Gene finding: putting the parts together*

Anders Krogh
Center for Biological Sequence Analysis
Technical University of Denmark
Building 206, 2800 Lyngby, Denmark

1 Introduction

Any isolated signal of a gene is hard to predict. Current methods for promoter prediction, for instance, will have either a very low specificity or a very bad sensitivity, such that they will either predict a huge number of false positives (fake promoters) or a very small number of true promoters. The same is essentially true for splice site prediction: if looked at in isolation, splice sites are very hard to recognize with good accuracy. This may seem like a contradiction, because there are programs that perform well on this task such as those by Brunak, Engelbrecht, & Knudsen (1991) and Solovyev, Salamov, & Lawrence (1994). The reason for the success is that both of these methods also use the statistics of the coding exon region next to the splice site. Apart from doing a very careful job of describing the regions right around the splice site, they can therefore also rule out splice sites which do not sit next to something looking like a good coding region. In bird-watching the surroundings often gives the necessary clues in deciding which bird you are watching in the distance, whether it is seen in an open field or in a wood for instance. Signal detection in genes is much like bird-watching: it is necessary to take the surroundings into account.

Therefore, to predict something like a splice site, you also need to predict coding exons and vice versa (disregarding the splice sites of introns in untranslated regions). In a long DNA sequence, you probably would not expect to see a coding exon with two associated splice sites unless there are other exons with which it can combine. In this way predictions of the various parts of a gene should influence each other, and prediction of the entire gene structure will also improve on the predictions of the individual signals. Therefore, in the last few years, gene prediction has moved more and more towards prediction of whole gene structures, and these methods typically use modules for recognition of coding regions, splice sites, translation initiation and termination sites, and some even use statistics of the 5' and 3' untranslated regions (UTRs), promoters, *etc.*. This combination of predictions has indeed improved the accuracy of gene prediction considerably, and as more knowledge is gained about transcription and translation, it is likely that the integration of other signals can improve it even further.

*Published in *Guide to Human Genome Computing*, 2nd edition, ed. Martin Bishop, p. 261-274, Academic Press 1998

In this paper I will describe some of the methods for combining the predictions of several signals into a prediction of a complete gene structure. Because most of the gene finding methods are already reviewed in this volume (Milanesi & Rogozin 1997), I will focus on the methods used.

2 Dynamic programming

A variety of methods have been proposed for combining predictions from various sensors into one predicted gene structure. Some of the early methods are GeneModeler (Fields & Soderlund 1990) and GeneID (Guigo *et al.* 1992), which both predict candidate exons and then combine them according to various rules. Many programs use some sort of dynamic programming for the combination (Snyder & Stormo 1993; 1995; Gelfand & Roytberg 1993; Xu *et al.* 1994; Wu 1996; Gelfand, Mironov, & Pevzner 1996; Solovyev, Salamov, & Lawrence 1995; Stormo & Haussler 1994). Also Genefinder uses dynamic programming (Phil Green and Richard Durbin, personal communication). A very simple dynamic programming algorithm will be given first, which represents the essence of many of the algorithms, such as those in (Snyder & Stormo 1993; Stormo & Haussler 1994; Wu 1996). The emphasis will be on the overall principles rather than on the rigorous details.

Assume you have the following functions, which all score a region between base number i and number j in the sequence at hand.

- `internal_exon`(i, j) scores the region for being an internal coding exon.
- `initial_exon`(i, j) scores the region for being the coding part of the first coding exon, which may include a score for the translation initiation signal a few bases upstream of the ATG start codon.
- `terminal_exon`(i, j) scores the region for being the coding part of the last coding exon. Included is a score for translation end.
- `single_exon`(i, j) scores the region for being the coding part of an unspliced gene including scores for translation start and stop.
- `intron`(i, j) scores the region for being an intron. This includes the scores for the splice sites, even if the splice site detectors use the first or last few bases of the neighboring exons outside the interval from i to j.
- `intergenic`(i, j) scores the region for being intergenic. Such a sensor is often omitted, which corresponds to setting the score to zero, so we include it without loss of generality. Note that despite the name this function is also used for scoring the 5' and 3' UTR of the gene. Explicit incorporation of UTR sensors will be addressed later.

Usually the exon score is calculated by adding up some sort of hexamer statistics along the sequence from i to j, as *e.g.* GeneParser (Snyder & Stormo 1993; 1995) and FGENEH (Solovyev, Salamov, & Lawrence 1995), but it can also be neural network based as in GRAIL (Xu *et al.* 1994) and Genie (Kulp *et al.* 1996; 1997). However, it is also possible to use database hits to score exons, which is used in *e.g.* GeneParser, PROCRUSTES (Gelfand, Mironov, & Pevzner 1996), Genie, and GeneWise (Birney & Durbin 1997).

One widely used method for scoring coding regions is the inhomogeneous Markov chain of nth order, which is used extensively in GeneMark (Borodovsky & McIninch 1993). In the simplest case (first order) the 16 conditional probabilities $p_k(a|b)$ for nucleotide a given the previous one (b) in the sequence are needed for the three positions in the reading frame $k = 1, 2, 3$. These can be estimated from a set of known coding regions by simply counting the occurrence of the 16 dinucleotides in each frame and normalizing properly. To score a sequence $x_i, \ldots, x_j$ as being coding and starting in frame 1, one multiplies all these probabilities along the sequence, $p_1(x_i|x_{i-1}) \times p_2(x_{i+1}|x_i) \times \ldots$. By taking the logarithm this score becomes additive. If we let $lp_k(a|b) = \log p_k(a|b)$ the score is

$$lp_1(x_i|x_{i-1}) + lp_2(x_{i+1}|x_i) + lp_3(x_{i+2}|x_{i+1}) + lp_1(x_{i+3}|x_{i+2}) + lp_2(x_{i+4}|x_{i+3}) + \ldots$$

In an inhomogeneous Markov chain of order n, the probability of each base is conditioned on the previous n bases, instead of just 1. For a second order chain there are 64 different conditional probabilities corresponding to the 64 possible trinucleotides. For such a second order chain the score would be

$$lp_1(x_i|x_{i-1}, x_{i-2}) + lp_2(x_{i+1}|x_i, x_{i-1}) + lp_3(x_{i+2}|x_{i+1}, x_i) + \ldots$$

A second order chain spans a full codon, and usually that is the minimum order to use for scoring coding regions. Most gene finders use order 4 or 5, which correspond to pentamer or hexamer statistics.

Usually the same sensor is used for coding potential in the four different exon functions above, so they only differ in the signal sensors they use. The signal scores, such as splice site scores, are usually obtained from position dependent score matrices (*e.g.* GeneParser and FGENEH) or neural networks (*e.g.* Genie). Often these 'signal sensors' are distinguished from the 'content sensors' in the dynamic programming algorithm. Although more natural in some formulations, it does not make any principal difference. Also, it does not make any difference if the score for splice sites is part of the intron score or the exon score.

Let us first consider the problem of combining coding regions optimally into exactly one complete gene without regard to the frame being consistent for the entire gene. It is assumed that the scores are additive, *i.e.*, the optimal gene structure is the one with the largest score added up over all the components. Then a dynamic programming algorithm goes like this: for each position j in the sequence calculate

1. Score for translation start at position j

$$S(j) = \texttt{intergenic}(1, j-1)$$

2. Score for best partial gene up to a donor at position j

$$D(j) = \max_{i<j} \begin{cases} S(i) + \texttt{initial\_exon}(i, j-1) \\ A(i) + \texttt{internal\_exon}(i, j-1) \end{cases}$$

3. Score for best partial gene up to an acceptor at position j

$$A(j) = \max_{i<j} [D(i) + \texttt{intron}(i, j-1)]$$

(j is the first base of the exon).

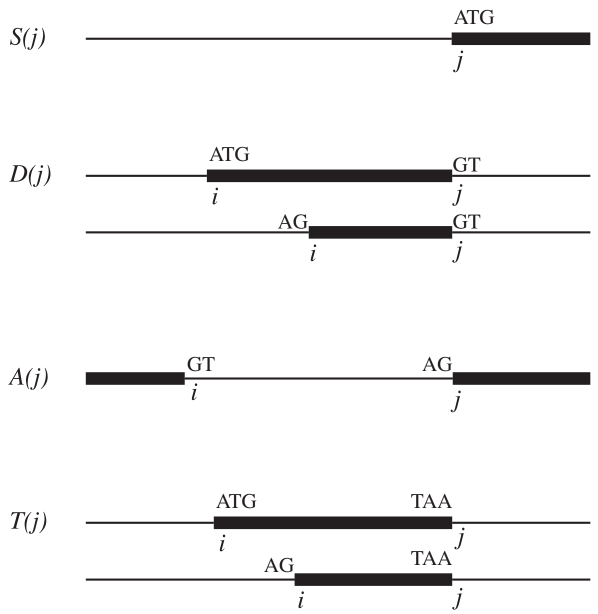

Figure 1: Illustration of the dynamic programming algorithm described in the text. Potential coding regions are shown by a heavy line. Possible splice sites are marked with their consensus pattern (GT for donor and AG for acceptor), possible start codons are marked with ATG, and possible stop codons with TAA (one of the three stop codons).

4. Score for best complete gene up to a stop codon at position j

$$T(j) = \max_{i<j} \begin{cases} A(i) + \texttt{terminal\_exon}(i, j-1) \\ S(i) + \texttt{single\_exon}(i, j-1) \end{cases}$$

(j is the base immediately after the stop codon which covers base $j-3$ to $j-1$).

See fig. 1 for a graphical representation of these terms. These numbers are calculated recursively from the beginning to the end of the sequence, $j = 1, \ldots, L$, and at the end

$$\max_{i<j}[T(i) + \texttt{intergenic}(i, L)] \quad (1)$$

yields the score of the optimal gene structure. To actually find that structure, you need to save the i maximizing the expression in each step of the process as well as which of the terms maximized it (whether it was the term with $A(i)$ or $S(i)$ in the steps where there is a choice). Then starting from the last step one can go backwards through each step and thus reconstruct the gene structure with the highest score.

As it stands, the computation time of the algorithm is proportional to the square of the length of the sequence. However, this can be reduced drastically by only considering valid signals, *e.g.*, only calculate $S(j)$ whenever there is a potential start codon (ATG) at position j, only calculate D and A at positions with potential splice sites, *etc.*. When calculating exon score, one need not consider positions (i) further back than the first stop codon, and since open reading frames are rarely very long, this will also limit the calculation time significantly. These and other constraints make such an algorithm feasible to use even for long sequences.

This is essentially the algorithm used in GeneParser, except that the requirement of exactly one complete gene is relaxed, so that no genes or a partial genes can also be predicted. Usually a genomic DNA sequence contains only parts of a gene or a mixture of whole genes and pieces of genes at the ends. Sometimes a cosmid from the human genome contains no genes or is part of huge intron. It is very easy to extend an algorithm like the one above to predict any number of genes and partial genes. For instance to allow for one or more genes one need only change rule 1 to

$$S(j) = \max_{i<j} \begin{cases} \mathtt{intergenic}(1, j-1) \\ T(i) + \mathtt{intergenic}(i, j-1) \end{cases}$$

This of course is still a fairly simplified algorithm. The second extension one should include is reading frame consistency between exons. This is fairly easy to do by having three different D values D_0, D_1, and D_2 and three different A values A_0, A_1, and A_2. The variables indexed with 0 are used for introns splicing between two codons, the ones indexed with 1 are used for those splicing after the first base in a codon, *etc.*. Now the rules above need to be changed to inforce consistency, so for instance rule 3 would change to

$$A_k(j) = \max_{i<j}[D_k(i) + \mathtt{intron}(i, j-1)]$$

Similar changes are needed for all the other rules that use A or D. Furthermore, the exon scoring functions need to be frame-aware, so one might have a score for each frame k, so $\mathtt{internal\_exon}_k(i, j)$ would mean an internal exon in which the first base corresponds to the k-th codon position. Then rule 2 would become

$$D_l(j) = \max_{i<j} \begin{cases} S(i) + \mathtt{initial\_exon}(i, j-1) & \text{if } l = (j-i) \bmod 3 \\ A_k(i) + \mathtt{internal\_exon}_{k+1}(i, j-1) & k = (j-i+l) \bmod 3 \end{cases}$$

where mod means modulus. The conditions on k and l are necessary for a consistent frame. Similarly rule 4 would change to

$$T(j) = \max_{i<j} \begin{cases} A_k(i) + \mathtt{terminal\_exon}(i, j-1) & \text{if } k = (j-i) \bmod 3 \\ S(i) + \mathtt{single\_exon}(i, j-1) \end{cases}$$

It is also possible to integrate sensors for promoters and the UTRs. This is done by adding new variables to the dynamic programming. For instance one might have a variable I_j for transcription initiation. In this case the translation start depends on that variable, so for instance the $S(j)$ score in rule 1 would have to change to something line $S(j) = \max i < jI_j + \mathtt{5'UTR_score}(i, j-1)$.

Now we have a method to combine sensors. There is, however, one problem that has not been addressed, which is how to weigh the output of the sensors against each other. Usually each sensor is estimated or trained independently of all the others, and they may use completely different scales. Similarly, if more than one measure of coding potential is used, it is likely that they are not equally reliable, and one would like to put a higher weight on the most reliable ones. There are two ways of solving this problem: one is to use fully probabilistic models and the other is to train the weights in some way. The first of these methods will be discussed in the next section. The second method is used in GeneParser, where several sensors are used for the various regions; there are for instance three different measures for coding potential (not counting database hits). The output of these sensors are combined by a neural network which is optimized so as to give the best predictions on a training set. In (Stormo & Haussler 1994) a general method is given for optimizing the weights of the individual sensors.

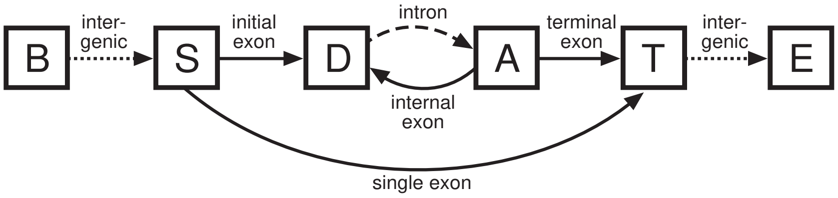

Figure 2: A finite state automaton corresponding to the simple DP algorithm.

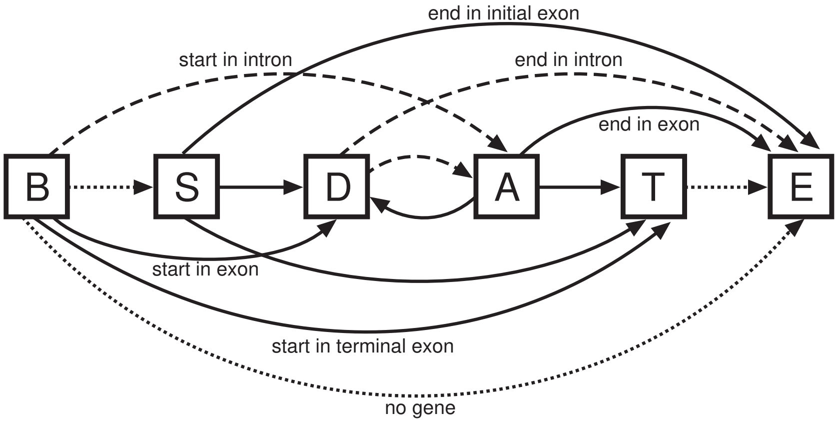

Figure 3: The model of fig. 2 with transitions added that allows for prediction of any number of genes and partial genes where the sequence starts or ends in the middle of an exon or an intron.

3 State models

The idea of combining the predictions into a complete gene structure is that the 'grammatical' constraints can rule out some wrong exon assemblies. The grammatical structure of the problem has been stressed by David Searls (Searls 1992; Dong & Searls 1994) who also proposed to use the methods of formal grammars from computer science and linguistics. The dynamic programming can often be described conveniently by some sort of finite state automaton (Searls & Murphy 1995; Durbin *et al.* 1997). A model might have a state for translation start (S), one for donor sites (D), one for acceptor sites (A), and one for translation termination (T). Each time a transition is made from one state to another a score (or a penalty) is added. For the transition from the donor state to the acceptor state the intron score is added to the total score, and so on. In fig. 2 the state diagram is shown for the simple dynamic programming algorithm above. For each variable in the algorithm there is a corresponding state with the same name, and also a begin and end state is needed.

The advantage of such a formulation is that the dynamic programming for finding the maximum score (or minimum penalty) is of a more general type, and therefore adding new states or new transitions is easy. For instance drawing the state diagram for a more general dynamic programming algorithm that allows for any number of genes and also partial genes is straight-forward (fig. 3), whereas it is a bit involved to write down. Similarly the state diagram for the frame-aware algorithm sketched out above is shown in fig. 4.

If the scores used are log probabilities or log-odds, then a finite state automaton is essentially a hidden Markov model (HMM), and these have been introduced recently into gene

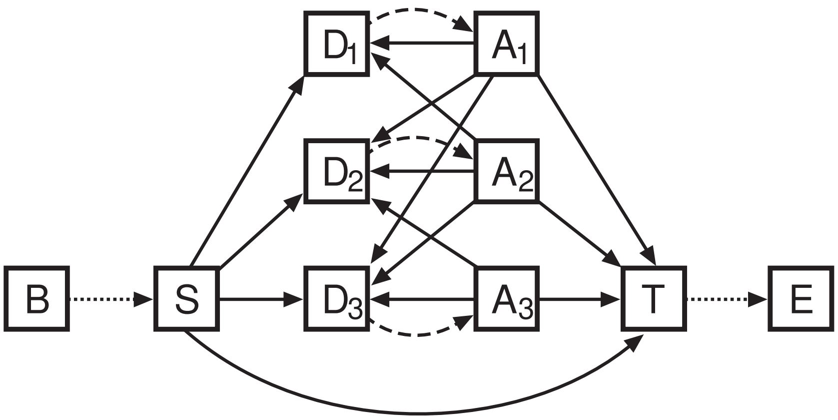

Figure 4: A model that ensures frame consistency throughout a gene. As in the two previous figures, dotted lines correspond to intergenic regions, dashed to introns, and full lines to coding regions (exons).

finding by several groups. The only fundamental difference from the dynamic programming schemes discussed in the previous section is that these models are fully probabilistic, which have certain advantages. One of the advantages is that the weighting problem is easier.

VEIL (Henderson, Salzberg, & Fasman 1997) is an application of an HMM to the gene finding problem. In this model all the sensors are HMMs. The exon module is essentially a first order inhomogeneous Markov chain, which is described above. This is the natural order for implementation in an HMM, because then each of the conditional probabilities of the inhomogeneous Markov chain corresponds to the probability of a transition from one state to the next in the HMM. It is not possible to avoid stop codons in the reading frame when using a first order model, but in VEIL a few more states are added in a clever way which makes the probability of a stop codon zero. Sensors for splice sites are made in a similar way. The individual modules are then combined essentially as in fig. 2, *i.e.*, frame consistency is not enforced. The combined model is one big HMM, and all the transitions have associated probabilities. These probabilities can be estimated from a set of training data by a maximum likelihood method. For combining the models this essentially boils down to counting occurrences of the different types of transitions in the data set. Therefore the implicit weighting of the individual sensors is not really an issue.

Although the way the optimal gene structure is found is very similar in spirit to the dynamic programming above, it looks quite different in practice. This is because the dynamic programming is done at the level of the individual states in all the submodels; there are more than 200 such states in VEIL. Because the model is fully probabilistic, one can calculate the probability of any sequence of states for a given DNA sequence. This state sequence (called a path) determines the assignment of exons and introns. If the path goes through the exon model, that part of the sequence is labeled as exon, if it goes through the intron model it is labeled intron and so forth. The dynamic programming algorithm, which is called the Viterbi algorithm, finds the *most probable* path through the model for a given sequence, and from this the predicted gene structure is derived. (See (Rabiner 1989) for a general introduction to HMMs.)

This probabilistic model gives the advantage of solving the problem of weighting the individual sensors. The maximum likelihood estimation of the parameters can be shown to be optimal if there is enough training data, and if the statistical nature of genes can be described by such a model. A weak part of VEIL is the first order exon model, which is probably not capable of capturing the statistics of coding regions, and most other methods use 4th or 5th order models.

A HMM based gene finder called HMMgene is currently being developed. The basic method is the same as VEIL, but it includes several extensions to the standard HMM methodology, which are described in (Krogh 1997). One of the most important is that coding regions are modeled by a 4th order inhomogeneous Markov chain instead of a first order. This is done by an almost trivial extension of the standard HMM formalism that allows a Markov chain of any order in a state state of the model, whereas the standard HMM has a simple unconditional probability distribution over the four bases (corresponding to zeroth order). The model is frame-aware and can predict any number of genes and partial genes, so the overall structure of the model is as in fig. 4 with transitions added to allow for begin and end in exons and introns as in fig. 3.

As already mentioned, the maximum likelihood estimation method works well if the model structure can describe the true statistics of genes. This is a very idealized assumption, and therefore HMMgene uses another method for estimating the parameters called conditional maximum likelihood (Juang & Rabiner 1991; Krogh 1994). Loosely speaking, maximum likelihood maximizes the probability of the DNA sequences in the training set, whereas conditional maximum likelihood maximizes the probability of the gene structures of these sequences, which, after all, is what we are interested in. This kind of optimization is conceptually similar to the one used in GeneParser, where the prediction accuracy is also optimized. HMMgene also uses a dynamic programming algorithm different from the Viterbi algorithm for prediction of the gene structure. All of these methods have contributed to a high performance of HMMgene.

Genie is another example of a probabilistic state model which is called a generalized HMM (Kulp *et al.* 1996; Reese *et al.* 1997). Figure 4 is in fact Genie's state structure, and both this figure and fig. 2 are essentially copied from (Kulp *et al.* 1996). In Genie the signal sensors (splice sites) and content sensors (coding potential) are neural networks, and the output of these networks are interpreted as probabilities. This interpretation requires estimation of additional probability parameters which work like weights on the sensors. So, although it is formulated as a probabilistic model, the weighting problem still appears in disguise. The algorithm for prediction is almost identical to to the dynamic programming algorithm of the last section. A version of Genie also includes database similarities as part of the exon sensor (Kulp *et al.* 1997).

There are two main advantages of generalized HMMs as compared to standard HMMs. First, the individual sensors can be of any type such as neural networks, whereas in a standard HMM they are restricted by the HMM framework. Second, the length distribution (of *e.g.* coding regions) can be taken into account explicitly, whereas the natural length distribution for an HMM is a geometric distribution, which decays exponentially with the length. However, it is possible to have a fairly advanced length modeling in an HMM if several states are used. The advantage of a system like HMMgene, on the other hand, is that it is one integrated model, which can be optimized all at once for maximum prediction accuracy.

Another gene finder based on a generalized HMM is GENSCAN (Burge & Karlin 1997). The main differences between the GENSCAN state structure and that of Genie or HMMgene is that GENSCAN models the sequence in both directions simultaneously. In many gene finders, such as those described above, genes are first predicted on one strand, and then on the other. Modeling both strands simultaneously was done very successfully in GeneMark, and a similar method is implemented in GENSCAN. One advantage (and perhaps the main one) is that this construction avoids predictions of overlapping genes on the two strands, which presumably are very rare in the human genome. GENSCAN models any number of genes and partial genes like HMMgene. The sensors in GENSCAN are very similar to those used in HMMgene. For instance the coding sensor is a 5th order inhomogeneous Markov chain. The signal sensors are essentially position dependent weight matrices, and thus are also very similar to those of HMMgene, but there are more advanced features in the splice site models. GENSCAN also model promoters and the 5' and 3' UTRs.

4 Performance comparison

Comparing gene finders is difficult. In (Burset & Guigo 1996) several gene finders were compared on a data set consisting of 570 mammalian sequences containing one complete gene with at least one intron. In table 1 the performance on this data set is shown for some of the gene finders mentioned in this paper (see (Milanesi & Rogozin 1997) for numbers on other gene finders). There are several problems with such a comparison. First, most of the gene finders were trained on data that had genes homologous to some of the sequences in this test set. Since this overlap varies among the gene finders the results are hard to compare. Second, whereas the initial test was done by an independent group, most of the numbers shown in the table are obtained by the developers themselves *after* the comparison was published. This makes it possible to keep tuning the method until it performs well on these particular data (only the numbers for FGENEH, GeneID, and GeneParser are from the original study). Finally, it is likely that some of the gene finders in the comparison have been further developed since the test.

To reasonably reliably test gene finders they need to be trained and tested on the same data sets. This would require that all developers agree on a data set and do cross-validation where the gene finder is trained on *e.g.* 9/10 of the data and tested on the remaining 1/10. This process is repeated 10 times until all the data have been used for testing and the performance measures are averaged. As a start in that direction the Genie group has made their data set of almost 400 human genes available for other researchers, and in fact Genie and HMMgene have been cross-validated on this set, see (Reese *et al.* 1997; Krogh 1997) for results. In the Genie data significant homologies between genes have been removed, and it is restricted to sequences with exactly one complete gene. The performance is generally lower on these data than on the Burset-Guigo data. First of all because the training set is non-homologous with the test set because of cross validation, and secondly because the sequences are longer and seems to be generally harder. Hopefully in the future there will be good data sets of reliably annotated genomic sequence for the gene finders to be trained and tested on.

Both Genie and HMMgene were trained on the Genie data set, and therefore the results shown in table 1 are comparable. GENSCAN also used this data set for training, but additionally used a set of almost 2000 complete human cDNA sequences. This much larger data set allows a reliable estimation of the 5th order coding model used, whereas my experiments with HMMgene showed that using a coding model of order higher than 4 based on the Genie data alone was not successful. Furthermore, it is almost certain that there are more homologies

Performance in %	Base		Exon			
Burset/Guigo data	Sn	Sp	Sn	Sp	ME	WE
GeneParser2	66	79	35	40	29	17
GeneID	63	81	44	46	28	24
FGENEH	77	88	61	64	15	12
VEIL	83	72	53	49	19	NA
Genie	78	84	61	64	15	16
HMMgene	88	94	74	78	13	8
GENSCAN	93	93	78	81	9	5

Table 1: A comparison of gene finders on the Burset/Guigo data. These ones do not use database searches. Sensitivity (Sn) and specificity (Sp) are shown at both the single nucleotide level and at the exact exon level. The last two columns show missing exons (ME): the percentage of true exons that do not overlap with a predicted one, and wrong exons (WE): the number of predicted ones that do not overlap a correct one. See (Milanesi & Rogozin 1997) for a discussion of these performance measures. The numbers for GeneParser2, GeneID, and FGENEH are taken from the original comparison (Burset & Guigo 1996), in which FGENEH showed the best performance. Numbers for Genie are from (Reese *et al.* 1997), those for HMMgene from (Krogh 1997), and (Burge & Karlin 1997) has the numbers for GENSCAN. Note that the results for VEIL are for 5-fold cross-validation on the Burset/Guigo data.

between GENSCAN's data set and the test set. GENSCAN stands out as the best in table 1, and there is no doubt that GENSCAN is currently among the best gene finders. However, direct comparison based on these numbers is impossible.

5 Conclusion

In this paper I discussed ways of combining information of splice sites, coding potential, *etc.*, into a prediction of a complete gene structure. It is quite easy to extend most of these methods to include sensors for promoters, 5' and 3' untranslated regions, poly-A site, and so forth. In HMMgene this has only given marginal improvements, and is currently not included. As more knowledge is gained about gene structure and regulation it is quite possible that incorporation of more sensors can significantly improve prediction.

Most current gene finders are developed for DNA sequences with no or very few errors and for 'text book genes' that all have the GT/AG consensus splice sites, no frame shifts, no alternative splicing, and so forth. Many of these complications can be dealt with in most of the models mentioned here, but no doubt at the cost of worse performance. Although some of the problems have been taken up already, these issues are mainly topics of future research.

HMMgene, which was described briefly, is still being developed and the numbers shown have improved somewhat. It is possible to follow the development and test the current version at the web site http://www.cbs.dtu.dk/services/HMMgene/. Links to some of the other gene finders mentioned can be found on my web page http://www.cbs.dtu.dk/krogh/genefinding.html.

Acknowledgements

Several discussions with Richard Durbin about dynamic programming and finite state machines are gratefully acknowledged. I would like to thank David Ussery and Nikolaj Blom for comments on the manuscript. My work is supported by the Danish National Research Foundation.

References

Birney, E., and Durbin, R. 1997. Dynamite: A flexible code generating language for dynamic programming methods used in sequence comparison. In T. Gaasterland et al., ed., *Proc. of Fifth Int. Conf. on Intelligent Systems for Molecular Biology*, 56–64. Menlo Park, CA: AAAI Press.

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* 268:78–94.

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

Dong, S., and Searls, D. B. 1994. Gene structure prediction by linguistic methods. *Genomics* 23:540–551.

Durbin, R. M.; Eddy, S. R.; Krogh, A.; and Mitchison, G. 1997. *Biological Sequence Analysis*. Cambridge University Press. To be published.

Fields, C. A., and Soderlund, C. A. 1990. gm: a practical tool for automating DNA sequence analysis. *Computer Applications in the Biosciences* 6:263–70.

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

Gelfand, M. S.; Mironov, A. A.; and Pevzner, P. A. 1996. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America* 93:9061–9066.

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. H. 1997. Finding genes in DNA with a hidden Markov model. To appear in Journal of Computational Biology.

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

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.

Krogh, A. 1997. Two methods for improving performance of a HMM and their application for gene finding. In T. Gaasterland et al., ed., *Proc. of Fifth Int. Conf. on Intelligent Systems for Molecular Biology*, 179–186. Menlo Park, CA: AAAI 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 D.J. States et al., ed., *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.

Milanesi, L., and Rogozin, B. 1997. Prediction of human gene structure. This volume.

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.

Searls, D. B., and Murphy, K. P. 1995. Automata-theoretic models of mutation and alignment. In C. Rawlings et al., ed., *Proc. of Third Int. Conf. on Intelligent Systems for Molecular Biology*, volume 3, 341–349. Menlo Park, CA: AAAI Press.

Searls, D. B. 1992. The linguistics of DNA. *American Scientist* 80:579–591.

Snyder, E. E., and Stormo, G. D. 1993. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucleic Acids Research* 21(3):607–13.

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

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

Solovyev, V. V.; Salamov, A. A.; and Lawrence, C. B. 1995. Identification of human gene structure using linear discriminant functions and dynamic programming. In C. Rawlings et al., ed., *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.

Wu, T. D. 1996. A segment-based dynamic programming algorithm for predicting gene structure. *Journal of Computational Biology* 3:375–394.

Xu, Y.; Einstein, J. R.; Mural, R. J.; Shah, M.; and Uberbacher, E. C. 1994. An improved system for exon recognition and gene modeling in human DNA sequences. In *Proc. of Second Int. Conf. on Intelligent Systems for Molecular Biology*, 376–384.