Chapter 4

An Introduction to Hidden Markov Models for Biological Sequences

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

Phone: +45 4525 2471
Fax: +45 4593 4808
E-mail: krogh@cbs.dtu.dk

In *Computational Methods in Molecular Biology*, edited by S. L. Salzberg, D. B. Searls and S. Kasif, pages 45-63. Elsevier, 1998.

Contents

4.1 Introduction

Very efficient programs for searching a text for a combination of words are available on many computers. The same methods can be used for searching for patterns in biological sequences, but often they fail. This is because biological 'spelling' is much more sloppy than English spelling: proteins with the same function from two different organisms are almost certainly spelled differently, that is, the two amino acid sequences differ. It is not rare that two such homologous sequences have less than 30% identical amino acids. Similarly in DNA many interesting signals vary greatly even within the same genome. Some well-known examples are ribosome binding sites and splice sites, but the list is long. Fortunately there are usually still some subtle similarities between two such sequences, and the question is how to detect these similarities.

The variation in a family of sequences can be described statistically, and this is the basis for most methods used in biological sequence analysis, see [1] for a presentation of some of these statistical approaches. For pairwise alignments, for instance, the probability that a certain residue mutates to another residue is used in a substitution matrix, such as one of the PAM matrices. For finding patterns in DNA, *e.g.* splice sites, some sort of weight matrix is very often used, which is simply a position specific score calculated from the frequencies of the four nucleotides at all the positions in some known examples. Similarly, methods for finding genes use, almost without exception, the statistics of codons or dicodons in some form or other.

A hidden Markov model (HMM) is a statistical model, which is very well suited for many tasks in molecular biology, although they have been mostly developed for speech recognition since the early 1970s, see [2] for historical details. The most popular use of the HMM in molecular biology is as a 'probabilistic profile' of a protein family, which is called a profile HMM. From a family of proteins (or DNA) a profile HMM can be made for searching a database for other members of the family. These profile HMMs resemble the profile [3] and weight matrix methods [4, 5], and probably the main contribution is that the profile HMM treats gaps in a systematic way.

The HMM can be applied to other types of problems. It is particularly well suited for problems with a simple 'grammatical structure,' such as gene finding. In gene finding several signals must be recognized and combined into a prediction of exons and introns, and the prediction must conform to various rules to make it a reasonable gene prediction. An HMM can combine recognition of the signals, and it can be made such that the predictions always follow the rules of a gene.

Since much of the literature on HMMs is a little hard to read for many biologists, I will attempt in this chapter to give a non-mathematical introduction to HMMs. Whereas the little biological background needed is taken for granted, I have tried to explain HMMs at a level that almost anyone can follow. First HMMs are introduced by an example and then profile HMMs are described. Then an HMM for finding eukaryotic genes is sketched, and finally pointers to the literature are given.

4.2 From regular expressions to HMMs

Most readers have no doubt come across regular expressions at some point, and many probably use them quite a lot. Regular expressions are used in many programs, in particular on Unix computers. In programs like awk, grep, sed, and perl, regular expressions can be used for searching text files for a pattern. With grep for instance, you can search a file for all lines containing 'C. elegans' or 'Caenorhabditis elegans' with the regular expression '`C[\.a-z]* elegans`'. This will match any line containing a C followed by any number of lower-case letters or '.', then a space and then elegans. Regular expressions can also be used to characterize protein families, which is the basis for the PROSITE database [6].

Using regular expressions is a very elegant and efficient way to search for some protein families, but difficult for other. As already mentioned in the introduction, the difficulties arise because protein spelling is much more free than English spelling. Therefore the regular expressions sometimes need to be very broad and complex. Imagine a DNA motif like this:

```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

(I use DNA only because of the smaller number of letters than for amino acids). A regular expression for this is

[AT] [CG] [AC] [ACGT]* A [TG] [GC] ,

meaning that the first position is A or T, the second C or G, and so forth. The term '[ACGT]*' means that any of the four letters can occur any number of times.

The problem with the above regular expression is that it does not in any way distinguish between the highly implausible sequence

```
T G C T - - A G G
```

which has the exceptional character in each position, and the consensus sequence

```
A C A C - - A T C
```

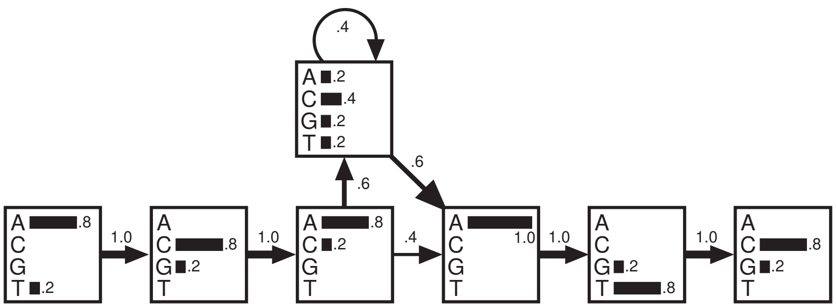

Figure 4.1: A hidden Markov model derived from the alignment discussed in the text. The transitions are shown with arrows whose thickness indicate their probability. In each state the histogram shows the probabilities of the four nucleotides.

with the most plausible character in each position (the dashes are just for aligning these sequences with the previous ones). What is meant by a 'plausible' sequence can of course be debated, although most would probably agree that the first sequence is not likely to be the same motif as the 5 sequences above. It is possible to make the regular expression more discriminative by splitting it into several different ones, but it easily becomes messy. The alternative is to score sequences by how well they fit the alignment.

To score a sequence, we say that there is a probability of $4/5 = 0.8$ for an A in the first position and $1/5 = 0.2$ for a T, because we observe that out of 5 letters 4 are As and one is a T. Similarly in the second position the probability of C is $4/5$ and of G $1/5$, and so forth. After the third position in the alignment, 3 out of 5 sequences have 'insertions' of varying lengths, so we say the probability of making an insertion is $3/5$ and thus $2/5$ for not making one. To keep track of these numbers a diagram can be drawn with probabilities as in Fig. 4.1.

This is a hidden Markov model. A box in the drawing is called a state, and there is a state for each term in the regular expression. All the probabilities are found simply by counting in the multiple alignment how many times each event occur, just as described above. The only part that might seem tricky is the 'insertion,' which is represented by the state above the other states. The probability of each letter is found by counting all occurrences of the four nucleotides in this region of the alignment. The total counts are one A, two Cs, one G, and one T, yielding probabilities $1/5$, $2/5$, $1/5$, and $1/5$ respectively. After sequences 2, 3 and 5 have made one insertion each, there are two more insertions (from sequence 2) and the total number of transitions back to the main line of states is 3 (all three sequences with insertions have to finish). Therefore there are 5 transitions in total from the insert state, and the probability of making a transition to itself is $2/5$ and the probability of making one to the next state is $3/5$.

	Sequence	Probability ×100	Log odds
Consensus	`A C A C - - A T C`	4.7	6.7
Original sequences	`A C A - - - A T G`	3.3	4.9
	`T C A A C T A T C`	0.0075	3.0
	`A C A C - - A G C`	1.2	5.3
	`A G A - - - A T C`	3.3	4.9
	`A C C G - - A T C`	0.59	4.6
Exceptional	`T G C T - - A G G`	0.0023	-0.97

Table 4.1: Probabilities and log-odds scores for the 5 sequences in the alignment and for the consensus sequence and the 'exceptional' sequence.

It is now easy to score the consensus sequence `ACACATC`. The probability of the first A is $4/5$. This is multiplied by the probability of the transition from the first state to the second, which is 1. Continuing this, the total probability of the consensus is

$$\begin{aligned} P(\mathtt{ACACATC}) &= 0.8 \times 1 \times 0.8 \times 1 \times 0.8 \times 0.6 \times \\ &\quad\; 0.4 \times 0.6 \times 1 \times 1 \times 0.8 \times 1 \times 0.8 \\ &\simeq 4.7 \times 10^{-2}. \end{aligned}$$

Making the same calculation for the exceptional sequence yields only 0.0023×10^{-2}, which is roughly 2000 times smaller than for the consensus. This way we achieved the goal of getting a score for each sequence, a measure of how well a sequence fits the motif.

The same probability can be calculated for the four original sequences in the alignment in exactly the same way, and the result is shown in Table 4.1. The probability depends very strongly on the length of the sequence. Therefore the probability itself is not the most convenient number to use as a score, and the log-odds score shown in the last column of the table is usually better. It is the logarithm of the probability of the sequence divided by the probability according to a null model. The null model is one that treats the sequences as random strings of nucleotides, so the probability of a sequence of length L is 0.25^L. Then the log-odds score is

$$\text{log-odds for sequence } S = \log \frac{P(S)}{0.25^L} = \log P(S) - L \log 0.25.$$

I have used the natural logarithm in Table 4.1. Logarithms are proportional, so it does not really matter which one you use; it is quite common to use the logarithm base 2. One can of course use other null models instead. Often one would use the over-all nucleotide frequencies in the organism studied instead of just 0.25.

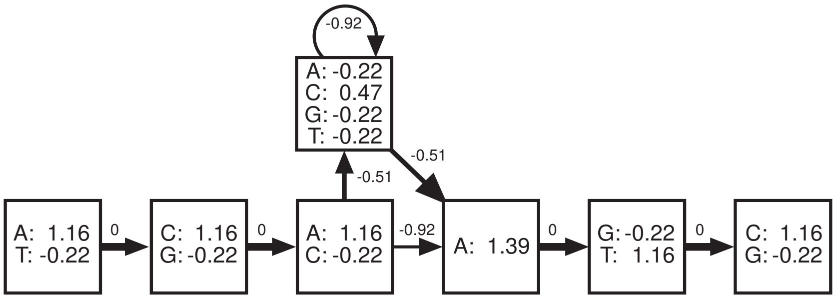

Figure 4.2: The probabilities of the model in Fig. 4.1 have been turned into log-odds by taking the logarithm of each nucleotide probability and subtracting $\log(0.25)$. The transition probabilities have been converted to simple logs.

When a sequence fits the motif very well the log-odds is high. When it fits the null model better, the log-odds score is negative. Although the raw probability of the second sequence (the one with three inserts) is almost as low as that of the exceptional sequence, notice that the log-odds score is much higher than for the exceptional sequence, and the discrimination is very good. Unfortunately, one cannot always assume that anything with a positive log-odds score is 'a hit,' because there are random hits if one is searching a large database. See Section 4.5 for references.

Instead of working with probabilities one might convert everything to log-odds. If each nucleotide probability is divided by the probability according to the null model (0.25 in this case) and the logarithm is applied, we would get the numbers shown in Fig. 4.2. The transition probabilities are also turned into logarithms. Now the log-odds score can be calculated directly by adding up these numbers instead of multiplying the probabilities. For instance, the calculation of the log-odds of the consensus sequence is

$$\begin{aligned} \text{log-odds}(\texttt{ACACATC}) &= 1.16 + 0 + 1.16 + 0 + 1.16 - 0.51 + \\ &\quad 0.47 - 0.51 + 1.39 + 01.16 + 0 + 1.16 \\ &= 6.64. \end{aligned}$$

(The finite precision causes the little difference between this number and the one in Table 4.1.)

If the alignment had no gaps or insertions we would get rid of the insert state, and then all the probabilities associated with the arrows (the transition probabilities) would be 1 and might as well be ignored completely. Then the HMM works exactly as a weight matrix of log-odds scores, which is commonly used.

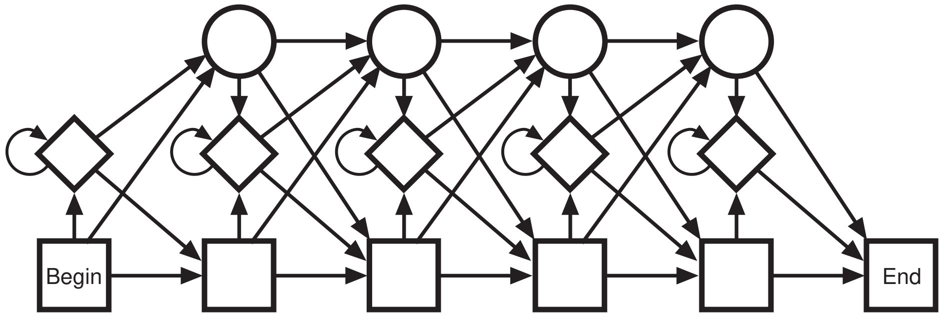

Figure 4.3: The structure of the profile HMM.

4.3 Profile HMMs

A profile HMM is a certain type of HMM with a structure that in a natural way allows position dependent gap penalties. A profile HMM can be obtained from a multiple alignment and can be used for searching a database for other members of the family in the alignment very much like standard profiles [3]. The structure of the model is shown in Fig. 4.3. The bottom line of states are called the main states, because they model the columns of the alignment. In these states the probability distribution is just the frequency of the amino acids or nucleotides as in the above model of the DNA motif. The second row of diamond shaped states are called insert states and are used to model highly variable regions in the alignment. They function exactly like the top state in Fig. 4.1, although one might choose to use a fixed distribution of residues, *e.g.* the overall distribution of amino acids, instead of calculating the distribution as in the example above. The top line of circular states are called delete states. These are a different type of state, called a silent or null state. They do not match any residues, and they are there merely to make it possible to jump over one or more columns in the alignment, *i.e.*, to model the situation when just a few of the sequences have a '–' in the multiple alignment at a position. Let us turn to an example.

Suppose you have a multiple alignment as the one shown in Fig. 4.4. A region of this alignment has been chosen to be an 'insertion,' because an alignment of this region is highly uncertain. The rest of the alignment (shaded in the figure) are the columns that will correspond to main states in the model. For each non-insert column we make a main state and set the probabilities equal to the amino acid frequencies. To estimate the transition probabilities we count how many sequences use the various transitions, just like the transition probabilities were calculated in the first example. The model is shown in Fig. 4.5. There are two transitions from a main state to a delete state shown with dashed lines in the figure, that from `begin` to the first delete state and from main state 12 to delete state

```
GGWWRGdy.ggkkqLWFPSNYV
IGWLNGynettgerGDFPGTYV
PNWWEGql..nnrrGIFPSNYV
DEWWQArr..deqiGIVPSK--
GEWWKAqs..tgqeGFIPFNFV
GDWWLArs..sgqtGYIPSNYV
GDWWDAel..kgrrGKVPSNYL
-DWWEArslssghrGYVPSNYV
GDWWYArslitnseGYIPSTYV
GEWWKArslatrkeGYIPSNYV
GDWWLArslvtgreGYVPSNFV
GEWWKAkslsskreGFIPSNYV
GEWCEAqt.kngq.GWVPSNYI
SDWWRVvnlttrqeGLIPLNFV
LPWWRArd.kngqeGYIPSNYI
RDWWEFrsktvytpGYYESGYV
EHWWKVkd.algnvGYIPSNYV
IHWWRVqd.rngheGYVPSSYL
KDWWKVev..ndrqGFVPAAYV
VGWMPGlnertrqrGDFPGTYV
PDWWEGel..ngqrGVFPASYV
ENWWNGei..gnrkGIFPATYV
EEWLEGec..kgkvGIFPKVFV
GGWWKGdy.gtriqQYFPSNYV
DGWWRGsy..ngqvGWFPSNYV
QGWWRGei..ygrvGWFPANYV
GRWWKArr.angetGIIPSNYV
GGWTQGel.ksgqkGWAPTNYL
GDWWEArsn.tgenGYIPSNYV
NDWWTGrt..ngkeGIFPANYV
```

Figure 4.4: An alignment of 30 short amino acid sequences chopped out of a alignment of the SH3 domain. The shaded areas are the most conserved and were chosen to be represented by the main states in the HMM. The unshaded area with lower-case letters was chosen to be represented by an insert state.

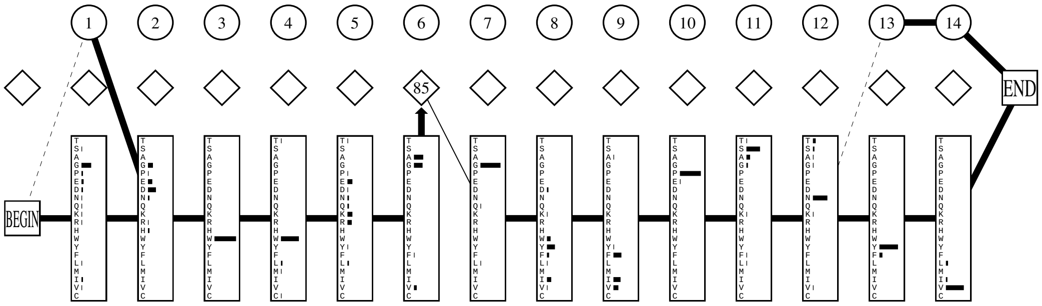

Figure 4.5: A profile HMM made from the alignment shown in Fig. 4.4. Transition lines with no arrow head are transitions from left to right. Transitions with probability zero are not shown, and those with very small probability are shown as dashed lines. Transitions from an insert state to itself is not shown; instead the probability times 100 is shown in the diamond. The numbers in the circular delete states are just position numbers. (This figure and Fig. 4.6 were generated by a program in the SAM package of programs.)

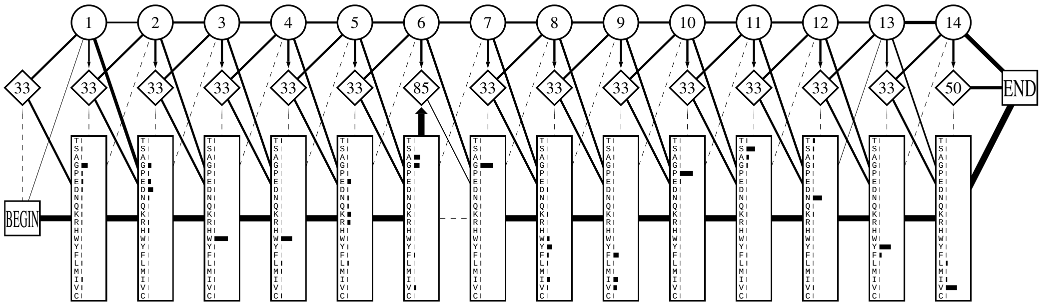

Figure 4.6: Model obtained in the same way as Fig. 4.5, but using a pseudocount of one.

13. Both of these correspond to dashes in the alignment. In both cases only one sequence has gaps, so the probability of these delete transitions is 1/30. The fourth sequence continues deletion to the end, so the probability of going from delete 13 to 14 is 1 and from delete 14 to the end is also 1.

4.3.1 Pseudocounts

It is dangerous to estimate a probability distribution from just a few observed amino acids. If for instance you have just two sequences with leucine at a certain position, the probability for leucine would be 1 and the probability would be zero for all other amino acids at this position, although it is well known that one often sees for example valine substituted for leucine. In such a case the probability of a whole sequence may easily become zero if a single leucine is substituted by a valine, or equivalently, the log-odds is minus infinity.

Therefore it is important to have some way of avoiding this sort of over-fitting, where strong conclusions are drawn from very little evidence. The most common method is to use pseudocounts, which means that one pretends to have more counts of amino acids than those from the data. The simplest is to add 1 to all the counts. With the leucine example it would mean that the probability of leucine would be estimated as $3/23$ and for the 19 other amino acids it would become $1/23$. In Fig. 4.6 a model is shown, which was obtained from the alignment in Fig. 4.6 using a pseudocount of 1.

Adding one to all the counts can be interpreted as assuming a priori that all the amino acids are equally likely. However, there are significant differences in the occurrence of the 20 amino acids in known protein sequences. Therefore, the next step is to use pseudocounts proportional to the observed frequencies of the amino acids instead. This is the minimum level of pseudocounts to be used in any real application of HMMs.

Because a column in the alignment may contain information about the preferred type of amino acids, it is also possible to use more sophisticated pseudocount strategies. If a column consists predominantly of leucine (as above), one would expect substitutions to other hydrophobic amino acids to be more probable than substitutions to hydrophilic amino acids. One can *e.g.* derive pseudocounts for a given column from substitution matrices. See Section 4.5 for references.

4.3.2 Searching a database

Above we saw how to calculate the probability of a sequence in the alignment by multiplying all the probabilities (or adding the log-odds scores) in the model along the *path* followed by that particular sequence. However, this path is usually not known for other sequences which are not part of the original alignment, and the next problem is how to score such a sequence. Obviously, if we can find a path through the model where the new sequence fits well in some sense, then we can score the sequence as before. We need to 'align' the sequence to the model. It resembles very much the pairwise alignment problem, where two sequences are aligned so that they are most similar, and indeed the same type of dynamic programming algorithm can be used.

For a particular sequence, an alignment to the model (or a path) is an assignment of states to each residue in the sequence. There are many such alignments for a given sequence. For instance an alignment might be as follows. Let us label the amino acids in a protein as A_1, A_2, A_3, *etc.* Similarly we can label the HMM states as M_1, M_2, M_3, *etc.* for match states, I_1, I_2, I_3 for insert states, and so on. Then an alignment could have A_1 match state M_1, A_2 and A_3 match I_1, A_4 match M_2, A_5 match M_6 (after passing through three delete states), and so on. For each such path we can calculate the probability of the sequence or the log-odds score, and thus we can find the *best* alignment, *i.e.*, the one with the largest probability. Although there are an enormous number of possible alignments it can be done efficiently by the above mentioned dynamic programming algorithm, which is called the Viterbi algorithm. The algorithm also gives the probability of the sequence for that alignment, and thus a score is obtained.

The log-odds score found in this manner can be used to search databases for members of the same family. A typical distribution of scores from such a search is shown in Fig. 4.7. As is also the case with other types of searches, there is no clear-cut separation of true and false positives, and one needs to investigate some of the sequences around a log-odds of zero, and possibly include some of them in the alignment and try searching again.

An alternative way of scoring sequences is to *sum* the probabilities of all possible alignments of the sequence to the model. This probability can be found by a similar algorithm called the forward algorithm. This type of scoring is not very common in biological sequence comparison, but it is more natural from a

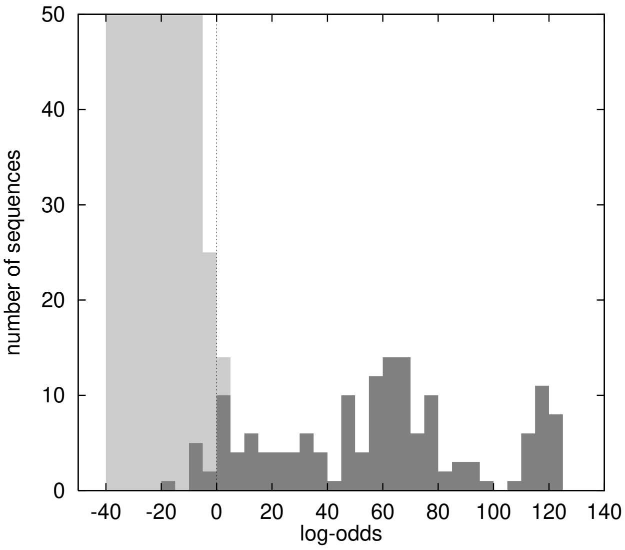

Figure 4.7: The distribution of log-odds scores from a search of Swissprot with a profile HMM of the SH3 domain. The dark area of the histogram represents the sequences with an annotated SH3 domain, and the light those that are not annotated as having one. This is for illustrative purposes only, and the sequences with log-odds around zero were not investigated further.

probabilistic point of view. However, it usually gives very similar results.

4.3.3 Model estimation

As presented so far, one may view the profile HMMs as a generalization of weight matrices to incorporate insertions and deletions in a natural way. There is however one interesting feature of HMMs, which has not been addressed yet. It is possible to estimate the model, *i.e.* determine all the probability parameters of it, from *unaligned* sequences. Furthermore, a multiple alignment of the sequences is produced in the process. Like many other multiple alignment methods this is done in an iterative manner. One starts out with a model with more or less random probabilities, or if a reasonable alignment of some of the sequences are available, a model is constructed from this alignment. Then, when all the sequences are aligned to the model, we can use the alignment to improve the probabilities in the model. These new probabilities may then lead to a slightly different alignment. If they do, we then repeat the process and improve the probabilities again. The process is repeated until the alignment does not change. The alignment of the sequences to the final model yields a multiple alignment.[1]

Although this estimation process sounds easy, there are many problems to consider to actually make it work well. One problem is choosing the appropriate model length, which determines the number of inserts in the final alignment. Another severe problem is that the iterative procedure can converge to suboptimal solutions. It is not guaranteed that it finds the optimal multiple alignment, *i.e.* the most probable one. Methods for dealing with these issues are described in the literature pointed to in Section 4.5.

4.4 HMMs for gene finding

One ability of HMMs, which is not really utilized in profile HMMs, is the ability to model *grammar*. Many problems in biological sequence analysis have a grammatical structure, and eukaryotic gene structure, which I will use as an example, is one of them. If you consider exons and introns as the 'words' in a language, the sentences are of the form exon-intron-exon-intron...intron-exon. The 'sentences' can never end with an intron, at least if the genes are complete, and an exon can never follow an exon without an intron in between. Obviously this grammar is greatly simplified, because there are several other constraints on gene structure, such as the constraint that the exons have to fit together to give a valid coding region after splicing. In Fig. 4.8 the structure of a gene is shown with some of the known signals marked.

Formal language theory applied to biological problems is not a new invention. In particular David Searls [7] has promoted this idea and used it for gene finding [8], but many other gene finders use it implicitly. Formally the HMM can only represent the simplest of grammars, which is called a regular grammar [7, 1], but that turns out to be good enough for the gene finding problem, and many other problems. One of the problems that has a more complicated grammar than the HMM can handle is the RNA folding problem, which is one step up the ladder of grammars, because base pairing introduces correlations between bases far from each other in the RNA sequence.

I will here briefly outline my own approach to gene finding with the weight on the principles rather than on the details.

[1] Another slightly different method for model estimation sums over all alignments instead of using the most probable alignment of a sequence to the model. This method uses the forward algorithm instead of Viterbi, and it is called the Baum–Welch algorithm or the forward–backward algorithm.

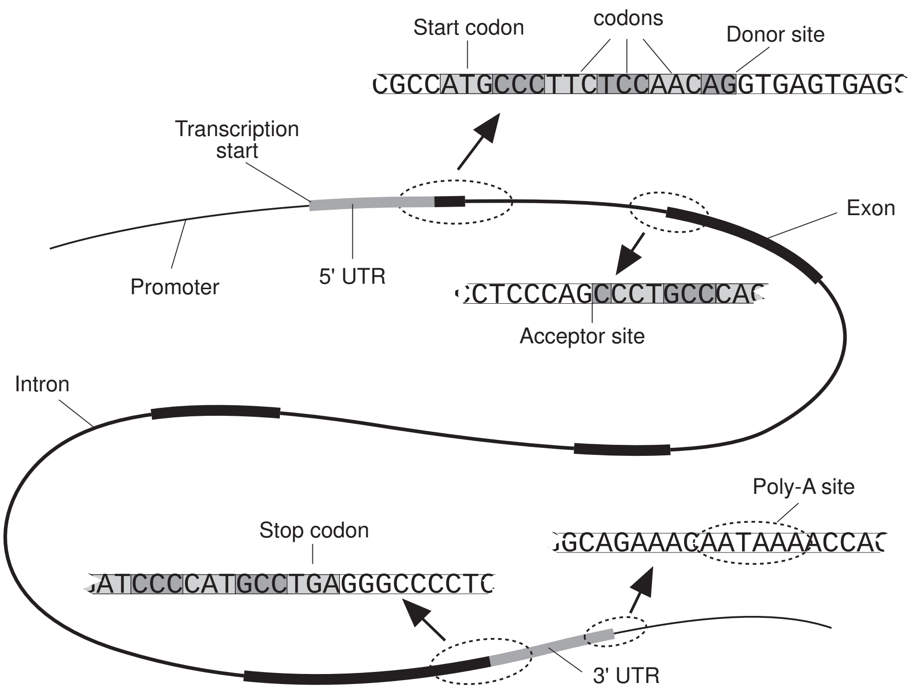

Figure 4.8: The structure of a gene with some of the important signals shown.

4.4.1 Signal sensors

One may apply an HMM similar to the ones already described directly to many of the signals in a gene structure. In Fig. 4.9 an alignment is shown of some sequences around acceptor sites from human DNA. It has 19 columns and an HMM with 19 states (no insert or delete states) can be made directly from it. Since the alignment is gap-less, the HMM is equivalent to a weight matrix.

There is one problem: in DNA there are fairly strong dinuclotide preferences. A model like the one described treats the nucleotides as independent, so dinucleotide preferences can not be captured. This is easily fixed by having 16 probability parameters in each state instead of 4. In column two we first count all occurrences of the four nucleotides given that there is an A in the first column and normalize these four counts, so they become probabilities. This is the *conditional probability* that a certain nucleotide appears in position two, given that the previous one was A. The same is done for all the instances of C in column 1 and similarly for G and T. This gives a total of 16 probabilities to be used in state two of the HMM. Similarly for all the other states. To calculate the probability of a sequence, say ACTGTC..., we just multiply the conditional probabilities

$$P(ACTGTC\ldots) = p_1(A) \times p_2(C|A) \times p_3(T|C) \times p_4(G|T) \times p_5(T|G) \times p_6(C|T) \times \ldots$$

```
C T C C C T G T G T C C A C A G G C T
T A T T G T T T T C T T A C A G G G C
G T T C C T T T G T T T C T A G C A C
T G C C T C T C T T T T C A A G G G T
T C C T A T A T G T T G A C A G G G T
T T C T G T T C C G A T G C A G G G C
T T G G G T T T C T T T G C A G A A C
C A C T T T G C T C C C A C A G C G T
C C C A T G T G A C C T G C A G G T A
T A T T T A T T T A A C A T A G G G C
A T G T G C A T C C C C C C A G G A G
T T T T C C T T T T C T A C A G A A T
T C G T G T G T C T C C C C A G C C C
T T C C A T G T C C T G A C A G G T G
A C G A C A T T T T C C A C A G G A G
G T G C C T C T C C C T C C A G A T T
```

Figure 4.9: Examples of human acceptor sites (the splice site 5' to the exon). Except in rare cases, the intron ends with AG, which has been highlighted. Included in these sequences are 16 bases upstream of the splice site and 3 bases downstream into the exon.

Here p_1 is the probability of the four nucleotides in state 1, $p_2(x|y)$ is the conditional probability in state 2 of nucleotide x given that the previous nucleotide was y, and so forth.

A state with conditional probabilities is called a first order state, because it captures the first order correlations between neighboring nucleotides. It is easy to expand to higher order. A second order state has probabilities conditioned on the two previous nucleotides in the sequence, *i.e.*, probabilities of the form $p(x|y,z)$. We will return to such higher order states below.

Small HMMs like this are constructed in exactly the same way for other signals: donor splice sites, the regions around the start codons, and the regions around the stop codons.

4.4.2 Coding regions

The codon structure is the most important feature of coding regions. Bases in triplets can be modeled with three states as shown in Fig. 4.10. The figure also shows how this model of coding regions can be used in a simple model of an unspliced gene that starts with a start codon (ATG), then consists of some number of codons, and ends with a stop codon.

Since a codon is three bases long, the last state of the codon model must be at least of order two to correctly capture the codon statistics. The 64 probabilities in such a state are estimated by counting the number of each codon in a set of known coding regions. These numbers are then normalized properly. For example the probabilities derived from the counts of CAA, CAC, CAG, and CAT are

$$p(\mathrm{A}|\mathrm{CA}) = c(\mathrm{CAA})/[c(\mathrm{CAA})+c(\mathrm{CAC})+c(\mathrm{CAG})+c(\mathrm{CAT})]$$

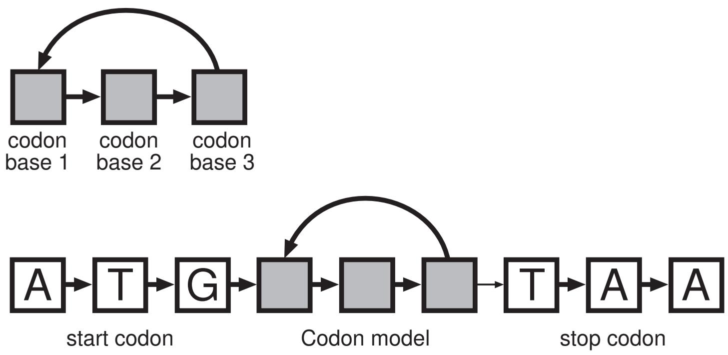

Figure 4.10: Top: A model of coding regions, where state one, two and three match the first, second and third codon positions respectively. A coding region of any length can match this model, because of the transition from state three back to state one. Bottom: a simple model for unspliced genes with the first three states matching a start codon, the next three of the form shown to the left, and the last three states matching a stop codon (only one of the three possible stop codons are shown).

$$
\begin{aligned}
p(\mathrm{C}|\mathrm{CA}) &= c(\mathrm{CAC})/[c(\mathrm{CAA})+c(\mathrm{CAC})+c(\mathrm{CAG})+c(\mathrm{CAT})] \\
p(\mathrm{G}|\mathrm{CA}) &= c(\mathrm{CAG})/[c(\mathrm{CAA})+c(\mathrm{CAC})+c(\mathrm{CAG})+c(\mathrm{CAT})] \\
p(\mathrm{T}|\mathrm{CA}) &= c(\mathrm{CAT})/[c(\mathrm{CAA})+c(\mathrm{CAC})+c(\mathrm{CAG})+c(\mathrm{CAT})]
\end{aligned}
$$

where $c(xyz)$ is the count of codon *xyz*.

One of the characteristics of coding regions is the lack of stop codons. That is automatically taken care of, because $p(A|TA)$, $p(G|TA)$ and $p(A|TG)$, corresponding to the three stop codons TAA, TAG and TGA, will automatically become zero.

For modeling codon statistics it is natural to use an ordinary (zeroth order) state as the first state of the codon model and a first order state for the second. However, there are actually also dependencies between neighboring codons, and therefore one may want even higher order states. In my own gene finder, I use three fourth order states, which is inspired by GeneMark [9], in which such models were first introduced. Technically speaking, such a model is called an inhomogeneous Markov chain, which can be viewed as a sub-class of HMMs.

4.4.3 Combining the models

To be able to discover genes, we need to combine the models in a way that satisfies the grammar of genes. I restrict myself to coding regions, *i.e.* the 5' and 3' untranslated regions of the genes are not modeled and also promoters are disregarded.

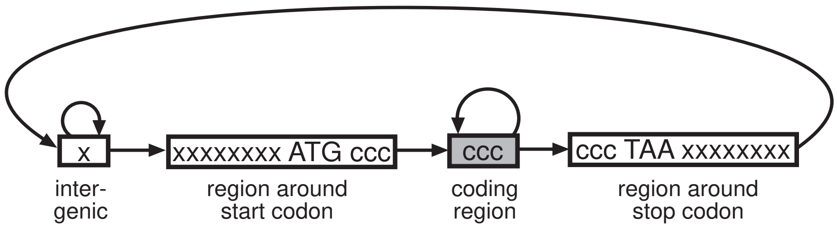

Figure 4.11: A hidden Markov model for unspliced genes. In this drawing an 'x' means a state for non-coding DNA, and a 'c' a state for coding DNA. Only one of the three possible stop codons are shown in the model of the region around the stop codon.

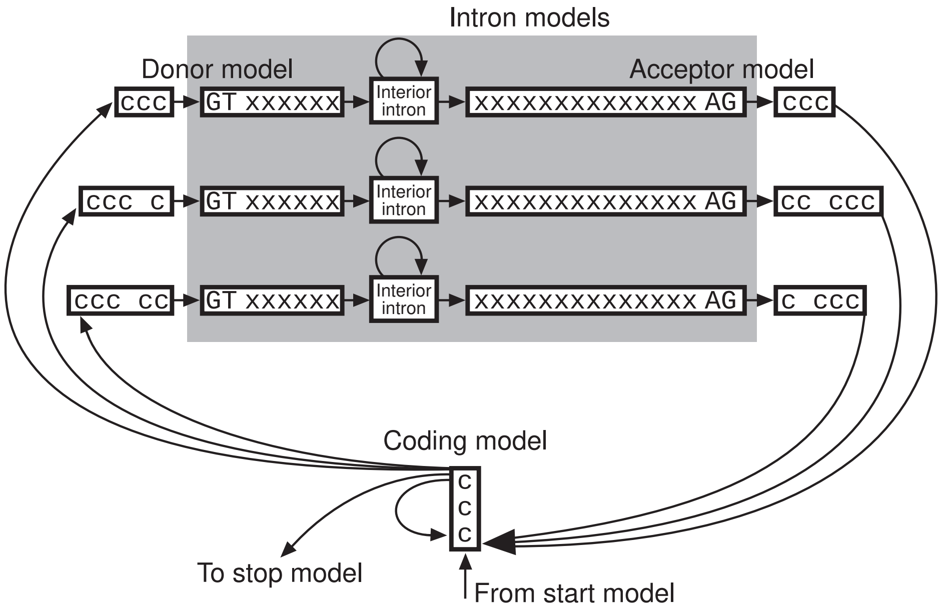

Figure 4.12: To allow for splicing in three different frames three intron models are needed. To get the frame correct 'spacer states' are added before and after the intron models.

First, let us see how to do it for unspliced genes. If we ignore genes that are very closely spaced or overlaps, a model could look like Fig. 4.11. It consists of a state for intergenic regions (of order at least 1), a model for the region around the start codon, the model for the coding region, and a model for the region around the stop codon. The model for the start codon region is made just like the acceptor model described earlier. It models eight bases upstream of the start codon,[2] the ATG start codon itself, and the first codon after the start. Similarly for the stop codon region. The whole model is one big HMM, although it was put together from small independent HMMs.

Having such a model, how can we predict genes in a sequence of anonymous DNA? That is quite simple: use the Viterbi algorithm to find the most probable path through the model. When this path goes through the ATG states, a start codon is predicted, when it goes through the codon states a codon is predicted, and so on.

This model might not always predict correct genes, but at least it will only predict sensible genes that obey the grammatical rules. A gene will always start with a start codon and end with a stop codon, the length will always be divisible by 3, and it will never contain stop codons in the reading frame, which are the minimum requirements for unspliced gene candidates.

Making a model that conforms to the rules of splicing is a bit more difficult than it might seem at first. That is because splicing can happen in three different reading frames, and the reading frame in one exon has to fit the one in the next. It turns out that by using three different models of introns, one for each frame, this is possible. In Fig. 4.12 it is shown how these models are added to the model of coding regions.

The top line in the model is for introns appearing between two codons. It has three states (labeled ccc) before the intron starts to match the last codon of the exon. The first two states of the intron model match GT, which is the consensus sequence at donor sites (it is occasionally another sequence, but such cases are ignored here). The next six states matches the six bases immediately after GT. The states just described model the donor site, and the probabilities are found as it was described earlier for acceptor sites. Then follows a single state to model the interior of the intron. I actually use the same probability parameters in this state as in the state modeling intergenic regions. Now follows the acceptor model, which includes three states to match the first codon of the next exon.

The next line in the model is for introns appearing after the first base in a codon. The difference from the first is that there is one more state for a coding

[2] A similar model could be used for prokaryotic genes. In that case, however, one should model the Shine-Dalgarno sequence, which is often more than 8 bases upstream from the start. Also, one would probably need to allow for other start codons than ATG that are used in the organism studied (in some eukaryotes other start codons can also be used).

base before the intron and two more states after the intron. This ensures that the frames fit in two neighboring exons. Similarly in the third line from the top there are two extra coding states before the intron and one after, so that it can match introns appearing after the second base in a codon.

There are obviously many possible variations of the model. One can add more states to the signal sensors, include models of promoter elements and untranslated regions of the gene, and so forth.

4.5 Further reading

A general introduction can be found in [2], and one aimed more at biological readers in [1]. The first applications for sequence analysis is probably for modeling compositional differences between various DNA types [10] and for studying the compositional structure of genomes [11]. The initial work on using hidden Markov models (HMMs) as ‘probabilistic profiles’ of protein families was presented at a conference in the spring of 1992 and in a technical report from the same year, and it was published in [12, 13]. The idea was quickly taken up by others [14, 15]. Independently, some very similar ideas were also developed in [16, 17]. Also the generalized profiles [18] are very similar.

Estimation and multiple alignment is described in [13] in detail, and in [19] some of the practical methods are further discussed. Alternative methods for model estimation are presented in [14, 20]. Methods for scoring sequences against a profile HMM were given in [13], but these issues have more recently been addressed in [21]. The basic pseudocount method is also explained in [13], and more advanced methods are discussed in [22, 23, 24, 25, 26].

A review of profile HMMs can be found in [27], and in [1] profile HMMs are discussed in great detail. Also [28] will undoubtedly contain good material on profile HMMs.

Some of the recent applications of profile HMMs to proteins are: detection of fibronectin type III domains in yeast [29], a database of protein domain families [30], protein topology recognition from secondary structure [31], and modeling of a protein splicing domain [32].

There are two program packages available free of charge to the academic community. One, developed by Sean Eddy, is called hmmer (pronounced ‘hammer’), and can be obtained from his web-site (http://genome.wustl.edu/eddy/hmm.html). The other one, called SAM (http://www.cse.ucsc.edu/research/compbio/sam.html), was developed by myself and the group at UC Santa Cruz, and it is now being maintained and further developed under the command of Richard Hughey.

The gene finder sketched above is called HMMgene. There are many details omitted, such as special methods for estimation and prediction described in [33].

It is still under development, and it is possible to follow the development and test the current version at the web site http://www.cbs.dtu.dk/services/HMMgene/.

Methods for automated gene finding go back a long time, see [34] for a review. The first HMM based gene finder is probably EcoParse developed for E. coli [35]. VEIL [36] is a recent HMM based gene finder for for human genes. The main difference from HMMgene is that it does not use high order states (neither does EcoParse), which makes good modeling of coding regions harder.

Two recent methods use so-called generalized HMMs. Genie [37, 38, 39] combines neural networks into an HMM-like model, whereas GENSCAN [40] is more similar to HMMgene, but uses a different model type for splice site. Also, the generalized HMM can explicitly use exon length distributions, which is not possible in a standard HMM. Web pointers to gene finding can be found at http://www.cbs.dtu.dk/krogh/genefinding.html.

Other applications of HMMs related to gene finding are: detection of short protein coding regions and analysis of translation initiation sites in Cyanobacterium [41, 42], characterization of prokaryotic and eukaryotic promoters [43], and recognition of branch points [44].

Apart from the areas mentioned here, HMMs have been used for prediction of protein secondary structure [45], modeling an oscillatory pattern in nucleosomes [46], modeling site dependence of evolutionary rates [47], and for including evolutionary information in protein secondary structure prediction [48].

Acknowledgements

Henrik Nielsen, Steven Salzberg, and Steen Knudsen are gratefully acknowledged for comments on the manuscript. This work was supported by the Danish National Research Foundation.

References

[1] Durbin, R. M., Eddy, S. R., Krogh, A., and Mitchison, G. (1998) *Biological Sequence Analysis* Cambridge University Press To appear.

[2] Rabiner, L. R. (1989) *Proc. IEEE* 77, 257–286.

[3] Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987) *Proc. of the Nat. Acad. of Sciences of the U.S.A.* 84, 4355–4358.

[4] Staden, R. (1984) *Nucleic Acids Research* 12, 505–519.

[5] Staden, R. (1988) *Computer Applications in the Biosciences* 4, 53–60.

[6] Bairoch, A., Bucher, P., and Hofmann, K. (1997) *Nucleic Acids Research* 25, 217–221.

[7] Searls, D. B. (1992) *American Scientist* 80, 579–591.

[8] Dong, S. and Searls, D. B. (1994) *Genomics* 23, 540–551.

[9] Borodovsky, M. and McIninch, J. (1993) *Computers and Chemistry* 17, 123–133.

[10] Churchill, G. A. (1989) *Bull Math Biol* 51, 79–94.

[11] Churchill, G. A. (1992) *Computers and Chemistry* 16, 107–115.

[12] Haussler, D., Krogh, A., Mian, I. S., and Sjölander, K. (1993) Protein modeling using hidden Markov models: Analysis of globins In Mudge, T. N., Milutinovic, V., and Hunter, L. (Eds.), *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences* volume 1 pp. 792–802 Los Alamitos, California. IEEE Computer Society Press.

[13] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994) *Journal of Molecular Biology* 235, 1501–1531.

[14] Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M. A. (1994) *Proc. of the Nat. Acad. of Sciences of the U.S.A.* 91, 1059–1063.

[15] Eddy, S. R., Mitchison, G., and Durbin, R. (1995) *J Comput Biol* 2, 9–23.

[16] Stultz, C. M., White, J. V., and Smith, T. F. (1993) *Protein Science* 2, 305–314.

[17] White, J. V., Stultz, C. M., and Smith, T. F. (1994) *Mathematical Biosciences* 119, 35–75.

[18] Bucher, P., Karplus, K., Moeri, N., and Hofmann, K. (1996) *Computers and Chemistry* 20, 3–24.

[19] Hughey, R. and Krogh, A. (1996) *CABIOS* 12, 95–107.

[20] Eddy, S. R. (1995) Multiple alignment using hidden Markov models. 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 pp. 114–120 Menlo Park, CA. AAAI Press.

[21] Barrett, C., Hughey, R., and Karplus, K. (1997) *Computer Applications in the Biosciences* 13, 191–199.

[22] Brown, M., Hughey, R., Krogh, A., Mian, I. S., Sjölander, K., and Haussler, D. (1993) Using Dirichlet mixture priors to derive hidden Markov models for protein families In Hunter, L., Searls, D., and Shavlik, J. (Eds.), *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology* pp. 47–55 Menlo Park, CA. AAAI/MIT Press.

[23] Tatusov, R. L., Altschul, S. F., and Koonin, E. V. (1994) *Proc. of the Nat. Acad. of Sciences of the U.S.A.* 91, 12091–12095.

[24] Karplus, K. (1995) Evaluating regularizers for estimating distributions of amino acids. 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 pp. 188–196 Menlo Park, CA. AAAI Press.

[25] Henikoff, J. G. and Henikoff, S. (1996) *Computer Applications in the Biosciences* 12, 135–143.

[26] Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. (1996) *CABIOS* 12, 327–345.

[27] Eddy, S. R. (1996) *Current Opinion in Structural Biology* 6, 361–365.

[28] Baldi, P. and Brunak, S. (1998) *Bioinformatics - The Machine Learning Approach* MIT Press Cambridge MA To appear.

[29] Bateman, A. and Chothia, C. (1997) *Curr. Biol.* 6, 1544–1547.

[30] Sonnhammer, E. L. L., Eddy, S. R., and Durbin, R. (1997) *Proteins* 28, 405–420.

[31] Di Francesco, V., Garnier, J., and Munson, P. J. (1997) *Journal of Molecular Biology* 267, 446–463.

[32] Dalgaard, J. Z., Moser, M. J., Hughey, R., and Mian, I. S. (1997) *Journal of Computational Biology* 4, 193–214.

[33] Krogh, A. (1997) Two methods for improving performance of a HMM and their application for gene finding In Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C., and Valencia, A. (Eds.), *Proc. of Fifth Int. Conf. on Intelligent Systems for Molecular Biology* pp. 179–186 Menlo Park, CA. AAAI Press.

[34] Fickett, J. W. (1996) *Trends Genet.* 12, 316–320.

[35] Krogh, A., Mian, I. S., and Haussler, D. (1994) *Nucleic Acids Research* 22, 4768–4778.

[36] Henderson, J., Salzberg, S., and Fasman, K. H. (1997) Finding genes in DNA with a hidden Markov model Journal of Computational Biology.

[37] 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* pp. 134–142 Menlo Park, CA. AAAI Press.

[38] 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.

[39] 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.

[40] Burge, C. and Karlin, S. (1997) *Journal of Molecular Biology* 268, 78–94.

[41] Yada, T. and Hirosawa, M. (1996) *DNA Res.* 3, 355–361.

[42] Yada, T., Sazuka, T., and Hirosawa, M. (1997) *DNA Res.* 4, 1–7.

[43] Pedersen, A. G., Baldi, P., Brunak, S., and Chauvin, Y. (1996) Characterization of prokaryotic and eukaryotic promoters using hidden Markov models In *Proc. of Fourth Int. Conf. on Intelligent Systems for Molecular Biology* pp. 182–191 Menlo Park, CA. AAAI Press.

[44] Tolstrup, N., Rouzé, P., and Brunak, S. (1997) *Nucleic Acids Research* 25, 3159–3164.

[45] Asai, K., Hayamizu, S., and Handa, K. (1993) *Computer Applications in the Biosciences* 9, 141–146.

[46] Baldi, P., Brunak, S., Chauvin, Y., and Krogh, A. (1996) *Journal of Molecular Biology* 263, 503–510.

[47] Felsenstein, J. and Churchill, G. A. (1996) *Molecular Biological Evolution* 13.

[48] Goldman, N., Thorne, J. L., and Jones, D. T. (1996) *Journal of Molecular Biology* 263, 196–208.