR, HUMCYPBX, HUMCYPX, HUMDCN, HUMDHLP, HUMDSF, HUMEBI, HUMELAM, HUMETN, HUMFERHC, HUMFIXG, HUMFSHBQ, HUMFUCAS, HUMGALC, HUMGCB, HUMGFI, HUMGFIAB, HUMGHRA, HUMGOAQ, HUMHD,

HUMHOX, HUMHPARS, HUMIDSGE, HUMIGFBP, HUMIGHBP, HUMITILC, HUMKALR, HUMLACI, HUMLAM, HUMLB2A, HUMLBR, HUMLCA, HUMLCT, HUMLDLR, HUMLI2D, HUMLPACI, HUMMANR, HUMMAOB, HUMMCCPA, HUMMHB51, HUMMHCD1, HUMMHCP, HUMMHDB, HUMMHDQA, HUMMHDXA, HUMMHSXB, HUMMSX, HUMNADHC, HUMNITOX, HUMNPYY1, HUMNQO, HUMPALF, HUMPDS, HUMPECAM, HUMPOMC, HUMPPTRH, HUMPRC, HUMPS, HUMPSPS, HUMRASFA, HUMRASK, HUMRBS, HUMS100B, HUMSCN4A, HUMSCNA, HUMSGLT, HUMSGP, HUMSIALP, HUMSPARC, HUMSPD, HUMSTATH, HUMTCGVA, HUMTGASI, HUMTOP, HUMTPA, HUMTRPM2, HUMTSHBA, HUMTYR, HUMU1RNP, HUMUKI, J00238, J00250, J00271, J00277, J00306, J00315, J02698, J02758, J02843, J02846, J02907, J02933, J03071, J03072, J03252, J03474, J03589, J03756, J03826, J03910, J03930, J04038, J04444, J04469, J04617, J04809, J04982, J04990, J05096, J05253, K00470, K00650, K01884, K02043, K02212, K02401, K02402, K03021, L01665, L03378, L04132, L05072, L07772, L07899, L09190, L10038, L10343, L10347, L10641, L10820, L10822, L12691, L12760, L13391, L13470, L14075, L14565, L14778, L15440, L15533, L17131, L18920, L19546, L19686, L22206, L23982, L24498, L25444, L25597, L27587, L29472, L32754, L33842, L34219, L35485, L36861, L39064, M10277, M11166, M11228, M11319, M11725, M11749, M11880, M12605, M13057, M13058, M13207, M13438, M13792, M14642, M15895, M15958, M16110, M16441, M16446, M16714, M17500, M18000, M18079, M19283, M19364, M19806, M20902, M20903, M21540, M22877, M23091, M23442, M23595, M24097, M24415, M24461, M24689, M26167, M26331, M26434, M26679, M26856, M27132, M27138, M27274, M28130, M28548, M28650, M30135, M30142, M31061, M31303, M31776, M31944, M31951, M32405, M33027, M33189, M33387, M33388, M34046, M34356, M34462, M34482, M35093, M35878, M36121, M36640, M38180, M55270, M57678, M57965, M58050, M58569, M58600, M59316, M59924, M60331, M60332, M60858, M61108, M61170, M61827, M62420, M63420, M63454, M63967, M64269, M64554, M64982, M68516, M68895, M69051, M69137, M72150, M72885, M74179, M74587, M77232, M77481, M79462, M79463, M80468, M80469, M80478, M81651, M81806, M83363, M83665, M84332, M84349, M84472, M84757, M89796, M91036, M91037, M91555, M92444, M92844, M94077, M94250, M94556, M95623, M96233, M96264, M96326, M96759, M97925, M97943, M98447, and M99412.

These sequences appeared in the test set:

HUMADPRF, HUMAGT, HUMALDB, HUMCAIII, HUMCATD, HUMCD, HUMCKMM, HUMCNFAR, HUMCTSE, HUMCYPB, HUMEL, HUMENK, HUMENKB, HUMF13A, HUMFLAP, HUMFOL, HUMFOLLI, HUMG6PDG, HUMGA7A, HUMHBA, HUMHBB, HUMHIS, HUMIRBPG, HUMKERP, HUMMCAD, HUMMUT, HUMP53A, HUMPGAMM, HUMRASR, HUMTNC, HUMTNFR, HUMTPO, HUMUQCR, J00314, J02763, J02986, J05008, J05412, L07287, L08010, L11016, L12690, L23210, L25648, L26261, L28101, L29766, L32831, M11726, M12523, M12967, M15205, M15894, M19159, M20543, M22403, M23178, M24110, M24485, M26857, M27024, M28638, M28879, M30838, M31651, M33494, M33764, M37271, M37818, M37984, M38193, M54883, M55913, M57424, M57506, M57888, M59199, M60830, M61829, M61831, M63391, M63871, M63962, M64231, M68519, M69197, M73255, M77144, M79464, M80185, M81740, M83094, M85276, M86593, M89914, M91463, M94579, M95529, M96955, and M98776.

B Complete test set results

These are the details of the results for the 100 test sequences. The meanings of the field headings are the same as those in Tables 2 and 3.

Locus	SqLen	TrCDS	PrCDS	TP	TN	FP	FN	TEx	PEx	TPE	Sn	Sp	CC	OVE	1ME	P(I)	P(ALL)
HUMADPRF	6789	3595	1308	1166	3052	142	2429	5	9	1	0.32	0.89	0.35	5	3	0.96	0.62
HUMAGT	2595	1156	1307	1080	1212	227	76	10	10	5	0.93	0.83	0.77	9	7	0.84	0.88
HUMALDB	14893	999	9	0	13885	9	999	7	1	0	0.00	0.00	0.01	0	0	1.00	0.93
HUMCAIII	2927	445	859	444	2067	415	1	4	4	1	1.00	0.52	0.65	4	3	0.83	0.86
HUMCATD	7193	1920	3854	1638	3057	2216	282	8	21	2	0.85	0.43	0.38	8	5	0.58	0.65
HUMCD	4463	1318	608	364	2901	244	954	6	2	0	0.28	0.60	0.26	2	0	0.92	0.73
HUMCKMM	5321	1013	1585	1009	3732	576	4	6	7	1	1.00	0.64	0.74	6	5	0.87	0.89
HUMCNFAR	3510	1566	1778	931	1097	847	635	10	9	1	0.59	0.52	0.16	7	5	0.56	0.58
HUMCTSE	3030	958	1090	791	1773	299	167	7	7	4	0.83	0.73	0.66	6	5	0.86	0.85
HUMCYPB	5267	1512	3345	1398	1808	1947	114	9	11	1	0.92	0.42	0.38	8	3	0.48	0.61
HUMEL	5140	674	3597	595	1464	3002	79	11	11	0	0.88	0.17	0.16	10	7	0.33	0.40
HUMENK	2068	666	888	666	1180	222	0	1	1	0	1.00	0.75	0.79	1	1	0.84	0.89
HUMENKB	4324	3575	1097	1097	749	0	2478	3	4	1	0.31	1.00	0.27	3	1	1.00	0.43
HUMF13A	7141	2199	2337	1828	4433	509	371	14	14	6	0.83	0.78	0.72	12	10	0.90	0.88
HUMFLAP	3683	397	191	74	3169	117	323	4	2	0	0.19	0.39	0.21	1	0	0.96	0.88
HUMFOL	5861	437	354	44	5114	310	393	5	1	0	0.10	0.12	0.05	1	0	0.94	0.88
HUMFOLLI	5966	1035	1136	701	4496	435	334	6	7	1	0.68	0.62	0.57	4	4	0.91	0.87
HUMG6PDG	3262	2224	2007	1755	786	252	469	13	14	7	0.79	0.87	0.52	13	13	0.76	0.78
HUMGA7A	4218	1274	291	0	2653	291	1274	8	2	0	0.00	0.00	0.18	0	0	0.90	0.63
HUMHBA	19858	584	8165	355	11464	7810	229	3	35	1	0.61	0.04	0.07	2	1	0.59	0.60
HUMHBB	4253	528	336	315	3704	21	213	4	2	1	0.60	0.94	0.72	2	2	0.99	0.94
HUMHIS	6669	470	51	0	6148	51	470	5	1	0	0.00	0.00	0.02	0	0	0.99	0.92
HUMIRBPG	4476	4289	3693	3693	187	0	596	4	4	1	0.86	1.00	0.45	3	2	1.00	0.87
HUMKERP	5063	1421	1910	1072	2804	838	349	8	11	2	0.75	0.56	0.49	5	4	0.77	0.77
HUMMCAD	3616	2192	96	75	1403	21	2117	12	1	0	0.03	0.78	0.06	2	0	0.99	0.41
HUMMUT	5878	2798	38	23	3065	15	2775	13	2	0	0.01	0.61	0.02	1	0	1.00	0.53
HUMP53A	3116	2625	963	952	480	11	1673	11	7	3	0.36	0.99	0.27	7	6	0.98	0.46
HUMPGAMM	4363	839	2727	799	1596	1928	40	3	10	0	0.95	0.29	0.33	3	1	0.45	0.55
HUMRASR	5684	657	2035	504	3496	1531	153	6	8	1	0.77	0.25	0.31	5	4	0.70	0.70
HUMTNC	2856	678	973	534	1739	439	144	6	6	1	0.79	0.55	0.53	6	3	0.80	0.80
HUMTNFR	4013	2120	1995	1356	1254	639	764	10	12	2	0.64	0.68	0.30	10	8	0.66	0.65
HUMTPO	30926	2708	9475	2019	20762	7456	689	15	42	3	0.75	0.21	0.30	12	6	0.74	0.74
HUMUQCR	1726	1203	273	214	464	59	989	2	1	0	0.18	0.78	0.08	1	0	0.89	0.39
J00314	5117	1335	2121	1335	2996	786	0	4	8	3	1.00	0.63	0.71	4	4	0.79	0.85
J02763	3671	273	1251	267	2414	984	6	2	7	0	0.98	0.21	0.38	2	2	0.71	0.73
J02986	6616	621	2134	555	4416	1579	66	3	9	0	0.89	0.26	0.39	3	2	0.74	0.75
J05008	12461	2033	804	0	9624	804	2033	5	6	0	0.00	0.00	0.12	0	0	0.92	0.77
J05412	4251	501	66	0	3684	66	501	5	1	0	0.00	0.00	0.05	0	0	0.98	0.87
L07287	2704	173	438	168	2261	270	5	1	1	0	0.97	0.38	0.57	1	0	0.89	0.90
L08010	3411	769	433	257	2466	176	512	6	3	1	0.33	0.59	0.34	2	2	0.93	0.80
L11016	6305	894	1258	679	4832	579	215	4	5	0	0.76	0.54	0.57	3	2	0.89	0.87
L12690	3710	452	15	0	3243	15	452	3	1	0	0.00	0.00	0.02	0	0	1.00	0.87
L23210	2200	179	891	103	1233	788	76	1	3	0	0.58	0.12	0.10	1	0	0.61	0.61
L25648	2378	601	1004	332	1105	672	269	4	4	1	0.55	0.33	0.15	3	1	0.62	0.60
L26261	12120	1288	1902	259	9189	1643	1029	9	8	1	0.20	0.14	0.04	3	1	0.85	0.78
L28101	9618	1284	3655	1260	5939	2395	24	4	18	1	0.98	0.34	0.49	4	4	0.71	0.75
L29766	24790	1856	4717	986	19203	3731	870	9	22	1	0.53	0.21	0.25	6	5	0.84	0.81
L32831	1262	1075	993	993	187	0	82	1	1	0	0.92	1.00	0.80	1	0	1.00	0.94
M11726	2775	288	1039	263	1711	776	25	3	5	0	0.91	0.25	0.38	2	1	0.69	0.71
M12523	19002	1860	6	0	17136	6	1860	15	1	0	0.00	0.00	0.01	0	0	1.00	0.90
M12967	4016	1089	1841	1089	2175	752	0	7	8	0	1.00	0.59	0.66	7	6	0.74	0.81
M15205	13500	705	2843	492	10444	2351	213	7	19	0	0.70	0.17	0.28	4	2	0.82	0.81
M15894	2740	651	1399	531	1221	868	120	5	5	0	0.82	0.38	0.34	4	4	0.58	0.64

M19159	4268	1608	2113	1608	2155	505	0	11	12	5	1.00	0.76	0.79	11	9	0.81	0.88
M20543	3778	1134	1344	1134	2434	210	0	6	7	5	1.00	0.84	0.88	6	6	0.92	0.94
M22403	6062	2389	2832	1889	2730	943	500	2	8	0	0.79	0.67	0.52	1	0	0.74	0.76
M23178	4102	279	403	91	3511	312	188	3	2	0	0.33	0.23	0.21	1	1	0.92	0.88
M24110	4788	282	447	91	4150	356	191	3	2	1	0.32	0.20	0.20	1	1	0.92	0.89
M24485	4261	722	1295	494	2738	801	228	7	7	0	0.68	0.38	0.37	4	3	0.77	0.76
M26857	4034	1471	2007	1225	1781	782	246	10	11	3	0.83	0.61	0.51	9	7	0.69	0.75
M27024	7393	2922	2303	2030	4198	273	892	11	9	4	0.69	0.88	0.67	9	9	0.94	0.84
M28638	4206	528	174	123	3627	51	405	3	1	0	0.23	0.71	0.36	1	1	0.99	0.89
M28879	4751	744	1483	744	3268	739	0	5	8	1	1.00	0.50	0.64	5	5	0.82	0.84
M30838	4778	747	1559	651	3123	908	96	4	12	0	0.87	0.42	0.50	3	1	0.77	0.79
M31651	6087	1156	1211	792	4512	419	364	8	7	2	0.69	0.65	0.59	6	5	0.92	0.87
M33494	2609	828	1767	742	756	1025	86	5	6	0	0.90	0.42	0.32	5	3	0.42	0.57
M33764	8841	1386	33	0	7422	33	1386	10	1	0	0.00	0.00	0.03	0	0	1.00	0.84
M37271	3280	723	2357	614	814	1743	109	4	9	1	0.85	0.26	0.15	4	3	0.32	0.44
M37818	9697	1862	4328	1440	4947	2888	422	12	24	1	0.77	0.33	0.32	11	5	0.63	0.66
M37984	4567	686	2207	649	2323	1558	37	6	11	1	0.95	0.29	0.39	6	3	0.60	0.65
M38193	4528	890	1587	689	2740	898	201	5	7	0	0.77	0.43	0.44	4	4	0.75	0.76
M54883	3649	1089	1723	1072	1909	651	17	7	10	2	0.98	0.62	0.67	7	5	0.75	0.82
M55913	2140	788	767	552	1137	215	236	4	4	1	0.70	0.72	0.54	3	1	0.84	0.79
M57424	4982	1225	819	598	3536	221	627	4	2	0	0.49	0.73	0.50	2	1	0.94	0.83
M57506	3709	542	153	153	3167	0	389	3	1	0	0.28	1.00	0.50	1	0	1.00	0.90
M57888	4452	988	1702	638	2400	1064	350	5	8	0	0.65	0.37	0.29	4	4	0.69	0.68
M59199	13662	3598	4220	1635	7479	2585	1963	9	20	2	0.45	0.39	0.19	9	6	0.74	0.67
M60830	2158	2078	588	588	80	0	1490	1	1	0	0.28	1.00	0.12	1	0	1.00	0.31
M61829	4705	2537	1850	1625	1943	225	912	1	6	0	0.64	0.88	0.55	1	0	0.90	0.76
M61831	2211	1299	1299	1299	912	0	0	2	2	2	1.00	1.00	1.00	2	2	1.00	1.00
M63391	11990	2220	3622	1733	7881	1889	487	9	18	2	0.78	0.48	0.50	9	8	0.81	0.80
M63871	8673	1527	2824	1328	5650	1496	199	8	14	2	0.87	0.47	0.54	7	4	0.79	0.80
M63962	17201	3556	6980	3341	10006	3639	215	22	29	6	0.94	0.48	0.55	22	15	0.73	0.78
M64231	7623	1649	2511	1274	4737	1237	375	8	14	0	0.77	0.51	0.50	8	3	0.79	0.79
M68519	4732	2139	2033	999	1559	1034	1140	6	11	2	0.47	0.49	0.07	5	3	0.60	0.54
M69197	38542	2268	132	0	36142	132	2268	12	1	0	0.00	0.00	0.01	0	0	1.00	0.94
M73255	5607	3103	2105	1694	2093	411	1409	9	6	1	0.55	0.80	0.39	6	4	0.84	0.68
M77144	9127	1673	9	0	7445	9	1673	4	1	0	0.00	0.00	0.01	0	0	1.00	0.82
M79464	2946	144	2475	144	471	2331	0	1	1	0	1.00	0.06	0.10	1	0	0.17	0.21
M80185	3044	144	2545	144	499	2401	0	1	3	0	1.00	0.06	0.10	1	0	0.17	0.21
M81740	9373	2028	33	0	7312	33	2028	12	1	0	0.00	0.00	0.03	0	0	1.00	0.78
M83094	4452	609	438	438	3843	0	171	2	2	1	0.72	1.00	0.83	2	1	1.00	0.96
M85276	6746	738	2295	610	4323	1685	128	5	13	1	0.83	0.27	0.36	5	4	0.72	0.73
M86593	4663	484	162	114	4131	48	370	5	1	0	0.24	0.70	0.37	1	0	0.99	0.91
M89914	9026	63	8520	63	506	8457	0	1	1	0	1.00	0.01	0.02	1	0	0.06	0.06
M91463	8402	3193	2908	1886	4187	1022	1307	11	10	2	0.59	0.65	0.40	10	5	0.80	0.72
M94579	11502	2347	5495	2228	5888	3267	119	11	23	5	0.95	0.41	0.48	11	8	0.64	0.71
M95529	3401	496	604	331	2632	273	165	3	3	0	0.67	0.55	0.53	2	0	0.91	0.87
M96955	7355	2014	366	0	4975	366	2014	6	2	0	0.00	0.00	0.14	0	0	0.93	0.68
M98776	6005	2369	2176	1844	3304	332	525	9	12	4	0.78	0.85	0.70	9	7	0.91	0.86
Average	6569	1352	1799	783	4200	1015	568	6	7	1	0.61	0.47	0.36	4	2	0.80	0.74
Summary	656903	135267	179985	78398	420049	101587	56869	633	778	111	0.58	0.44	0.35	426	292	0.81	0.76
Locus	SqLen	TrCDS	PrCDS	TP	TN	FP	FN	TEx	PEx	TPE	Sn	Sp	CC	OVE	1ME	P(I)	P(ALL)