In Proc. Fifth Int. Conf. Intelligent Systems for Molecular Biology
Eds. T. Gaasterland *et al.*, p. 179-186, AAAI Press 1997

# Two methods for improving performance of an HMM and their application for gene finding

**Anders Krogh***
Center for Biological Sequence Analysis
Technical University of Denmark
Building 206, 2800 Lyngby, Denmark
and
The Sanger Centre
Wellcome Trust Genome Campus
Hinxton, Cambs, CB10 1SA, UK.

**Abstract**

A hidden Markov model for gene finding consists of submodels for coding regions, splice sites, introns, intergenic regions and possibly more. It is described how to estimate the model as a whole from labeled sequences instead of estimating the individual parts independently from subsequences. It is argued that the standard maximum likelihood estimation criterion is not optimal for training such a model. Instead of maximizing the probability of the DNA sequence, one should maximize the probability of the correct prediction. Such a criterion, called conditional maximum likelihood, is used for the gene finder 'HMMgene'. A new (approximative) algorithm is described, which finds the most probable prediction summed over all paths yielding the same prediction. We show that these methods contribute significantly to the high performance of HMMgene.

**Keywords**: Hidden Markov model, gene finding, maximum likelihood, statistical sequence analysis.

## Introduction

As the genome projects evolve automated annotation of the DNA sequence becomes increasingly important. One of the difficult tasks is to reliably identify genes, in particular in organisms with splicing. Recent successful approaches to this problem include GRAIL (Uberbacher & Mural 1991), NetGene (Brunak, Engelbrecht, & Knudsen 1991), GeneID (Guigo *et al.* 1992), FGENEH (Solovyev, Salamov, & Lawrence 1995), and many other. Several groups are applying probabilistic modeling using hidden Markov models (HMMs) or neural networks; examples are GeneParser (Snyder & Stormo 1995) and Genie (Kulp *et al.* 1996). The two central ideas of most of this work is (1) the integration of several recognition modules into one big model, and (2) for prediction to find the most probable gene given the model.

Hidden Markov models are usually being estimated by maximum likelihood (ML), and this is true also for the HMM based gene finders reported previously (Krogh, Mian, & Haussler 1994; Henderson, Salzberg, & Fasman 1997). The standard ML method optimizes the probability of the observed sequence, but for prediction, one would rather want to maximize the probability of the correct prediction. In this paper a method is described for estimation of a HMM from labeled sequences, which maximizes the probability of the correct labeling, and it is shown that it can substantially improve the performance of a gene finder. Another problem with the standard approach to gene finding is that the prediction is done by finding the most probable state sequence (by the Viterbi algorithm) instead of the most probable gene prediction. I introduce here an algorithm, which is an approximation to the latter, and which always yields a prediction with a probability at least as high as that of the Viterbi algorithm.

These methods are an integral part of HMMgene, which is an HMM for gene prediction. HMMgene has a better performance than almost all other gene finders on human genes. The present paper focuses on the description of the methods, and the results of HMMgene will be used to illustrate the power of these methods, whereas the details of HMMgene will be described elsewhere (Krogh 1997, in preparation).

## The CHMM

The gene finding problem is part of a general class of problems which has a class label associated with each observation. In gene finding the observations are nucleotides and the classes are for instance coding, intron, or intergenic; see Figure 1 for an example of a labeled DNA sequence. Of course, this labeling can be made more elaborate. For instance, 3' and 5' untranslated regions of the mRNA could also be labeled and so could transcription factors and so on. Here however, only the

*Please send correspondence to my current address at Center for Biological Sequence Analysis. Phone: +45 4525 2470. Fax: +45 4593 4808. E-mail: krogh@cbs.dtu.dk


Figure 1: A DNA sequence containing one gene. For each nucleotide its label is written below. The coding regions are labeled 'C', the introns 'I', and the intergenic regions '0'. The shaded areas are the coding regions. (Genbank sequence HUMGALT54X, Homo sapiens galactose-1-phosphate uridyl transferase (GALT) mutant).

three classes mentioned first will be used. I will use label 'C' for coding, 'I' for intron, and '0' for intergenic as in Figure 1.

The traditional approach to this problem is to separately estimate models for sequences representing each of the classes and later merge them into one big model. In (Stormo & Haussler 1994), for instance, a general method is described for estimating relative weights of the individual fixed models so as to optimize the prediction accuracy. However, in the HMM framework it is possible to estimate the whole combined model directly from the labeled sequences if the states of the model are also labeled. In (Krogh 1994) such a model was called a class HMM (CHMM) and it was shown how it is possible to even have a probability distribution over *labels* for each state of the model. Here I will limit myself to one label for each state, which is probably the most useful approach for problems like gene finding.

The starting point is a standard HMM which consists of a set of states each with a probability distribution over letters of the alphabet (A, C, G, and T in this case) and with transition probabilities giving the probability of one state following another one. In the CHMM each state is labeled. For a gene finder it means that some states are labeled 'C' for coding, some are labeled '0' for intergenic, and the rest are labeled 'I' for intron.

One can think of an HMM as emitting sequences of letters. A state emits a letter according to its probability distribution over letters. One can then view a state of a CHMM as 'emitting' its class label along with the letter. Therefore it can emit sequences with associated labels as the one shown in Figure 1.

To estimate the parameters of a CHMM a set of labeled sequences is needed. Let us call a sequence $x = x_1, \ldots, x_L$ and the corresponding labels $y = y_1, \ldots, y_L$. To estimate the parameters of this model by standard ML, one needs to maximize the likelihood of the sequences with their corresponding labels. If the model parameters are called $\theta$, and we assume there is only one sequence in the training set, then the ML solution is

$$\theta^{\mathrm{ML}} = \operatorname*{argmax}_{\theta} P(x, y|\theta). \tag{1}$$

The probability $P(x, y|\theta)$ can be calculated by a trivial modification of the standard forward algorithm (Rabiner 1989; Krogh 1994), where only valid paths through the model are allowed. A valid path is one in which the state labels agree with the observed labels. For instance, a part of the sequence labeled by 'C' for coding can only match a state also labeled 'C'. The same holds for the backward algorithm, and thus the forward-backward (or Baum-Welch) reestimation procedure (Rabiner 1989) can be used to maximize (1).

The model obtained in this way is in most cases identical to a model combined of submodels estimated from each class independently, which is the usual approach, and the main advantage is that the combined model can be estimated in one go from scratch. One additional advantage is that the transitions between the submodels are also automatically estimated in the CHMM. If there are several transitions between submodels for the individual classes the estimation of these is not straightforward when the submodels are trained independently.

When the CHMM has been estimated from the training data, a prediction of labels for a sequence $x = x_1, \ldots, x_L$ can be done the usual way: First the

most probable state path $\pi^* = \pi_1^*, \ldots, \pi_L^*$,

$$\pi^* = \underset{\pi}{\operatorname{argmax}} P(x, \pi|\theta) \tag{2}$$

is found with the Viterbi algorithm (Rabiner 1989). By reading off the label of each state in the path, this is translated to a sequence of labels. Below I will discuss an alternative to this method.

## Conditional Maximum Likelihood

In the ML approach the probability that a submodel can emit the DNA sequences in a certain class (*e.g.* exons) is maximized. Even after maximizing this probability it may well happen that these sequences have a higher probability given a model estimated for a different class (*e.g.* introns). ML is not *discriminative*. The probability of the observed DNA sequence under the model is not the relevant quantity for prediction; one really only cares about getting the predicted labeling correct. Therefore, maximizing the probability of the labeling instead seems more appropriate (Krogh 1994). This is called *conditional* maximum likelihood (CML) in (Juang & Rabiner 1991). Instead of (1) the CML estimate is

$$\theta^{\mathrm{CML}} = \underset{\theta}{\operatorname{argmax}} P(y|x, \theta). \tag{3}$$

This probability can be rewritten as

$$P(y|x, \theta) = \frac{P(x, y|\theta)}{P(x|\theta)}. \tag{4}$$

The numerator is the same probability as in (1) and can be calculated as described above. The denominator is even simpler: it is the probability of the unlabeled DNA sequence, *i.e.*, the probability usually calculated in an HMM disregarding any labeling. This probability is calculated by the standard forward algorithm (Rabiner 1989).

The probability $P(x, y|\theta)$ is a sum over all valid paths through the model and $P(x|\theta)$ is the same *plus* the probability over all the invalid paths, so $P(x, y|\theta) \leq P(x|\theta)$. Therefore, maximizing (4) corresponds to making $P(x, y|\theta)$ come as close as possible to $P(x|\theta)$, *i.e.*, to minimize the probability of all the invalid paths.

To actually maximize (4) is more tricky than the maximization of (1), because an expectation-maximization (EM) algorithm like the forward-backward algorithm does not exist. In this work the extended Baum-Welch algorithm (Normandin & Morgera 1991) is used. Both this algorithm and various gradient descent algorithms that were tested require some parameter tuning, as they easily become unstable.

The conditional maximum likelihood method is obviously not limited to HMMs. Indeed it has proven successful in a neural network/HMM hybrid called a hidden neural network or HNN (Riis & Krogh 1997). The difference between this method and the previously mentioned method of (Stormo & Haussler 1994) is that the parameters of the submodels are estimated along with the weights between them.

## The most probable labeling

Predicting the labels of an unknown sequence is called decoding in speech recognition, where almost all the HMM algorithms originate. The most common decoding algorithm is the Viterbi algorithm, which finds the most probable path through the model as already described. If there are several states with the same label there are usually many paths that give the same labeling. Therefore it would be more natural to sum the probabilities of all these paths and find the overall most probable *labeling* instead. If there are many possible paths for the same labeling, this is quite different from the Viterbi decoding.

There is no exact algorithm for finding the most probable labeling, but here I will present an approximative algorithm which gives very good results. It is a variant of the N-best algorithm (Schwarz & Chow 1990). A partial hypothesis $h_i$ is a labeling of the sequence up to nucleotide number $i$ in the sequence. Assume the probability of this hypothesis $\gamma_k(h_i)$ is known for every state $k$. If the last label (the $i$th) is 'C', this probability can only be non-zero for states also labeled by 'C', and similar for other labels. With three classes, as we use here for gene finding, this hypothesis can spawn three new hypotheses for the next nucleotide $(i + 1)$ in the sequence: $h_i$ with a 'C', an 'I', or a '0' appended. We call these hypotheses $h_iC$, $h_iI$, and $h_i0$. The probability of these three new hypotheses are found by propagating the probabilities $\gamma_k(h_i)$ forward as in the forward algorithm,

$$\gamma_l(h_i Y_l) = \left[\sum_k a_{kl}\gamma_k(h_i)\right] b_l(x_{i+1}), \tag{5}$$

where the label of state $l$ is called $Y_l$ and $a_{kl}$ is the probability of a transition from state $k$ to $l$. The emission probability of base number $i+1$ in state $l$ is called $b_l(x_{i+1})$.

In the 1-best algorithm the best partial hypothesis for each state is kept for position $i$ in the sequence ('best' of course means the one with the highest probability in that state). These hypotheses are then propagated to the next position $i + 1$, and for each state the best is selected again. This continues to the end of the sequence, where the best overall hypothesis is the final answer. In summary the 1-best algorithm works like this:

1. Propagate the empty hypothesis forward to all states $(i = 1)$. At this stage there are 3 different hypotheses '0', 'C', and 'I'. In state $l$ the probability is $\gamma_l(h_i) = a_{0l}b_l(x_1)$, where $h_i$ is one if the hypotheses and $a_{0l}$ is the probability of starting in state $l$.
2. Propagate the hypotheses forward yielding three new hypotheses for every old one, and a probability $\gamma_l(h_i)$ given by (5) in each state for these hypotheses.
3. In each state, choose the hypothesis with the highest probability. Discard all hypotheses that were not

chosen in any state. If you have not reached the end of the sequence, go to step 2.

4. Sum the probabilities for each hypothesis over the states and save the one with the highest.

If a model has a one-to-one correspondence between a path and a labeling, the result of 1-best would be identical to the result of the Viterbi algorithm. Generally, the 1-best algorithm will always produce a labeling with a probability at least as high as the result of the Viterbi algorithm, so in this sense it can never be worse.

The main draw-back of 1-best is that the computation time is significantly longer than Viterbi. However, the number of active hypotheses can be reduced in various ways by discarding ones with a probability less than a cut-off. An advantage of the algorithm is that the memory requirement is only weakly dependent on sequence length; it depends on the number of states in the model times the number of exons in the hypotheses. This is because there is no back tracking as in the Viterbi algorithm, and the hypotheses use very little memory because one just have to store the positions of the borders between labels. The Viterbi algorithm requires memory proportional to sequence length times number of states, although there are more memory efficient implementations which then takes more time. The low memory requirement is particularly nice for parsing very long genomic sequence, where the simple Viterbi algorithm requires megabytes of memory.

The 1-best algorithm will fail to find the correct result if the best hypothesis does not have the highest probability in any state of the model for a given position in the sequence. This seems to be a rare phenomenon in HMMgene. In the algorithm I tried to keep also the partial hypothesis with the largest probability summed over all states, but this did not change the result in a small number of test experiments.

Although I call the algorithm 1-best, it is not identical to the one described in (Schwarz & Chow 1990). In speech recognition, one is interested only in the short sequence of phonemes represented by a speech signal. For gene parsing it would correspond to collapsing the labeling of the sequence in Figure 1 to 'intergenic, coding, intron, coding, intergenic' or '0CIC0'. Therefore, in the original $n$-best algorithm the hypotheses are of this form. The number of hypotheses for each state can be greater than one, and this can be used when searching for suboptimal predictions. This will be the subject of future study.

## HMMgene

A CHMM gene finder called HMMgene is currently being developed. It consists of states modeling coding regions including a crude length modeling, states modeling splice sites and introns, and states for modeling intergenic regions including states for the region upstream of the start codon and downstream of the stop codon.

Two very important features of the model may not be familiar to all readers, even if they know HMMs. Instead of having the usual '0th order' emission probability distribution over letters of the alphabet, I allow a state to have the emission probability conditioned on the $n$ previous letters in the sequence, which corresponds to an $n$th order Markov chain. These $n$th order states are particularly useful for modeling coding regions. Here the basic model consists of three 4th order states, which is essentially like the inhomogeneous Markov chains used to model coding regions in GeneMark (Borodovsky & McIninch 1993). Most other states in the model are first order, *i.e.*, capturing dinucleotide statistics, except for the state modeling internal introns and the one for intergenic regions which are both 3rd order. Notice that the state sequence is still first order, and therefore the high order states do not change the HMM formalism significantly, and all the standard algorithms are unchanged.

The second feature that may be unfamiliar is called *tying*, which is used extensively in speech recognition. (The same technique is called 'weight-sharing' for neural networks.) Tying of two states means that the emission probabilities and/or the transition probabilities are always identical in the two states. During estimation a group of tied states are updated by the sum of the changes calculated for the individual states in the group, so it is like having the same state appearing several places in the model. This is used for the intron modeling. In order to incorporate splicing constraints it is necessary to have three copies of the intron model, and the states in these three models are tied. It is also used to model the exon length by having several tied copies of the basic three states for modeling a codon.

In this work the models were regularized by the simple pseudocount method (see *e.g.* (Krogh *et al.* 1994)). After a few initial experiments these pseudocounts were set to 25 for each letter, except in the high order states for the coding regions where a larger pseudocount of 250 for each letter was used to avoid over-fitting. When reestimating an $n$th order state, the expected number of each $(n+1)$-mer is calculated to obtain the conditional probabilities. For such states the regularizer is spread evenly over all $(n+1)$-mers, *i.e.*, the pseudocount for a letter is divided by $4^n$. In the beginning of the training a decreasing amount of noise was added as described in (Krogh *et al.* 1994) in an attempt to avoid unfavorable local minima. Because the models used in this study were of relatively low complexity, the local minimum problem was not severe, and models trained on the same data did not differ much in performance.

To speed up training, the ML estimated model was used as the initial model for conditional ML. During the iteration of the extended Baum-Welch algorithm the model accuracy on the *training set* was monitored, and after a maximum number of iterations, the model

with the highest accuracy was chosen. It would probably be even better to choose the model from some independent validation set, but it was not tried because of the limited amount of data. The maximum number of iterations was 20.

## Results

The conditional maximum likelihood estimation method and the 1-best decoding algorithm were tested on HMMgene. A dataset collected by David Kulp and Martin Reese (Kulp *et al.* 1996; Reese *et al.* 1997) from GenBank release 95.0 was used (see also the Web site http://www-hgc.lbl.gov/pub/genesets.html). It contains 353 human genes with at least one intron. The set only includes genes with correct start and stop codons as well as consensus splice sites, and strong homologies have been weeded out. The tests were done using 10 fold cross validation, *i.e.*, the set was split at random into 10 subsets of approximately equal size. The model was estimated from 9 of the subsets and tested on the last. After testing on all the test sets the numbers were averaged.

Six numbers were calculated for evaluating performance:

| | |
|---|---|
| Base sensitivity: | The percentage of bases in coding region that are correctly predicted as coding |
| Base specificity: | The percentage of bases predicted as coding that are actually coding. |
| Exon sensitivity: | The percentage of coding exons that are correctly predicted. |
| Exon specificity: | The percentage predicted coding exons that are correct. |
| Missing exons: | Number of real coding exons that are not overlapping a predicted one. |
| Wrong exons: | Number of predicted coding exons that are not overlapping a real one. |

By a correct exon prediction is meant that the exon is exactly as labeled in the database, *i.e.*, both splice sites have to be correctly assigned (or a splice site and a stop or start correct for initial or terminal exons).

To test the effect of conditional maximum likelihood and 1-best decoding, I did the experiment in steps. First a model was estimated in the usual way with ML and the performance tested using the Viterbi algorithm. Results after 10-fold cross validation are shown in the first row of Table 1. The 1-best decoding yields only insignificant (or no) improvements at this stage, and the results are not shown.

Starting from the ML estimated model training was continued with the extented Baum-Welch algorithm as described earlier. The cross validated results using Viterbi decoding are shown in the second row of Table 1. A quite significant improvement is seen although the base sensitivity decreases slightly and the number of missing exons increases. The results for the same model, but decoded with the 1-best algorithm, are shown next. An increase in sensitivity and a decrease in wrong exons is observed at the expense of a slightly lower sensitivity and more wrong exons.

Note that a decrease in specificity does not always mean a worse prediction. For the sake of argument, assume there are 1000 exons in the test set, and that only one is predicted, and it happens to be correct. Then the exon specificity is 100%. If however, 1000 are predicted of which for instance 70% are exactly correct the specificity is only 70%, even though the prediction is much better by most other measures. Although less extreme, the same phenomenon is seen when comparing the performance of 1-best to Viterbi, see Table 2.

Clearly, the main effect of using the 1-best algorithm is that more exons are predicted, balancing the specificity and the sensitivity better. This is because the exon length modeling allows several different paths through the model for the same labeling, and only when they are summed up will the probability be higher than the probability of labeling the exon as an intron. There is still a tendency to under-predict, *i.e.*, there are more real exons than predicted ones. This shows up also as a large number of missing exons compared to the relatively low number of wrong exons. However, because the data set does not contain long stretches of DNA with no genes, it is likely that the number of wrong exons is underestimated and the specificity overestimated. In other words, it is likely that more false gene predictions will occur when using this and other gene finders on newly sequenced anonymous DNA.

One of the main problems for HMMgene is that it does not do a good job on sequences with a low G/C content, which is a common problem for automated gene fining methods, see *e.g.* (Snyder & Stormo 1995). Therefore the data were split up into four groups with G/C content of less than 40%, between 40% and 50%, between 50% and 60%, and more than 60%. The model was then trained further using these data producing four new models specialized for these ranges of G/C content. Although the predictions are still not very good for low G/C content they do improve, and the overall results are shown in Table 1. The main problem seems to be too little data with low G/C content to reliably estimate a good model.

The final row in Table 1 shows the performance of Genie (Reese *et al.* 1997) on the same data for comparison. In Genie (as opposed to HMMgene) the model is constrained to predicting exactly one complete gene in each sequence, which of course is unrealistic if gene finding in anonymous DNA is the goal. However, I would imagine that the results are not strongly dependent on this fact. Although the number of missing

| Performance in % | Base | | Exon | | | |
|---|---|---|---|---|---|---|
| Genie data | Sens. | Spec. | Sens. | Spec. | Miss. | Wrong |
| ML Estimated | 81 | 78 | 58 | 65 | 24 | 15 |
| CML Estimated | 79 | 95 | 61 | 82 | 27 | 4 |
| CML with 1-best | 82 | 94 | 64 | 79 | 23 | 6 |
| G/C adaption | 85 | 93 | 69 | 76 | 17 | 9 |
| Genie | 82 | 81 | 65 | 64 | 14 | 21 |

Table 1: Performance of HMMgene trained by the standard ML method and the conditional ML method described in this paper (first two rows). The third row shows the performance when using 1-best decoding. The fourth row are the results obtained when using several models trained for different G/C content. The last row shows the performance of the Genie gene finder for comparison.

| | Viterbi | 1-best |
|---|---|---|
| Number of exons | 2107 | 2107 |
| Number of predicted exons | 1588 | 1711 |
| Correctly predicted exons | 1295 | 1354 |
| Exon sensitivity (%) | 61.46 | 64.26 |
| Exon specificity (%) | 81.55 | 79.14 |
| Missing exons | 565 (26.82%) | 491 (23.30%) |
| Wrong exons | 61 ( 3.84%) | 103 ( 6.02%) |

Table 2: The numbers for calculating the exon sensitivity and specificity for Viterbi decoding and 1-best decoding. The main difference is that 1-best predicts more exons. This lowers the specificity but increases sensitivity.

exons is lower for Genie, the over-all performance of HMMgene (even before G/C adaption) is better.

Finally, in Table 3 a comparison to some other gene finders is shown. For this comparison HMMgene was trained on the full dataset but tested on the dataset of vertebrate genes described in (Burset & Guigo 1996). In the comparison of gene finders in (Burset & Guigo 1996) FGENEH performed the best of gene finders that did not use homology information. The overall best in the test was GeneID+ which uses protein homology information. Genie can also use homology information (Kulp *et al.* 1997) and those numbers are also shown. Surprisingly HMMgene performs essentially as well as the methods using homology information. However, caution is needed when interpreting these results, because the various methods were trained on different data sets with varying degree of homology with the Burset/Guigo test set.

I recently became aware of two new similar gene finders. One called VAIL is HMM based (Henderson, Salzberg, & Fasman 1997). The main differences with HMMgene are that VAIL does not use CML estimation, it uses Viterbi for prediction, and it does not use high order states. The performance of VAIL is not as high as HMMgene, and I believe that the most important reason is that the model of coding regions is simpler. The other method called GENSCAN is a generalized HMM like Genie, which have very impressive performance (Burge & Karlin 1997). The Genie data set was used for training GENSCAN, but it was supplemented by a large number of cDNA sequence for training the coding region submodels, so the results are not directly comparable. One of the problems with HMMgene is the lack of data for estimating the model for low GC content, and I expect that it can reach a performance similar to GENSCAN, if more training data is used.

## Conclusion

Two methods are described which can significantly improve recognition accuracy of hidden Markov models, and results are shown for an HMM based gene finder for human DNA. The conditional maximum likelihood (CML) estimation criterion is designed to optimize prediction accuracy, and improves the exon sensitivity and (in particular) exon specificity. The 1-best algorithm finds the most probable prediction summed over all possible paths that gives the same prediction. It is an alternative to the commonly used Viterbi algorithm, which only finds the most probable path. It is particularly useful for CML estimated models with many possible paths for the same prediction.

These methods are central to the HMM based gene finder HMMgene, and the results indicate that HMMgene is among the best gene finders for human DNA. One of the advantages of HMMgene is that it is very easy to train and test, because it is one integrated model, and therefore it will be easy to extend to other organisms. HMMgene can be accessed via the World Wide Web:

http://www.cbs.dtu.dk/services/HMMgene/

| Performance in % | Base | | Exon | | | |
|---|---|---|---|---|---|---|
| Burset/Guigo data | Sens. | Spec. | Sens. | Spec. | Miss. | Wrong |
| Not using homology: | | | | | | |
| HMMgene | 88 | 94 | 74 | 78 | 13 | 8 |
| Genie | 78 | 84 | 61 | 64 | 15 | 16 |
| FGENEH | 77 | 88 | 61 | 64 | 15 | 12 |
| Using homology: | | | | | | |
| Genie | 95 | 91 | 77 | 74 | 4 | 13 |
| GeneID+ | 91 | 91 | 73 | 70 | 7 | 13 |

Table 3: A comparison with other gene finders on the Burset/Guigo data. The numbers for FGENEH and GeneID+ are from (Burset & Guigo 1996), and those for Genie are from (Reese *et al.* 1997; Kulp *et al.* 1997).


## Acknowledgements

Discussions with Søren Riis, Richard Durbin, Martin Reese, Søren Brunak, David Haussler, and Frank Eeckman are gratefully acknowledged. Thanks to Martin Reese and Steven Salzberg for showing me their papers before publication, and to Steen Knudsen and David Kulp for comments on the manuscript. David Kulp and Martin Reese are acknowledged for making their data available. This work was supported by the Wellcome Trust and the Danish National Research Foundation.


## References


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

Brunak, S.; Engelbrecht, J.; and Knudsen, S. 1991. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology* 220(1):49–65.

Burge, C., and Karlin, S. 1997. Prediction of complete gene structure in human genomic DNA. *Journal of Molecular Biology* 268. To appear.

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

Guigo, R.; Knudsen, S.; Drake, N.; and Smith, T. 1992. Prediction of gene structure. *Journal of Molecular Biology* 226(1):141–57.

Henderson, J.; Salzberg, S.; and Fasman, K. 1997. Finding genes in DNA with a hidden Markov model. To appear in Journal of Computational Biology.

Juang, B., and Rabiner, L. 1991. Hidden Markov models for speech recognition. *Technometrics* 33(3):251–272.

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

Krogh, A.; Mian, I. S.; and Haussler, D. 1994. A hidden Markov model that finds genes in *e. coli* DNA. *Nucleic Acids Research* 22:4768–4778.

Krogh, A. 1994. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, 140–144. Los Alamitos, California: IEEE Computer Society Press.

Kulp, D.; Haussler, D.; Reese, M. G.; and Eeckman, F. H. 1996. A generalized hidden Markov model for the recognition of human genes in DNA. In States, D.; Agarwal, P.; Gaasterland, T.; Hunter, L.; and Smith, R., eds., *Proc. Conf. on Intelligent Systems in Molecular Biology*, 134–142. Menlo Park, CA: AAAI Press.

Kulp, D.; Haussler, D.; Reese, M. G.; and Eeckman, F. H. 1997. Integrating database homology in a probabilistic gene structure model. In Altman, R. B.; Dunker, A. K.; Hunter, L.; and Klein, T. E., eds., *Proceedings of the Pacific Symposium on Biocomputing*. New York: World Scientific.

Normandin, Y., and Morgera, S. D. 1991. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proc. ICASSP*, 537–540.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77(2):257–286.

Reese, M. G.; Eeckman, F. H.; Kulp, D.; and Haussler, D. 1997. Improved splice site detection in Genie. In Waterman, M., ed., *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB)*. New York: ACM Press.

Riis, S., and Krogh, A. 1997. Hidden neural networks: A framework for HMM/NN hybrids. In *Proceedings of ICASSP'97*. New York, USA: IEEE. To appear.

Schwarz, R., and Chow, Y.-L. 1990. The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses. In *Proceedings of ICASSP'90*, 81–84.

Snyder, E., and Stormo, G. 1995. Identification of protein coding regions in genomic DNA. *Journal of Molecular Biology* 248(1):1–18.

Solovyev, V.; Salamov, A.; and Lawrence, C. 1995. Identification of human gene structure using linear discriminant functions and dynamic programming. In Rawlings, C.; Clark, D.; Altman, R.; Hunter, L.; Lengauer, T.; and Wodak, S., eds., *Proc. of Third Int. Conf. on Intelligent Systems for Molecular Biology*, volume 3, 367–375. Menlo Park, CA: AAAI Press.

Stormo, G. D., and Haussler, D. 1994. Optimally parsing a sequence into different classes based on multiple types of evidence. In *Proc. of Second Int. Conf. on Intelligent Systems for Molecular Biology*, 369–375.

Uberbacher, E., and Mural, R. 1991. Locating protein-coding regions in human DNA sequences by a multiple sensor - neural network approach. *Proceedings of the National Academy of Sciences of the United States of America* 88(24):11261–11265.