1 Advances in Hidden Markov Models for Sequence Annotation

BROŇA BREJOVÁ[1], DANIEL G. BROWN[2], and TOMÁŠ VINAŘ[1]

[1] Dept. of Biological Statistics and Computational Biology
Cornell University, Ithaca, NY, USA
[2] Cheriton School of Computer Science, University of Waterloo
Waterloo, Ontario, Canada

1.1 INTRODUCTION

One of the most basic tasks of bioinformatics is to identify features in a biological sequence. Whether those features are the binding sites of a protein, the regions of a DNA sequence that are most subject to selective pressures, or coding sequences found in an expressed sequence tag, this phase is fundamental to the process of sequence analysis.

While a variety of computational tools have been used over the course of the time that people have been needing to perform this task, the currently dominant tool in biological sequence annotation is the hidden Markov model (HMM). HMMs have been used in so many contexts over the course of the last fifteen years that they almost require no introduction. They are used in computational gene finders, to predict the structure of genes in newly sequenced genomes. They are used in protein sequence analysis, to identify substructural elements. They are used to discover regulatory sequences in DNA, to identify ancestry patterns in pedigrees, and truly for almost any feature detection problem in biological sequences.

As such, it may seem that their use is so routinized that there is nothing more to learn about them: that fifteen years of their use in biological sequence analysis mined the field for all of its interesting problems many years ago. Fortunately, this is anything but the case. As the fields of genomics and proteomics advance, a variety of new challenges have come to fore in the algorithmic analysis of HMMs. For example, if we have a large amount of training data, and can train an HMM to closely model the many complex features of the data, will that necessarily improve the quality of our predictions on new data? How can we properly model the distributions of the lengths of complex sequence features in HMMs? How can we incorporate

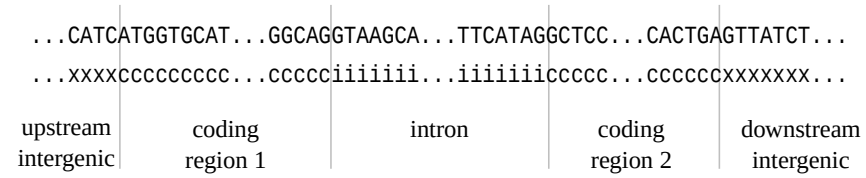


Fig. 1.1 In gene finding, the goal is to label each nucleotide of a given DNA sequence as coding (c), intron (i) or intergenic (x).

evolutionary conservation information into the creation of HMMs that properly model DNA sequences, and into the algorithms for their analysis?

This chapter considers the use of HMMs in sequence analysis, starting from the simplest cases (simple HMMs, with simple structures, simple training algorithms and simple decoding procedures), and moving to situations of great complexity, incorporating very recent ideas from machine learning theory. We present the basic algorithms, and their extensions, and give suggestions of places where future research can be most useful. Throughout, we make reference to the important applications in which these algorithms are used, and to why the field has experienced continuous advancement over the past many years.

1.2 HIDDEN MARKOV MODELS FOR SEQUENCE ANNOTATION

In this section, we illustrate the use of HMMs for biological sequence annotation. We will focus on a simplification of one of the most prominent uses of HMMs in sequence annotation: the problem of gene finding. Assume we are given a section of a DNA sequence containing a single protein-coding gene, and our goal is to locate the regions of this sequence that code for a protein. In eukaryotes, such regions may be interrupted by non-coding segments, called *introns*. Therefore, our task is to label each nucleotide of the DNA sequence with one of three labels, indicating whether the nucleotide comes from a coding region, an intron, or an intergenic region (see Figure 1.1).

More generally, the problem of labeling every symbol of a biological sequence with its functional category is the *sequence annotation problem*; such a sequence of labels is an *annotation* of a sequence.

For our gene finding problem, we use our knowledge of gene structure and a collection of training data to design an HMM that characterizes typical DNA sequences and their gene annotations. Then we will use this model to find the highest probability annotations for novel, unannotated DNA sequences.

1.2.1 Hidden Markov models

A hidden Markov model is a generative probabilistic model for modeling sequence data that comes from a finite alphabet. An HMM consists of a finite set of states

and three sets of parameters, called the initial, emission, and transition probabilities. The initial probability s_k is defined for each state k of the model. The transition probability $a_{k,\ell}$ is defined for each pair of states (k, ℓ), and the emission probability $e_{k,b}$ is defined for each state k and each symbol b of the output alphabet. The initial probabilities form a probability distribution, as do the transition probabilities $a_{k,\ell}$ at each state k and the emission probabilities $e_{k,b}$ for each k.

An HMM generates a sequence step-by-step, one symbol in each step. First, a start state is randomly generated according to the initial probabilities. Then, in each step, the model randomly generates one symbol and then moves to a new state. Both the new symbol and the next state depend only on the current state. If the current state is k, the symbol b will be generated with probability $e_{k,b}$, and the next state will be ℓ with probability $a_{k,\ell}$.

In n steps, the HMM generates a sequence $X = x_1, \ldots, x_n$ and traverses a sequence of states (or *state path*) $H = h_1, \ldots, h_n$. For a fixed length n, the HMM defines a probability distribution over all possible sequences X and all possible state paths H; in particular, the probability that the model will traverse the state path H and generate the sequence X is the following product of the model parameters:

$$\Pr(H, X) = s_{h_1} \left(\prod_{i=1}^{n-1} e_{h_i,x_i} a_{h_i,h_{i+1}} \right) e_{h_n,x_n}. \tag{1.1}$$

1.2.2 Choosing the topology and parameters of an HMM

To approach our gene finding problem, we will first build an HMM that models DNA sequences and their corresponding genes. Our model will have four states: one state representing the intergenic region upstream of the gene, one representing coding regions of the gene, one representing introns, and one representing the region downstream of the gene. Each state will emit symbols over the alphabet {A,C,G,T}. In this way, the sequence generated by the HMM will represent a DNA sequence, with the corresponding state path identifying its correct annotation.

Transitions between some pairs of states should never occur. There will be no transitions between introns and intergenic regions, nor between the two states representing upstream and downstream intergenic regions. To visualize the structure of the HMM (also called its *topology*), we use a directed graph, where vertices correspond to the states, and edges to non-zero probability transitions (see Figure 1.2).

Next, we determine the emission and transition probabilities for each state, using a *training set* T containing sequences with known annotation. Because we have designated each state in our model to represent a region of a particular function, we can use these annotations to determine the proper state path H_i for each of the sequences X_i in the training set T. We would like our generative model to generate sequences whose distributions and annotations are similar to those observed in the training set T. Formally, using the *maximum likelihood estimation* principle, we want to set the emission and transition probabilities to maximize the likelihood of the training

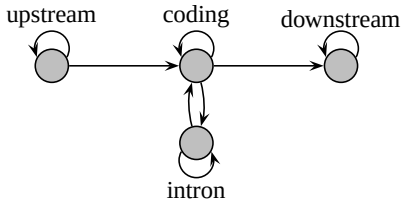


Fig. 1.2 Topology of a simplified HMM for gene finding.

data: that is, to maximize $\prod_i \Pr(H_i, X_i)$, over all possible parameters for the model. To maximize this probability, it is sufficient to count the frequency of using each transition in the training set to estimate the transition probabilities, and the frequency of emission of each symbol in each state to estimate the emission probabilities. In other contexts, this training process can be quite a bit more complicated; for example, when the training set T is unannotated, or when a given sequence and annotation could correspond to multiple paths in the HMM; we discuss this scenario in Section 1.6.

However, in our simple case, we have created a probabilistic model of sequences and their annotations. In the next section, we show how to use this probabilistic model to annotate a novel DNA sequence.

1.2.3 HMM decoding: the Viterbi algorithm

Once the HMM topology is set, and its parameters trained, we can use it to find genes in a new unlabeled DNA sequence X. That is, we seek an appropriate state path H^* that best explains how the model could have produced X; this process is called *HMM decoding*.

The simplest measure of "best" is to find the path that has the maximum probability in the HMM, given the sequence X. Recall that the model gives the joint probabilities $\Pr(H, X)$ for all sequence/annotation pairs; as such, it also gives the posterior probability $\Pr(H|X) = \Pr(H, X)/\Pr(X)$, for every possible state path H through the model, conditioned on the sequence X. We will seek the path with maximum posterior probability. Given that the denominator $\Pr(X)$ is constant in the conditional probability formula for a given sequence X, maximizing the posterior probability is equivalent to finding the state path H^* that maximizes the joint probability $\Pr(H^*, X)$.

The most probable state path can be found in time linear in the sequence length by the Viterbi algorithm (Viterbi, 1967; Forney, 1973). This simple dynamic programming algorithm computes the optimal paths for all prefixes of X; when we move from the i-length prefix to the $(i + 1)$-length prefix, we need only add one edge to one of the pre-computed optimal paths for the i-length prefix.

For every position i in the sequence and every state k, the algorithm finds the most probable state path $h_1 \ldots h_i$ to generate the first i symbols of X, provided that $h_i = k$. The value $V[i, k]$ stores the joint probability $\Pr(h_1 \ldots h_i, x_1 \ldots x_i)$ of this

optimal state path. Again, if $h_1 \dots h_i$ is the most probable state path generating $x_1 \dots x_i$ that ends in state h_i, then $h_1 \dots h_{i-1}$ must be the most probable state path generating $x_1 \dots x_{i-1}$ and ending in state h_{i-1}. To compute $V[i,k]$, we consider all possible states as candidates for the second-to-last state, h_{i-1}, and select the one that leads to the most probable state path, as expressed in the following recurrence:

$$V[i,k] = \begin{cases} s_k \cdot e_{k,x_1} & \text{if } i = 1, \\ \max_\ell V[i-1,\ell] \cdot a_{\ell,k} \cdot e_{k,x_i} & \text{otherwise.} \end{cases} \tag{1.2}$$

The probability $\Pr(H^*, X)$ is then the maximum over all states k of $V[n,k]$, and the most probable state path H^* can be traced back through the dynamic programming table by standard techniques. The running time of the algorithm is $O(nm^2)$, where n is the length of the sequence and m is the number of states in the HMM.

1.2.4 More complex HMMs

We have demonstrated the basic techniques needed to use HMMs for sequence annotation. However, the models actually used in practice are more complex than the one shown in Figure 1.2. We rarely have only one state for each feature in the HMM, and it is quite possible that we need to incorporate more positional dependencies into the probabilities of the HMM. We will explain this in the context of our gene-finding example.

First, note that coding regions are composed of codons that each encode one amino acid. Therefore it is advisable to model coding regions by a three-state cycle rather than a single state, to properly keep this structure. Codons can be interrupted by an intron, so we use multiple copies of the intron submodel, a technique that originated in finite state machines, to enforce that the next coding region after the intron starts at the proper codon position. Boundaries of coding regions are marked by special sequence signals which require additional states in the model. Finally, DNA sequence usually contains multiple genes on both strands. Figure 1.3 shows an HMM topology that encodes all of these additional constraints.

And, as noted, we may want to incorporate positional dependencies into the HMM. This is most often done by allowing higher-order states. In a state of order o, the probability of generating the character b is a function of the o previously generated characters (all states in a standard HMM are of order zero). The emission table has the form $e_{k,b_1,\dots,b_o,b}$, where $\sum_b e_{k,b_1,\dots,b_o,b} = 1$, for a fixed state k and characters $b_1, \dots, b_o$. In an HMM with all states of order o, Formula (1.1) generalizes as follows (we ignore the special case of the first o characters):

$$\Pr(H,X) = s_{h_1} \left(\prod_{i=1}^{n-1} e_{h_i,x_{i-o},\dots,x_i} a_{h_i h_{i+1}} \right) e_{h_n,x_{n-o},\dots,x_n}. \tag{1.3}$$

The Viterbi algorithm for finding the most probable state path can be adapted easily to handle higher order states with the same running time. Similarly, training the parameters of higher-order HMMs by maximum likelihood is straightforward

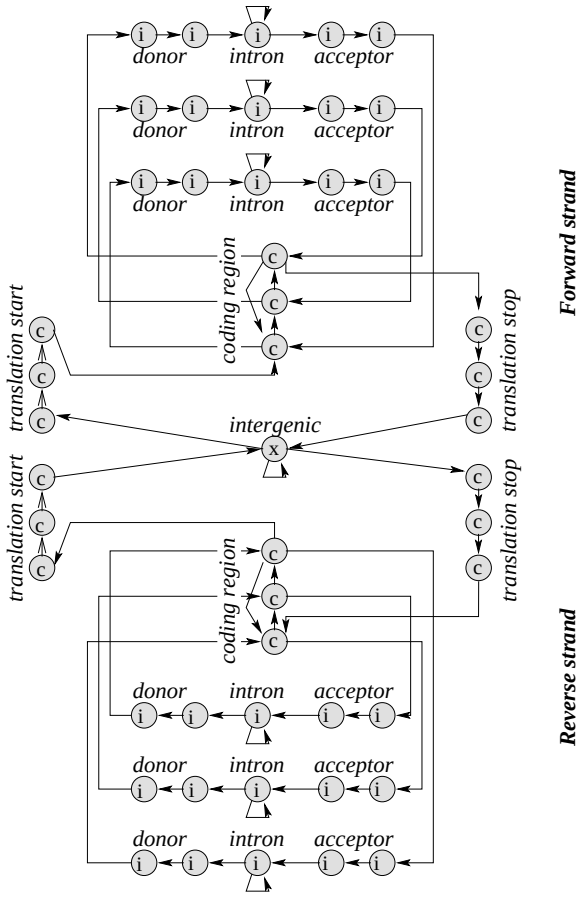


Fig. 1.3 Topology of a simple HMM gene finder. The acceptor and donor regions correspond to the signals at the ends and beginnings of introns.

using procedures analogous to those shown in Section 1.2.2. HMMs for gene finding typically use states of order between two and five.

1.2.5 More examples of biological sequence annotation with HMMs

The use of HMMs in sequence annotation is not limited to gene finding, of course; they have been used in a host of applications across the field.

One of the first applications of HMMs in bioinformatics was to segment DNA sequences into regions with similar GC content levels (Churchill, 1989). Similarly, we can partition sequence to homogeneous regions based on other criteria, for example the degree of sequence conservation in multiple species (Siepel and Haussler, 2003).

In DNA sequences, eukaryote and prokaryote gene finding is the dominant HMM application. In eukaryotic organisms the difficulty in the problem stems from the presence of introns and the often small, and highly variable, proportion of protein coding sequence in the genome (Krogh, 1997; Burge and Karlin, 1997; Stanke and Waack, 2003). The existence of alternative splicing also complicates the field, as individual positions of a sequence may be found in both intron and exon, depending on the transcript, though recent work (Cawley and Pachter, 2003; Allen and Salzberg, 2006) has moved in this direction. Gene finding in prokaryotes and viruses needs to handle overlapping genes and the problem of insufficient training data in newly sequenced genomes (Larsen and Krogh, 2003; McCauley and Hein, 2006). HMMs can also be used for other tasks related to gene finding, such as promoter detection (Ohler et al., 2001).

Proteins are generally hard to analyze from sequence only since their function is determined largely by their fold. Amino acids that are distant in the sequence may interact once the protein is folded, because they are physically close. However, HMMs can be successfully applied to recognize aspects of protein function that are governed by motifs located in contiguous stretches of the sequence.

One such example is transmembrane protein topology prediction. Transmembrane proteins are partially embedded inside the cellular membrane. The topology of such a protein identifies which regions are found in transmembrane helices (parts traversing the membrane), cytoplasmic loops (parts inside the cell), and non-cytoplasmic loops (parts extending outside the cell).

Figure 1.4 shows an overview of a simple HMM that could be used for predicting these topologies. The HMM topology enforces the simple physical constraint that cytoplasmic loops must be separated from non-cytoplasmic loops by transmembrane helices. Krogh et al. (2001) and Tusnády and Simon (1998) used similar HMMs in their topology prediction tools. A special class of transmembrane proteins, β-barrel proteins, are also successfully modeled by HMMs (Martelli et al., 2002; Fariselli et al., 2005). More generally, we can try to predict the secondary structure of arbitrary proteins, labeling each amino acid as a part of an α-helix, β-sheet or loop (Goldman et al., 1996; Bystroff et al., 2000; Aydin et al., 2006).

HMMs are closely related to the problem of aligning two DNA or protein sequences. In Section 1.5 we discuss pair HMMs, which provide a probabilistic framework for scoring pairwise alignments.

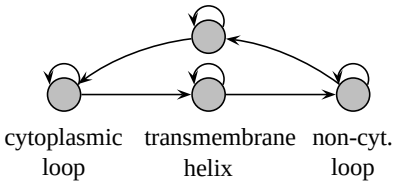


Fig. 1.4 Simplified topology of an HMM for transmembrane topology prediction

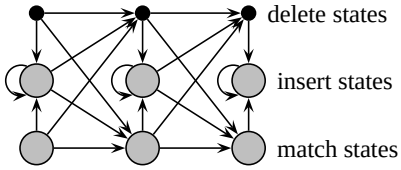


Fig. 1.5 A section of an profile HMM with three match states. Delete states are silent, that is, they do not emit any characters.

In protein sequence analysis, HMMs are often used to align the residues of a newly sequenced protein to a model of a family of homologous proteins. This is most typically done using a prominent class of hidden Markov models called profile HMMs (Krogh et al., 1994). A profile HMM has a regular structure consisting of a match state for every conserved column of the multiple alignment and insert and delete states that model insertions and deletions in the alignment, as shown in Figure 1.5. Thanks to their regular structure, they can be created automatically and stored in a database, such as Pfam (Finn et al., 2006). Computing the maximum probability path in a profile HMM is equivalent to optimizing a very simple form of multiple alignment.

We can also create a hand-crafted topology for recognizing a particular signal, protein family or fold. Examples include models to identify signal peptides (Nielsen and Krogh, 1998), and for discovering coiled-coil proteins (Delorenzi and Speed, 2002) and glycolipid-anchored membrane proteins (Gilson et al., 2006).

Schultz et al. (2006) use profile HMMs to detect recombination in HIV strains. They build a profile HMM for each known subtype, adding transitions with a low probability between states corresponding to the profiles of different subtypes. In their formulation, the annotation of a given query sequence identifies which parts belong to which subtype.

Another interesting recent use of HMMs that incorporates recombination is due to Rastas et al. (2005), who use an HMM to assist them in haplotype inference and in discovering recombination points in genotype data. From genotype data, they train a hidden model to represent approximations of ancestral haplotypes, and allow transitions between these due to recombinations over the course of evolutionary time scales. Given a genotype, which is the conflation of two haplotypes, each of which

represents a path through the network, they compute the maximum probability pair of paths that can give rise to that genotype. The sequences from these two paths are then the inferred haplotypes.

Finally, we note that although the focus of this chapter is biological sequence annotation, hidden Markov models are used for similar tasks in other domains. Speech recognition was one of the first HMM application areas (Rabiner, 1989). In natural language processing, HMMs were applied to several tasks, for example tagging the words with their parts of speech (Church, 1988), segmentation of text to topics (Yamron et al., 1998) and information extraction (Seymore et al., 1999). They can also be applied to areas as diverse as music composer recognition (Pollastri and Simoncelli, 2001) and fire detection (Müller, 2001).

1.3 ALTERNATIVES TO VITERBI DECODING

The Viterbi decoding algorithm is widely used, due to its simplicity and efficiency. It is not the only appropriate decoding algorithm for all HMM applications. This section presents several alternative decoding contexts, and appropriate algorithms for them.

1.3.1 Maximizing the number of correctly explained states: posterior decoding

Posterior decoding focuses on individual positions in the sequence, and tries to maximize the probability that they are properly explained. This is in contrast to Viterbi decoding, which computes the globally optimal state path. The most simple posterior decoding question is: what state most likely generated symbol i in the HMM output?

The most probable path is not necessarily helpful in answering this question. Many different state paths in the HMM can generate the same sequence s, and in position i, it is possible that many of them will agree on the same state. To compute the posterior probability $P(h_i = k \,|\, X)$ of state k at position i, conditioned on the entire sequence X, we add the probabilities of all paths using state k at position i. The posterior probability can be decomposed as follows:

$$\Pr(h_i = k \,|\, X) = \sum_{\ell} \frac{F_i(k, X) \cdot a_{k,\ell} \cdot B_{i+1}(\ell, X)}{\Pr(X)}, \tag{1.4}$$

where $F_i(k, X) = \Pr(h_i = k, x_1 \ldots x_i)$, the probability of generating the first i symbols of X and ending in the state k, is called the *forward probability* of state k at position i, and $B_{i+1}(\ell, X) = \Pr(h_{i+1} = \ell, x_{i+1} \ldots x_n)$, the probability of starting in state ℓ and generating the rest of the sequence, $x_{i+1} \ldots x_n$, is called the *backward probability* of state ℓ at position $i + 1$. The forward probabilities for a given sequence X and a given hidden Markov model can be computed in $O(nm^2)$ time using the

standard forward algorithm (Baum and Eagon, 1967); the backward probabilities can be computed by the backward algorithm in the same running time.

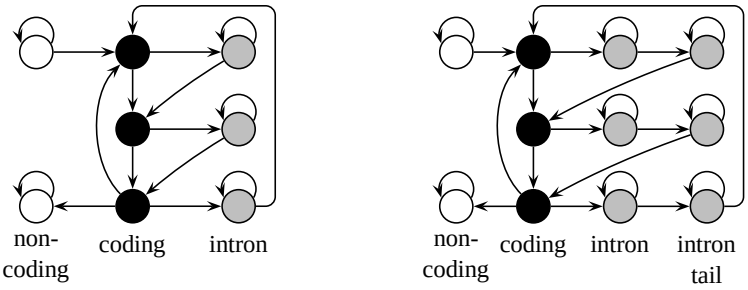
Using Formula (1.4) and the results of the forward and backward algorithms, we can compute the posterior probabilities of all states at all positions of the sequence X in $O(nm^2)$ time. Note that the posterior probability of the whole sequence $\Pr(X)$, which is the denominator in Formula (1.4), is also obtained as a side product of the forward algorithm: it is $\sum_\ell F_n(\ell, X)$.

We can use the posterior probabilities in a number of ways. A human user can simply examine them to look for interesting features; Krogh et al. (2001) display a plot of the posterior probabilities of individual states along with the most probable annotation. The plot highlights which parts of the annotation are most certain and what other hypotheses might be reasonably likely. We can also compute the posterior probability of an entire candidate sequence feature, such as an exon, by summing the probabilities of all paths sharing that feature in a specific location of the sequence. Genscan (Burge and Karlin, 1997) provides a list of the most probable alternative exons, including ones not found on the most probable path. These exons can be then tested experimentally or used as an input for further processing. Larsen and Krogh (2003) go one step further and compute the statistical significance of discovered genes, computing the expected number of genes with a given score that would occur in a random sequence of certain length.

Or, we can decode sequences using posterior probabilities. In *posterior decoding*, we choose the highest posterior probability state at each position of the sequence: $h_i^* = \arg\max_k \Pr(h_i = k \mid X)$. This approach maximizes the expected number of positions in the decoding that have the right state. By contrast, Viterbi decoding maximizes the probability of the entire state path, even though this path may have exceedingly low probability. It may be the case that the posterior decoding has better overall quality.

Still, the posterior decoding can be a composition of unrelated high probability paths. This can reach a point of ridiculousness: two adjacent states in the posterior annotation may not even be connected by an edge in the HMM. The probability of such a sequence of states being the source of the query sequence is zero: it is inconsistent with the basic assumptions encoded in the model topology.

Different authors have addressed this concern through adding a post-processing step where we attempt to maximize a different objective function. After computing all posterior state probabilities, using the forward-backward algorithm, we restrict the choice to the paths that use only transitions present in the HMM topology. Käll et al. (2005) find the path that maximizes the sum of the posterior state probabilities, trying to maximize the number of correctly predicted states. This is done by straightforward dynamic programming, similar to the Viterbi algorithm, in time $O(nm^2)$. Using a similar method, Fariselli et al. (2005) maximize the product of posterior probabilities in the postprocessing step.



(a) without multiple path problem (b) with multiple path problem

Fig. 1.6 Simple models of exon/intron structure

1.3.2 Maximizing the annotation probability: the multiple path problem

Each state in an HMM used to annotate sequences is labeled with the feature to which it corresponds. In gene finding, we label states as coming from exons, introns and so on. Each state path naturally corresponds to a sequence of labels, or an *annotation*. This annotation encapsulates the semantic meaning given to the sequence by the HMM path.

This mapping between state paths and annotations is not always one-to-one: several state paths may correspond to the same annotation. Such paths provide "alternative origins" of the sequence, but have the same semantic meaning. Thus, if we seek the most probable meaning, or annotation, for the sequence, we should add probabilities of all of these state paths.

We will describe an HMM that has multiple state paths with the same annotation as having the *multiple path problem*. Figure 1.6a shows a simplified HMM for gene finding with its state labels depicted by the state colors. If the start state of the HMM is fixed, this HMM does not have the multiple path problem, even though multiple states share the same color. Given an annotation, we can identify which single state corresponds to each black and gray position.

However, if we move to a slightly more complex model, things quickly change. The model in Figure 1.6a embodies the assumption that the nucleotide composition of introns is homogeneous. However, vertebrate intronic sequences contain a variable-length tail that is rich in nucleotides C and T (Burge and Karlin, 1997). To incorporate this information, we can include a second intron state representing such a tail, as shown in Figure 1.6b, where the new state has substantially different emission probabilities from the first. This change creates the multiple-path problem because there are always several high-probability alternatives for the transfer from the "intron" state to the "tail" state. The probabilities of all of these paths may be quite low, and Viterbi decoding may thus lead us to a completely different gene structure that results from fewer paths.

Even though the model in Figure 1.6b is a more truthful representation of real sequences than the one in Figure 1.6a, it may provide worse results when used with

the Viterbi algorithm (Brejová et al., 2004). This paradoxical conclusion results because we will be biased towards annotations with fewer uses of the intron module, since each use of that module tends to greatly drop path probabilities.

In practice, gene finders often solve this problem by fixing the number of nucleotides in the pyrimidine-rich intron tail (Burge and Karlin, 1997; Stanke and Waack, 2003; Brejová et al., 2005). The resulting model does not have the multiple path problem and can be decoded by the Viterbi algorithm.

Sometimes, though, the multiple path problem is not easily removed. In these cases, we would like to compute the most probable annotation directly. Unfortunately, this is not feasible for all model topologies. Brejová et al. (2004) constructed an HMM with 34 states for which it is NP-hard to compute the most probable annotation. As such, we are not likely to find an efficient algorithm to find the most probable annotation.

We can respond to this negative conclusion by resorting to heuristic algorithms, not guaranteed to find the most probable annotation, that perform better than the Viterbi algorithm. A popular example is the N-best algorithm (Schwartz and Chow, 1990), which was shown to give good results in several biological applications (Krogh, 1997; Krogh et al., 2001). We can also use posterior decoding, as in Section 1.3.1, and thereby join together all of the many paths that go through all states with the same label. Still, this approach will be prey to all of the other limitations of the posterior decoding technique.

However, we can characterize special classes of HMMs for which the most probable annotation can be computed efficiently. For example, for HMMs that do not have the multiple path problem, we can find the most probable annotation by the Viterbi algorithm in $O(nm^2)$ time. Vinař (2005) has shown a hierarchy of algorithms that can decode increasingly wider classes of HMMs, but at a cost of increasing running time $O(n^{d+1}m^{d+2})$ for a parameter d. In the rest of this section, we describe the most practical of these algorithms which runs in $O(n^2m^3)$ time.

This running time is feasible for analyzing protein or mRNA sequences, which are much shorter than genomic DNA. This algorithm can find the most probable labeling for a wide class of models with the multiple path problem, including the gene finding HMM shown in Figure 1.6b and models used for predicting the topology of transmembrane proteins and finding coding regions in mRNA sequences. It can also be applied as a heuristic to HMMs outside of its target class, much as the N-best algorithm can.

The main observation is that many HMMs with the multiple path problem still have a fair amount of structure in the way that sequence features flow from one to another. Specifically, for these HMMs, while many paths may represent the same annotation, the edges used to transition between the sequence features in the annotation are always the same for all of the paths. We call the edges that transition between states of different labels *critical edges*.

The *extended annotation* of a state path $h_1h_2 \ldots h_n$ is the pair (L, C), where $L = \lambda_1, \lambda_2, \ldots, \lambda_n$ is the sequence of labels of each state in the path and $C = c_1, c_2, \ldots, c_k$ is the sequence of all critical edges followed on that path. There can be several state paths with the same extended annotation; for example in Figure 1.6b,

these are the paths that differ only in position of entering the intron tail state; they all follow the same edge from grey to white.

We can extend the Viterbi algorithm to compute the most probable extended annotation. Fortunately, many HMMs (including the one Figure 1.6b) have one-to-one correspondence between extended annotations and annotations, and thus can be decoded by this algorithm. We can even test automatically if a given HMM has this property (Brejová et al., 2004), called the *critical edge condition*.

The algorithm again uses dynamic programming, summing all of the paths within every feature, to obtain the maximum probability extended annotation. In the Viterbi algorithm, we compute the values $V[i,k]$, the maximum probability of a state path for the sequence $x_1 \ldots x_i$ over all paths ending in state k. In the extended Viterbi algorithm, we instead compute $L[i,k]$, the maximum probability of an extended annotation (L,C) of the sequence $x_1 \ldots x_i$, where the model is in state k at position i; that is, $L[i,k] = \max \Pr(x_1 \ldots x_i, (L,C), h_i = k)$.

At each step, we examine all possible durations of the last segment with the same label and instead of choosing the single most probable path in that segment with that length, we compute the sum of all possible appropriate-length state paths in this segment. If the segment starts at position j of the sequence, let $P[j,i,k,\ell]$ be this sum; it is the probability of generating the sequence $x_j \ldots x_i$, starting in state k, and ending in state ℓ, using only states with the same label λ (both states k and ℓ must also have this same label). We get the following recurrence:

$$L[i,k] = \max_{j \le i} \max_{\ell} \max_{\ell'} L[j-1,\ell'] \cdot a_{\ell',\ell} \cdot P[j,i,\ell,k] \tag{1.5}$$

We compute the values of L in order of increasing i. For each i, we compute all relevant values of $P[j,i,k,\ell]$ in order of decreasing j by the following recurrence (this is similar to the standard backward algorithm):

$$P[j,i,k,\ell] = \sum_{\ell' \text{ with label } \lambda} e_{k,x_j} \cdot a_{k,\ell'} \cdot P[\ell',\ell,j+1,i] \tag{1.6}$$

This algorithm finds the most probable extended annotation in any HMM in $O(n^2m^3)$ time.

1.3.3 Finding many paths: sampling from the posterior distribution

Instead of finding the most probable state path, we can also sample a collection of state paths according to the conditional probability distribution $\Pr(H \mid X)$ defined by the HMM. The following algorithm for sampling from HMM was introduced by Zhu et al. (1998).

We first pre-compute all values of $B_i(k,X)$ by the backward algorithm as outlined in Section 1.3.1. In the first step, we randomly choose initial state h_1, where the probability of starting in state k is proportional to $s_k \cdot B_1(k,X)$. After that, in the i-th step, we choose the next state, h_i, with probability proportional to $a_{h_{i-1},h_i} \cdot B_i(h_i,X)$. The probability of choosing path $H = h_1, \ldots, h_n$ by this randomized algorithm is

exactly $\Pr(H \mid X)$, so we are sampling from the conditional distribution of state paths, given the output sequence X.

Sampling may be useful if we need to provide several alternative annotations, instead of a single prediction. For example, several possible high-probability annotations may be needed for the purpose of experimental verification. In gene finding, genes may have several splicing variants; the same DNA sequence is transcribed into multiple proteins using different combinations of splice sites. SLAM (Cawley and Pachter, 2003) and AUGUSTUS (Stanke et al., 2006a) use this method to generate multiple gene annotations as potential alternative transcripts. On the other hand, as each of these will likely have extremely low probability, they are likely unreliable as overall predictions for the entire sequence.

1.4 GENERALIZED HIDDEN MARKOV MODELS

The lengths of features found in biological sequences can come from extremely complex distributions. Unfortunately, simple HMMs are not necessarily effective at modeling these distributions. For example, the simplest way to model a region of variable length is with a single HMM state that has a transition to itself (a self-loop), with transition probability p. The probability that the HMM stays in such a state for exactly ℓ steps is $(1-p)p^{\ell-1}$, so the distribution of lengths of regions generated by this state will be geometric. However, length distributions of biological sequence elements are far from geometric. Figure 1.7a shows length distribution of internal exons in human genes and its best approximation by a geometric distribution.

This section shows a variety of methods to address this problem. Some involve changes to the generative behaviour, to improve the ability to model more complicated distributions. The simplest such approaches can substantially increase the decoding time, from $O(nm^2)$ to $O(n^2m^2)$; for long DNA sequences, this order of magnitude change is unacceptable. We thus present methods that compromise between modeling accuracy and decoding time.

1.4.1 Generalized HMMs and explicit state duration

In *generalized HMMs*, self-loop transitions are replaced by states generating their state durations explicitly. Upon entering a state, the generative model first chooses the duration d, which is the number of symbols that will be generated in this state. For each state h, the probability distribution δ_h that determines these random variables is explicitly represented in the model. After d symbols are generated in the state, the model follows a transition to a new state.

To compute the most probable state path that generates a particular sequence of symbols, we must modify the Viterbi algorithm. In each step of the dynamic programming, in addition to examining all potential last transitions, we also have to consider all possible durations of the last state. If $V[i,k]$ is again the probability of the most probable path generating the first i symbols $x_1, \ldots, x_i$ and finishing in state k, assuming that in the next step the model will transit out of state k or finish

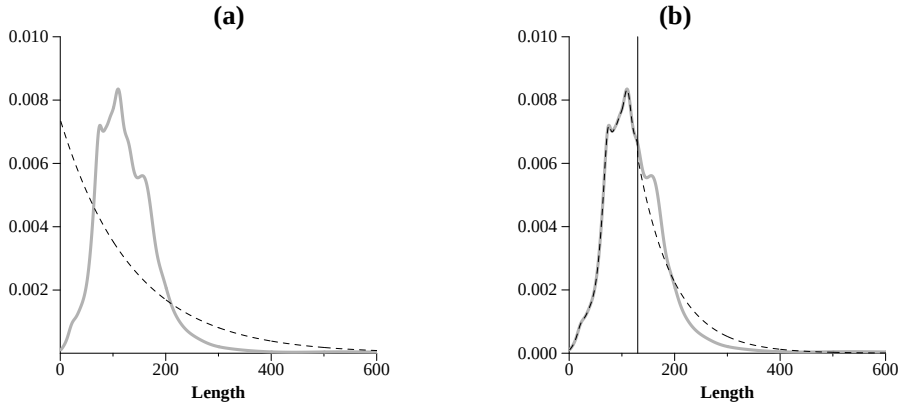


Fig. 1.7 Length distribution of internal exons on human chromosome 22. (a) Best fit by geometric distribution. (b) Best fit by geometric-tail distribution with $t = 130$.

the generation process, then the recurrence characterizing the dynamic programming must change as follows:

$$V[i,k] = \max_{1 \le j \le i} [emit(k,j,i) \cdot \delta_k(i-j+1) \cdot \max_{\ell} V(j-1,\ell) \cdot a_{\ell,k}], \qquad (1.7)$$

where $emit(k,j,i)$ is the emission probability of generating the sequence of symbols $x_j, \ldots, x_i$ in state k. The straightforward implementation of this dynamic programming gives an $O(n^3m^2)$ running time, where n is the length of the sequence and m is the number of the states, since the computation of $emit(v,j,i)$ takes $O(n)$ time in the worst case. However, it is possible to reduce the running time to $O(n^2m^2)$ using a pre-computation that requires $O(nm)$ time, after which it is possible to compute $emit(v,j,i)$ in constant time for any i and j (see Mitchell et al. (1995) for details).

This sort of runtime, which is quadratic in the length of the query sequence, is reasonable for short sequences, such as proteins. It is not feasible for long DNA sequences. Two straightforward solutions to reduce the running time are used in practice.

First, we can place an upper bound of d on the number of characters produced by each state (as in Rabiner (1989)). Then, the running time will be $O(ndm^2)$. In speech recognition applications, it is usually possible to keep the bound d relatively small, as the state durations may be phonemic durations, so this approach yields a reasonable decoding algorithm with practical running time. However, such a bound is often hard to find in biological applications.

Second, we observe that we can stop our dynamic programming search for lengths that may be emitted by the current state whenever $emit(k,j,i) = 0$. For example, this is a common stopping condition for exon states in gene finding: we can stop searching upon reading an in-frame stop codon. Burge and Karlin (1997) used this

approach in their gene finder Genscan to model exons with generalized states and complex distributions, still achieving reasonable decoding runtimes. Unfortunately, this approach does not extend to intron distributions: there is no common sequence forbidden to them.

1.4.2 Distributions with geometric tails

One way of decreasing the running time, even when no upper bound on the length of the state durations is available, is to restrict the family of length distributions allowed in the generalized states. One example of this approach is due to Brejová and Vinař (2002), which restricts the family of durations to ones with *geometric tails*. Such distributions are robust enough to characterize the lengths of many important biological elements effectively.

A geometric-tail distribution for the duration of a state is the joining of two distributions: the first part is an arbitrary length distribution, and the second part is a geometric tail. Specifically, there is a parameter t where, for values of i less than or equal to t, the probability $\delta_k(i)$ is explicitly set, while for values of i greater than t, $\delta_k(i) = \delta_k(t) \cdot q_k^{i-t}$. The values of $\delta_k(t)$ and q_k are set to maximize the likelihood of the length distributions of training data, and the explicit probabilities found in $\delta_k(i)$ for $i < t$ are set to match observed values after smoothing.

Such distributions can model the lengths of many functional segments of biological sequences, even with small values of the tail start parameter t. For example, Figure 1.7b shows the geometric-tail distribution with $t = 130$ that best approximates the length distribution of human internal exons.

Brejová and Vinař (2002) emphasize models with small values of the parameter t because they also design an efficient decoding algorithm with $O(nmt + nm^2)$ runtime. The Viterbi algorithm for generalized HMMs in recurrence (1.7) explicitly considers all possible durations of state k. For geometric-tail distributions, we can reduce the running time by distinguishing between two cases: durations less than or equal to t_k, and durations longer than t_k.

In particular, let $Q[i,k]$ be the probability of the most probable path generating the first i symbols of the sequence, and spending at least last t_k steps in state k. To compute the value of $Q[i,k]$, we consider two cases: either the i-th character extends the duration of the state k, which was already at least t_k, or generating the i-th character brings the duration of state k to exactly t_k steps. The value of $Q[i,k]$ can be then used in computing $V[i,k]$, instead of checking all durations longer than t:

$$V[i,k] = \max \begin{cases} Q[i,k], & \text{(duration at least } t_k\text{)} \\ \max\limits_{1 \le d < t_k} [emit(k, i-d+1, i) \cdot \delta_k(d) & \\ \qquad \cdot \max\limits_{\ell} V[i-d,\ell] \cdot a_{\ell,k}] & \text{(duration less than } t_k\text{)} \end{cases} \tag{1.8}$$

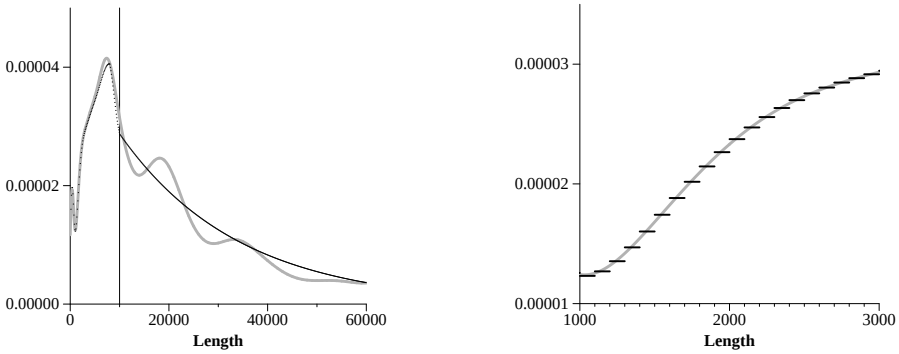


Fig. 1.8 Step-function approximation of intergenic length distribution in human chromosome 22. The right plot shows detail of the step-function character of the distribution.

$$Q[i,k] \quad = \quad \max \begin{cases} Q[i-1,k] \cdot q_k \cdot e_{k,x_i} & \text{(duration more than } t_k\text{)} \\ emit(k, i-t_k+1, i) \cdot \delta_k(t_k) \cdot \max\limits_{\ell} V[i-t_k, \ell] \cdot a_{\ell,k} & \\ \qquad \text{(duration exactly } t_k\text{)} & \end{cases} \tag{1.9}$$

A straightforward dynamic programming algorithm implemented based on this recurrence would take $O(ntm^2)$ time, which Brejová and Vinař (2002) improve to $O(nmt + nm^2)$ by precomputing values of $\max_\ell V[i,\ell] \cdot a(\ell,k)$.

In gene finding, this technique was used in ExonHunter (Brejová and Vinař, 2002; Brejová et al., 2005) to model the length distributions of exons and introns; the gene finder Augustus (Stanke and Waack, 2003) uses a similar approach shown in 1.4.3 to model the length distributions of introns.

The distributions of much longer features can also be modeled in an extension of this approach. The gene finder ExonHunter (Brejová et al., 2005) models the lengths of intergenic features, for which a simple geometric tail distribution would require $t \approx 10^4$, by replacing a single-state model of intergenic region with a two-state model that allows one to approximate this distribution. The first state generates symbols in blocks of length $\sqrt{t}$, where the number of blocks is determined by a geometric-tail distribution, where the tail begins at $\sqrt{t}$. The second state generates only up to $\sqrt{t}$ symbols, with uniform length distribution. This method replaces the original length distribution with a step-function approximation, where the steps happen at intervals of $\sqrt{t}$, as shown in Figure 1.8. The model that represents this distribution can be decoded in $O(nm\sqrt{t} + nm^2)$ time, which is practical even for the large values of t needed to model intergenic regions.

1.4.3 Gadgets of states

An alternative way of avoiding the geometric length distributions for individual states in hidden Markov models is to model a single sequence element by multiple states

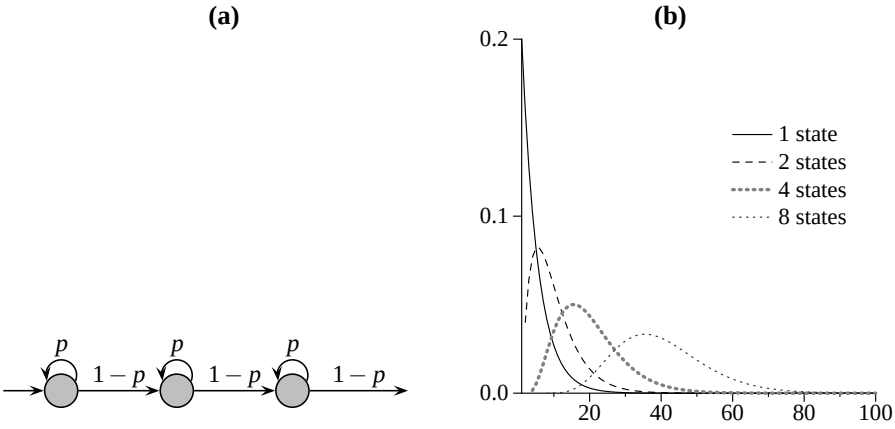


Fig. 1.9 (a) A gadget of states generating non-geometric length distributions; (b) depending on the number of states and probability p, different distributions from a subclass of the discrete gamma distributions $\Gamma(p\ell, 1)$ can be generated.

instead of a single state. Durbin et al. (1998) (recently also re-examined by Johnson (2005)) discuss several ways to model non-geometric length distributions by replacing a single state with a group of states that share the same set of emission probabilities. Transitions are added inside this group so that the probability of staying within the group for ℓ steps is close to the probability that the modeled feature has length ℓ.

Consider the gadget in Figure 1.9a. The left-most transition is the sole entry point to the sub-model, and the right-most transition is the exit. If the gadget consists of n states, the probability of generating a feature of length $\ell > n$ is $f(\ell) = \binom{\ell-1}{n-1} p^{\ell-n}(1-p)^n$, which can be used to model a wide variety of gamma distributions (see Figure 1.9b). One example of this approach is found in Larsen and Krogh (2003), who used three copies of their codon model, each with its own self-loop, to model the length distribution of genes in bacteria.

The geometric-tail distributions with parameter t discussed in the previous sections can be generated by a gadget of t states, shown in Figure 1.10; for $i < t$, the probability of generating a feature with length i is $\prod_{j<i}(1-p_j)p_i$, while if $i \geq t$, then $\delta_k(i) = \prod_{j=1\ldots t-1}(1-p_j)q^{i-t-1}(1-q)$.

Such a construction was used by Nielsen and Krogh (1998) for protein modeling and by Stanke and Waack (2003) in gene finding. The modified Viterbi algorithm for geometric-tail distributions shown in the previous section is essentially equivalent to running the classical Viterbi algorithm on such an HMM, though it is more memory efficient, since the Viterbi probabilities $V[i,k]$ are not stored for the extra states within the gadget.

In general, one can use any topology of states in a gadget; distributions that can be represented in such a way are called *phase-type distributions*, and they play an important role in queuing and systems theory (see Commault and Mocanu (2003)

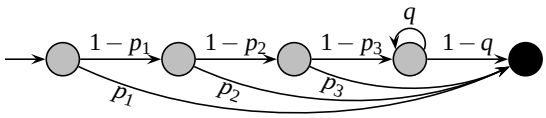


Fig. 1.10 A gadget of states generating a geometric-tail length distribution with $t = 4$. The black circle represents the first state of the next submodel of the HMM.

for a recent overview). This approach of using phase-type distributions suggests what appears to be an ideal framework for modeling general length distributions in HMMs: fix the number of states in each gadget depending on the desired running time, and then find the best approximation of the length distribution observed in training data. With increasing size of the gadget, we can approximate any desired length distribution arbitrarily well (Asmussen et al., 1996).

Unfortunately, most gadgets, such as the one shown in Figure 1.9a, introduce the multiple path problem discussed in Section 1.3.2, so Viterbi decoding is inappropriate for them. Indeed, Vinař (2005) showed that the result of decoding the HMM with a gadget shown in Figure 1.9a with Viterbi decoding is equivalent to the result of decoding an HMM where the same feature has essentially a geometric length distribution.

This unhappy result leaves us with two options: compute the most probable labeling by the extended Viterbi algorithm from Section 1.3.2, or use other decoding strategy, such as posterior decoding. Note that since the extended Viterbi runs in quadratic time in the length of the sequence, the former strategy is no better than using arbitrary length distributions and the algorithm from Section 1.4.1.

1.5 HMMS WITH MULTIPLE OUTPUTS OR EXTERNAL INFLUENCES

In the previous sections, we have considered HMMs that modeled a single DNA or protein sequence and its annotation. This approach, however, is not appropriate to the more contemporary domain in which we may have much external information that is helpful in annotating a sequence accurately. In this section, we consider a variety of ways in which HMMs can incorporate such external evidence. Many of these change the structure of the output of the HMM, while others influence the decoding algorithms.

Perhaps the most readily available source of information is predicted evolutionary homology. Great amounts of DNA and protein sequences are publicly available in databases such as GenBank (Benson et al., 2000). For a given sequence of interest we may find its likely homologs in a database and exploit typical patterns of evolution to improve the annotation. Functionally important regions usually evolve much more slowly and are well conserved even between relatively distant species; on the other hand, random mutations often accumulate more quickly in regions with fewer

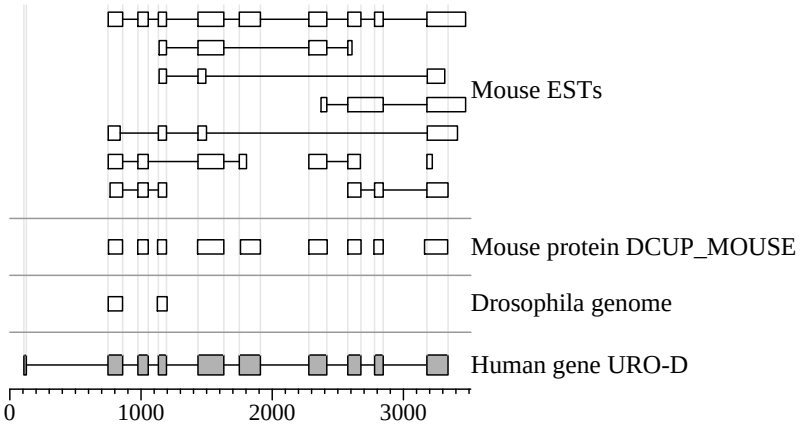


Fig. 1.11 Evidence supporting annotation of human URO-D gene. Significant alignments from fruit fly genome, known mouse proteins, and mouse ESTs are represented as boxes.

functional constraints (Siepel et al., 2005). Another source of evidence is the results of biological experiments aimed at elucidating sequence features and their function. For example, in gene finding, EST sequencing and tiling array experiments may confirm that certain regions of the genome are exons.

An example of additional information in gene finding is illustrated in Figure 1.11. The figure shows significant alignments of a distantly related genome, known proteins, and expressed sequence tags to a genomic regions containing the human URO-D gene. In this case, the additional evidence provides a human observer enough information to have a very good idea about the structure of the gene. The process of incorporating such information into the automatic annotation that results from decoding an HMM, on the other hand, is not necessarily nearly as simple: we must design systems that are efficient to decode and efficiently trained, and which are able to accommodate errors and imprecisions in the external sources of information.

1.5.1 HMMs with multiple outputs

One way of incorporating additional evidence into HMMs is to represent each source of evidence as a new *informant* sequence. We can then extend the HMM to generate the informant sequences as part of its output, alongside with the original query sequence whose annotation we seek.

These extensions are perhaps most easily described in the framework of Bayesian networks. A Bayesian network is a generative probabilistic model whose output is N variables. The dependencies among these variables are shown by representing the variables as the vertices of a directed acyclic graph. We generate values for the variables in topological order, so that the values of all of the variables that are the predecessors of a variable are determined before its value. To be more specific, consider a variable X, with parents $X_1, \ldots, X_k$. The parameters of the Bayesian

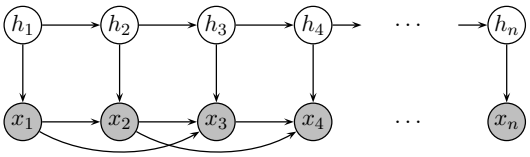


Fig. 1.12 A hidden Markov model with second-order states, represented as a Bayesian network. The top row of variables represents the state path, $h_1, \ldots, h_n$, through the HMM. The bottom row represents the emitted DNA sequence, $x_1, \ldots, x_n$. The conditional probabilities of the Bayesian network are defined by the initial, transition, and emission probabilities of the HMM: $\Pr(h_1) = s_{h_1}$, $\Pr(h_i|h_{i-1}) = a_{h_i,h_{i-1}}$, and $\Pr(x_i|h_i, x_{i-1}, x_{i-2}) = e_{h_i,x_{i-2},x_{i-1},x_i}$. The observed variables, which indicate the DNA sequence, are shaded in the figure.

network specify the conditional probability $\Pr(X = x \mid X_1 = x_1, \ldots X_k = x_k)$, for all combinations of the values $x, x_1, \ldots, x_k$. Once the values of the parent variables are fixed, we can generate the value of X from this conditional distribution.

HMMs easily fit into this Bayesian network framework: an HMM that generates a sequence of a fixed length n can be represented as a Bayesian network with $2n$ variables: for each emitted symbol, we have one variable representing the symbol itself and one variable representing the hidden state emitting the symbol (see Figure 1.12). We can also represent higher order states by including additional edges between the observed variables as demonstrated in the figure.

One approach to incorporating external evidence into the HMM is to represent the evidence sources by informant sequences, which also depend on the hidden states of the network. We translate each external source into a sequence of n symbols from a finite alphabet, where each symbol in the informant sequence must correspond to one symbol of the query sequence. For example, we can encode a genome-to-genome alignment as a sequence of n symbols from the alphabet $\{0, 1, .\}$ by characterizing each base of the query DNA sequence as "aligned with match" (symbol '1'), "aligned with mismatch" (symbol '0'), or "unaligned" (symbol '.'); This is the encoding scheme used in the gene finder TwinScan (Korf et al., 2001).

We can represent this approach by adding a variable for each informant sequence at each sequence position to our Bayesian network. If we have $k - 1$ external information sources, the network will have $n(k + 1)$ variables: n state variables, n variables for the query sequence and n variables for each of the $k - 1$ informant sequences. The simplest way to add these new variables is to make the symbols of all k sequences conditionally independent given the state at each position. Figure 1.13 shows such a model for $k = 2$. Korf et al. (2001) used this approach to incorporate genome-to-genome alignments into gene finding. Pavlović et al. (2002) transformed the outputs of a collection of gene finding programs into informant sequences, and used this same sort of approach to join their predictions into a single prediction; their system does not even involve the query DNA sequence as one of the network's outputs.

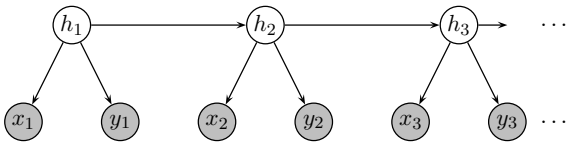


Fig. 1.13 A representation of the generative probabilistic model of the TwinScan gene finder (Korf et al., 2001) as a Bayesian network. The h_i variables each represent one state of the HMM; variable x_i represents one nucleotide of the query DNA sequence, and y_i represents the conservation between this nucleotide and some other genome, over a special alphabet with symbols for matched, mismatched and unaligned positions. (TwinScan actually uses emission tables of order five, which can be depicted by adding additional edges, as in Figure 1.12.)

Training and decoding of these extended HMMs is analogous to regular HMMs: maximum likelihood parameters can be obtained by simple frequency counting from annotated sequences, and we can straightforwardly modify the Viterbi algorithm (and other decoding algorithms) to account for the multiple emission probabilities in each step. The main limiting factor of these models is not their algorithms, but is the assumption of conditional independence between individual output sequences, which is clearly violated in most applications.

Instead, when the evidence consists of multiple alignment of sequences known to have evolved from a common ancestor, we can use *phylogenetic HMMs*, a model design that reflects known evolutionary relationships between those sequences. In particular, we can arrange the Bayesian network so that the topology of the network is identical to the phylogenetic tree representing the evolutionary history of the sequences, as in Figure 1.14, which shows a model of a human query sequence, and additional sequences from mouse, rat, and chicken. In this Bayesian network, we can partition all sequence variables into two sets at every position i: the set of observed variables O_i, corresponding to the sequences in the leaves of the phylogenetic tree, and the set of unobserved variables U_i, corresponding to the unknown ancestral sequences.

The unobserved variables complicate both training and decoding. To train the model, we must use the EM algorithm instead of simple frequency counting (Dempster et al., 1977). For decoding, at each position i and for each state h_i, we need to compute the likelihood of the corresponding tree submodel $\Pr(O_i \mid h_i)$. This probability can be computed from the probability distribution $\Pr(O_i, U_i \mid h_i)$ defined by the phylogenetic tree model by marginalizing unobserved variables:

$$\Pr(O_i \mid h_i) = \sum_{U_i} \Pr(O_i, U_i \mid h_i) \tag{1.10}$$

The number of terms in this sum is exponential in the number of unobserved variables. However, since the generative model has a tree structure, we can compute this sum in time linear in the number of all variables by using Felsenstein's peeling

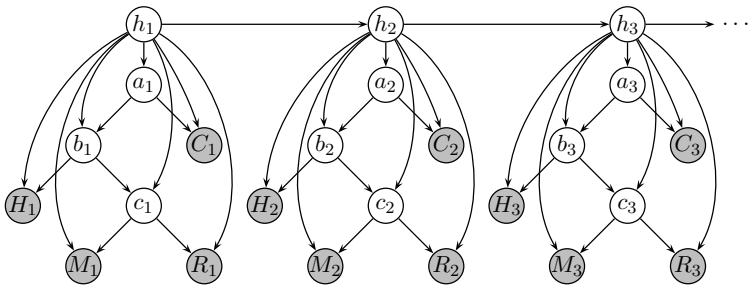


Fig. 1.14 A simple phylogenetic hidden Markov model depicted as a Bayesian network. Each variable h_i represents one state of the HMM, the variables H_i, M_i, R_i, C_i each represent single positions of human, mouse, rat and chicken from one column of a multiple genome alignment, and the variables a_i, b_i, c_i represent the unknown ancestral sequences. Observed variables are shaded. For example, the value of H_i depends on its ancestor b_i and on the HMM state h_i. The state determines mutation rate, since mutations occur more frequently in non-coding regions.

algorithm (Felsenstein, 1981), which performs dynamic programming by starting at the leaves and proceeding to the root of the tree.

We can introduce higher order states for the observed variables, as described at the beginning of this section. However, introducing higher order states for the unobserved variables is more complicated: it requires substantial modification of the decoding algorithm (Siepel and Haussler, 2003), and the running time becomes exponential in the order of the states.

Another modification of phylogenetic HMMs (Gross and Brent, 2005) involves rooting the phylogenetic tree in the query sequence rather than in the common ancestor (see Figure 1.15). The advantage of this approach is that the resulting probability distribution can be decomposed into a product of two terms: the probability that the HMM generates the query sequence and the contribution from the variables introduced by the other sequences. The emission and the transition probabilities of HMM states can be trained and tuned separately as in a single-sequence gene finder, and the parameters required for including additional evidence can be trained afterward.

An important issue is the parametrization of random variables associated with the query and informant sequences. In phylogenetic HMMs, most variables have two parents: the state variable and the parent in the phylogenetic tree. Thus if the alphabet size is σ, the number of states is m, and the number of sequences is N, we must train $\Theta(Nm\sigma^2)$ parameters. We can reduce this number by employing a nucleotide substitution model based on a standard continuous Markov chain model of evolution. For example, the simplest Jukes–Cantor model (Jukes and Cantor, 1969), which assumes a uniform rate for all single-point mutations, requires only a single parameter per sequence and state. In more complex models of evolution, such as the general reversible model of Rodriguez et al. (1990), the substitution rate

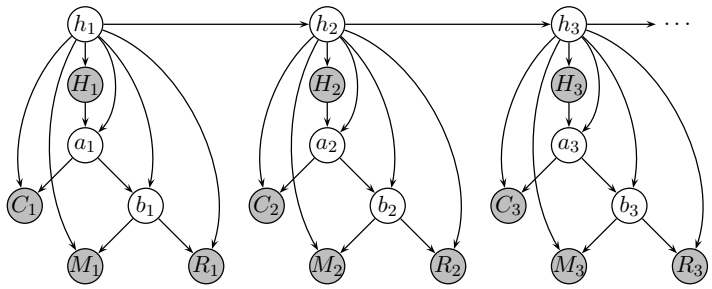


Fig. 1.15 Modified phylogenetic hidden Markov model, with query sequence positioned at the root of the phylogenetic tree.

matrix (requiring $\Theta(\sigma^2)$ parameters for each state) is shared among all branches of the phylogenetic tree, and one parameter corresponding to the branch length of an edge in the phylogenetic tree, needs to be trained for each sequence and state. Using such a model of evolution will reduce the number of parameters to $\Theta(Nm + m\sigma^2)$, a substantial savings even for moderate number of species.

Phylogenetic HMMs were first introduced in evolution studies (Yang, 1995; Felsenstein and Churchill, 1996). Goldman et al. (1996) were the first to apply them for sequence annotation, in the problem of secondary structure prediction. As genomes of multiple organisms have become available, phylogenetic HMMs have been applied to genomic sequences, for tasks such as gene finding (Pedersen and Hein, 2003; McAuliffe et al., 2004; Siepel and Haussler, 2004; Gross and Brent, 2005) and identifying conserved elements in genomes (Siepel et al., 2005). Phylogenetic HMMs are also useful for finding overlapping genes in compact viral genomes (McCauley and Hein, 2006).

The accuracy of HMM when used to analyze protein sequences can also be improved by using multiple sequence alignments of several proteins that are known to be homologous with a query sequence. However, we typically do not know the phylogenetic tree representing the evolution of these proteins. Instead, researchers have developed variants of HMMs that emit a profile specifying the relative frequency of each amino acid at each position of the sequence. Unlike phylogenetic HMMs, these models do not capture the strong correlation between closely related sequences, but only summarize the features of the many rows of the alignment. However, they require far simpler parameter estimation. HMMs emitting profiles were used to predict secondary structure of proteins by Bystroff et al. (2000), topology of β-barrel membrane proteins by Martelli et al. (2002), and topology of helical transmembrane proteins by Viklund and Elofsson (2004).

1.5.2 Positional score modification

We can incorporate external evidence into an HMM using other methods besides Bayesian network approaches. In an HMM, the joint probability $\Pr(H, X)$ of se-

quence X and state path H is computed as a product of emission and transition probabilities (see Equation (1.1)). The methods presented in this section place additional factors into this product, while keeping the decoding algorithm viable.

All possible annotations of a particular sequence are represented as different state paths through the HMM. Consider a piece of additional evidence E. It can be seen as a probabilistic hypothesis about the true annotation, whose validity depends on whether E comes from a believable source: if the origin of the evidence is trustworthy (with some probability P_E), then only paths from some set H_E should be considered. On the other hand, with probability $1 - P_E$, the evidence is untrustworthy and we should disregard it.

For example, in transmembrane topology prediction, we may see a motif that suggests that the i-th amino acid in the query sequence is found inside the cytoplasm. Then the set H_E consists of all paths through the HMM that mark the i-th amino acid as being from a cytoplasmic loop, and the probability $(1 - P_E)$ is the probability that the match is not a real functional occurrence of this motif, and we should disregard the evidence entirely.

When given such an evidence, we recognize two events: E_+ (the evidence is correct), and E_- (the evidence is wrong). We can write:

$$\Pr(H, X \mid E) = P_E \cdot \Pr(H, X \mid E_+) + (1 - P_E) \cdot \Pr(H, X \mid E_-) \tag{1.11}$$

Note, that $\Pr(H, X \mid E_+) = 0$ for paths H not found in H_E; if the evidence is correct, it is specifically eliminating certain paths from being possible. If the evidence is wrong, it should have no effect on predictions, and therefore we say $\Pr(H, X \mid E_-) = \Pr(H, X)$. If we already know that $H \in H_E$, additional evidence does not give us any new information, and addition of such evidence should not change relative probabilities of paths; consequently, we can say $\Pr(H \mid H_E, X) = \Pr(H \mid E_+, X)$. Finally, we assume (obviously unrealistically) that the probability of the sequence should be independent of the event E_+, and we can say $\Pr(X) = \Pr(X \mid E_+)$.

Using these assumptions, we obtain after simple manipulation the following updated probability distribution over all possible annotations:

$$\Pr(H, X \mid E) = \begin{cases} (1 - P_E) \cdot \Pr(H, X), & \text{if } H \notin H_E, \\ \left(1 - P_E + \frac{P_E}{\Pr(H_E \mid X)}\right) \cdot \Pr(H, X), & \text{if } H \in H_E. \end{cases} \tag{1.12}$$

Intuitively, the probabilities of all paths that agree with the evidence are multiplied by a factor greater than one, and probabilities of all paths that do not agree with the evidence are multiplied by a factor smaller than one. The relative probability of paths within each category remains unchanged.

The computational complexity of decoding under this new probabilistic model depends on the form of the set H_E of paths that are consistent with evidence. If H_E contains all the paths that annotate a point in the sequence with a particular label, or with any label from a set of labels, we can slightly modify the Viterbi algorithm

to compute the most probable state path. The quantity $\Pr(H_E \mid X)$ needed for the bonus factor can be obtained by the forward–backward algorithm.

This method was first derived and used in a gene finding program GenomeScan (Yeh et al., 2001) to incorporate protein homology into gene finding. The same method was also used to improve prediction of transmembrane protein topology by Xu et al. (2006a). In their case, the evidence was composed of motif hits that indicate strong preference for cytoplasmic or non-cytoplasmic loops at certain sites in the sequence.

A disadvantage of the GenomeScan approach is that it is unclear how to integrate multiple pieces of additional evidence (such as multiple protein hits or multiple motifs), particularly if they are not independent. In an attempt to solve this problem, the next method incorporates evidence in the form of additional multiplicative terms at each position of the sequence. An important difference is that given a particular alignment, GenomeScan method alters the probability at one position only, while in what follows, we boost the probability independently at each position covered by the alignment.

Assuming independence between the sequence X and all additional evidence E, we can use Bayes' rule to obtain:

$$\Pr(H \mid X, E) \propto \Pr(H \mid X) \cdot \frac{\Pr(H \mid E)}{\Pr(H)} \tag{1.13}$$

Though this independence assumption is not true in practice, we can often limit dependencies by avoiding using the same features of the sequence in both the HMM and the additional evidence. For example in gene finding, the HMM mostly models short windows of the sequence (signals, local coding potential, *etc.*), while the additional evidence may represent database searches, such as alignments to EST or protein sequences.

Whether we can develop an efficient decoding algorithm depends mostly on the family of probability distributions that we use to represent the contribution of the additional evidence $\Pr(H \mid E)/\Pr(H)$. In the simplest case, we assume positional independence, for both the posterior probability conditioned on the evidence $\Pr(H \mid E) = \prod_{i=1}^{n} \Pr(h_i \mid E)$ and the prior probability $\Pr(H) = \prod_{i=1}^{n} \Pr(h_i)$. To partially compensate for the positional independence assumption, we can add a scaling factor $\alpha < 1$ as follows:

$$\Pr(H \mid X, E) \propto \Pr(H \mid X) \cdot \left(\frac{\Pr(H \mid E)}{\Pr(H)} \right)^{\alpha}. \tag{1.14}$$

In this particular scenario, we can easily modify the Viterbi algorithm to find the most probable annotation H given both sequence X and evidence E in time linear in the length of the sequence.

For a single source of evidence, we can directly estimate the posterior probabilities $\Pr(h_i \mid E)$ from a training data set. However, multiple sources of evidence would typically present many combinations of local information, requiring exponential number of parameters to train. Brejová et al. (2005) developed a method for

expressing and combining information from several sources of additional evidence using partial probabilistic statements to express the implications of the evidence and quadratic programming to combine all the statements concerning a particular position in the sequence into a posterior distribution $\Pr(h_i \mid E)$.

In the context of gene finding, the method of multiplying $\Pr(H, X)$ by additional factors was successfully used to incorporate variety of sources of information (such as genome, EST, and protein alignments) in a single model; two examples are HMMGene by Krogh (2000), and ExonHunter by Brejová et al. (2005).

Stanke et al. (2006b) designed a method that tries to overcome the positional independence assumptions. Let us assume that the evidence E is expressed as a set of "hints": intervals in the query sequence. In the simplest case, each hint supports a single state of the generalized HMM (more complex variations are possible). We say that a given state path is *compatible* with hint (i, j) if the part of the query sequence $x_i \ldots x_j$ is all generated in the state supported by the interval. Otherwise, we say that the state path is *incompatible*. For example in gene finding, we can represent EST alignments as a set of intervals, each supporting an exon state in the HMM.

Each hint is assigned a position in the sequence at its right end. Only a single hint e_i is allowed to end at each position i. Also, if there is no hint ending at position i, we will say $e_i = \Cap$, corresponding to a vacuous hint. We will create a model that will not only generate the sequence X, but also the sequence of hints as follows:

$$\Pr(H, X, e_1, \ldots, e_n) = \Pr(H, X) \cdot \prod_{i=1}^{n} \Pr(e_i \mid H, X) \tag{1.15}$$

The probability $\Pr(e_i \mid H, X)$ is either $q^{\Cap}$, if the hint at position i is $\Cap$, or q^+, if the hint is compatible with H, or q^- if the hint is incompatible with H. These parameters are trained by frequency counting on the training data. Note, that this model is not truly a generative model for hints, since we do not generate the left ends of the hints; yet, we use them to determine compatibility or incompatibility of each state path. The Viterbi algorithm can be again easily modified to accommodate these interval hints, and if $q^+ > q^-$, it takes asymptotically no longer than the underlying decoding of the generalized HMM.

The interval hints were used in the gene finder AUGUSTUS+ (Stanke et al., 2006b). They enforce better consistency of final predictions with the evidence, since the bonus factor q^+ is not awarded for state paths that match an interval only partially.

1.5.3 Pair hidden Markov models

In the previous sections, we have reviewed several methods that break the problem of sequence annotation into two steps. First, a general search tool is used to identify local alignments between the query sequence and a sequence database. Next, this information is incorporated using some HMM-based method. The main disadvantage of the two-step approach is that the initial general-purpose alignment algorithm does not take into account the structure of the annotation problem.

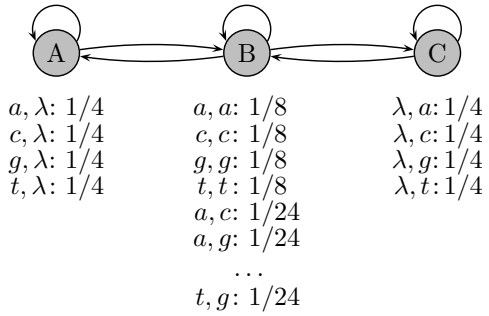


Fig. 1.16 A simple pair HMM. The symbol λ in the emission probability tables represents empty string. State B generates the ungapped portion of the alignment. State A generates characters only in the first sequence, and state C generates characters only in the second sequence. The alignment gaps induced by states A and C have geometrically distributed lengths.

For example, in gene finding, alignments of a protein or EST with the query DNA may extend beyond exon boundaries to surrounding introns, and alignments of two homologous genes may have misaligned splice sites. Such mistakes are propagated to the second stage, and may affect the accuracy of gene finding.

This problem can be avoided by simultaneously annotating and aligning two sequences, in a single step. This process can be modeled by a *pair HMM*. Pair HMMs are HMMs that generate two sequences at the same time, but where a state of a model can generate a character in one sequence or both sequences. Pairs of characters generated in the same step correspond to homologous positions from the two sequences. If only one character is generated in a given step, it corresponds to a sequence position in that sequence with no homolog in the other sequence, due to an insertion or deletion. Simple pair HMMs, such as the one in Figure 1.16, can be used to represent a traditional global alignment of two sequences (Durbin et al., 1998), with a natural relationship between the logarithm of the probability of a path in the HMM and the score of an alignment according to traditional schema. More complex pair HMMs can represent pairwise alignments that incorporate more flexibility in the models of the lengths and conservation levels of different parts of the alignment.

Pair HMMs differ in an essential way from the multiple output HMMs introduced in Section 1.5.1: those have an alignment of the output sequences fixed, and in each step generate a character in each output sequence. If the alignment contains a gap, they generate a special character, for example a dash. On the other hand, the output sequences of pair HMMs do not identify which pairs of characters were emitted in the same step; when we decode a pair HMM, the goal is to *discover* such homologies.

The program SLAM, by Alexandersson et al. (2003), predicts genes simultaneously in two homologous genomic sequences, under the assumption that they have the same exon structure. Their pair HMM has separate states for exons, introns, signals and intergenic regions, as in HMMs for gene finding. Each state emits pairs

of sequences with conservation patterns typical for the sequence feature represented by the state, but can also allow for insertions or deletions, where a position in one sequence is not matched in the other. DoubleScan, by Meyer and Durbin (2002), is similar, but can also predict genes with different exon-intron structure. GeneWise, by Birney et al. (2004), uses pair HMMs to align a protein sequence to a genomic sequence. The non-coding states emit characters only in the genomic sequence, while coding states emit a triplet of nucleotides in the genomic sequence, and a single amino acid in the protein sequence.

The main disadvantage of pair HMMs is their high running time. Given two sequences generated by a pair HMM, we do not know which pairs of characters from these two sequences were generated at the same time; indeed, this is what decoding is to discover. The modified Viterbi algorithm that finds the most probable alignment of two sequences, and their annotations, is equivalent to an extension of classic global alignment algorithms, and as for those algorithms, its runtime is proportional to the product of the sequence lengths. Although such a running time is infeasible in many situations, different heuristics can be used to make the pair HMM approach more practical (Alexandersson et al., 2003; Meyer and Durbin, 2002). This approach is also hard to extend to multiple sources of information because its running time grows exponentially with the number of sequences, again as is true for classical algorithms for multiple alignment.

1.6 TRAINING THE PARAMETERS OF AN HMM

In the previous sections, we considered the simplest scenario of HMM parameter estimation: maximum likelihood training in an HMM without the multiple paths problem, on a completely annotated training set. This method is applied if we can determine the target state path for each sequence in the training set. In this case, it is sufficient to count the frequency of each transition and emission to estimate the model parameters that maximize the likelihood of the training data. Unfortunately, HMM training is not always so simple.

In this section, we explore several other scenarios for HMM training. First, when only unannotated or partially annotated sequences are available, we need to use unsupervised or semi-supervised training to estimate the parameters of the model. Second, often a single parameter set does not capture properties of all query sequences well, and we may want to adapt the parameter set to the query sequence before making a prediction. Finally, we may choose to use different optimization criteria instead of maximum likelihood principle.

1.6.1 Unsupervised and semi-supervised training

Supervised learning can be applied only when the annotation is known for each sequence in the training set, and there is a one-to-one correspondence between such an annotation and the state paths in the HMM. If this is not the case, we need to apply more complex methods for training. The task is, as in the supervised case, to find the

parameters of the HMM with a given topology that maximize the likelihood of the training set.

There is no general exact algorithm known for solving this unsupervised training problem efficiently; some modifications have even been shown to be NP-hard (Abe and Warmuth, 1992; Gillman and Sipser, 1994). The method most commonly used, the Baum-Welch algorithm (Baum, 1972), is an iterative heuristic and can be considered a special case of the general EM algorithm for learning maximum likelihood models from incomplete data (Dempster et al., 1977).

The Baum-Welch algorithm starts from an initial set of model parameters θ_0. In each iteration, it changes the parameters as follows:

1. Calculate the expected number of times each transition and emission is used to generate the training set T in an HMM whose parameters are θ_k.

2. Use the frequencies obtained in step 1 to re-estimate the parameters of the model, resulting in a new set of parameters, θ_{k+1}.

The first step of the algorithm can be viewed as creating a new *annotated* training set $T^{(k)}$, where for each *unannotated* sequence $X \in T$, we add every possible pair (X, H) of the sequence X and any state path, weighted by the conditional probability $\Pr(H|\, X, \theta_k)$ of the path H in the model with parameters θ_k, given the sequence X. The second step then estimates new parameters θ_{k+1}, as in the supervised scenario based on the new training set $T^{(k)}$. The Baum-Welch algorithm achieves the same result in $O(nm^2)$ time per iteration using the forward and backward algorithms to avoid explicitly creating this exponentially large training set. Details can be found, for example, in Durbin et al. (1998, Chapter 3.3).

Baum (1972) has shown that the likelihood of the training set improves (or stays the same) in each iteration of this algorithm. However, this does not guarantee that the Baum-Welch algorithm reaches optimal model parameters: it may instead reach a local maximum or a saddle point in the parameter space (Dempster et al., 1977).

A modification of the Baum-Welch algorithm, called *Viterbi training*, is often also used in practice. In the first step of the algorithm, instead of considering all possible paths through the model, we only consider the most probable path. However, this algorithm is not guaranteed to increase the likelihood of the observed data in each step (Durbin et al., 1998, Chapter 3.3).

The Baum-Welch algorithm can also be used in the semi-supervised scenario. For example, Krogh et al. (2001) train a transmembrane topology predictor on a data set where the exact boundaries of transmembrane helices are not known. Therefore, they allow the boundary to occur anywhere within a short window of the sequence. We can modify step 1 of the algorithm to include only paths that agree with such partial annotations.

1.6.2 Adjusting models to query sequences

Supervised and semi-supervised training assume that the training and testing sets contain samples independently generated from the same underlying distribution of

sequences and their annotations. In some situations such an assumption is not appropriate.

For example, Tusnády and Simon (1998) argue that the amino-acid composition of transmembrane helices cannot be adequately described by the same set of emission probabilities for all transmembrane proteins. Instead they propose to segment a given protein so that the difference in distribution between helix and non-helix regions is maximized. This is essentially achieved by optimizing the HMM emission probabilities with respect to the query sequence using unsupervised training. We can train the parameters not only on the single query sequence, but also on its homologs, assuming that they represent independent samples, generated by the same HMM. In this way we can use the information from homologous sequences without constructing multiple sequence alignment and without assuming that the annotation is the same in all sequences. Tusnády and Simon (1998) use emission parameters estimated on an annotated training set as pseudocounts in each step of the Baum-Welch algorithm.

Chatterji and Pachter (2005) use a similar approach to find genes in multiple homologous genomic regions by biasing parameters of a typical HMM gene finder specifically to match the genes on the input. The parameters of the model and gene predictions are iteratively improved by Gibbs sampling. Thus, after each iteration, gene predictions in all input sequences will tend to be more similar to each other, and the parameters of the model will fit the input sequences more closely.

We may also need to adjust parameters of a gene finder when applying it to a newly sequenced genome. In such a case we rarely have sufficiently large training set of manually annotated sequences. One approach is to identify easy to find genes, such as those with a strong protein match in a database and train the HMM using those genes (Larsen and Krogh, 2003). Korf (2004) has considered adjusting parameters trained on a different species by Viterbi training on the new species. Lomsadze et al. (2005) have shown that a careful procedure can obtain parameters of a eukaryotic gene finder on a new species in a completely unsupervised fashion, starting with a very simple set of manually created parameters.

1.6.3 Beyond maximum likelihood

So far, we considered algorithms that trained HMM parameters by maximizing the likelihood of the training set. A common criticism of the maximum likelihood (ML) approach in the machine learning literature is that it maximizes the wrong objective (see for example Krogh (1997)). Our goal in decoding is to retrieve the annotation H that maximizes $\Pr(H|X)$, where the sequence X is fixed. Therefore, instead of maximizing the joint probability $\Pr(H, X)$ of the training sequences, this perspective argues that we should concentrate on maximizing the conditional probability $\Pr(H|X)$, since the sequence X is fixed in the decoding phase, and it does not matter whether its probability is low or high. This optimization criterion is known as *conditional maximum likelihood* (CML).

In context of hidden Markov models, CML was used in applications in bioinformatics (Krogh, 1997) and natural language processing (Klein and Manning, 2002). Even if the sequences are annotated, there is no known closed formula or EM

algorithm that would estimate the parameters of the model to optimize the conditional maximum likelihood. Instead, numerical gradient descent methods are used to achieve local maximum. In these studies, slight (Klein and Manning, 2002) to significant (Krogh, 1997) improvement was observed compared to models trained by ML.

A theoretical analysis is available in the context of the simpler data classification problem, where a similar dichotomy occurs between the naive Bayes classifier (which is equivalent to ML) and logistic regression (equivalent to CML). In this context, Ng and Jordan (2002) have shown that even though using CML gives asymptotically lower error, ML requires significantly fewer training samples to converge to the best model: it requires only a logarithmic number of samples with respect to the number of parameters, compared to the linear number of samples required for convergence in CML. Thus ML training is appropriate if only a small number of samples is available, while it is better to use CML when the training set is large. It is not known whether these results extend to the case of more complex models, such as HMMs, where we are doing more than merely classifying a sample into categories. We may also ask (and no known answer exists to this question) whether the better response to an increase in training data is to switch from ML to CML, or to switch to a more accurate model of reality which requires a larger number of parameters.

One major disadvantage of HMMs optimized for CML is that it is hard to interpret their emission and transition probabilities. The generative process associated with the HMM no longer generates sequences that look like sequences from the training set. The probabilities no longer represent frequencies observed directly in the sequence, which makes it hard to incorporate prior knowledge about the problem into the probabilistic model by applying restrictions on parameters of the model, or by creating a custom model topology.

For example, the HMM modeling the topology of transmembrane proteins in Figure 1.4 has two states representing transmembrane helices. It may be reasonable to assume that since the sequences corresponding to these two states serve the same function (membrane transition), that in an ML model, both states should share the same emission probabilities. Based on this assumption, we can reduce the number of parameters (and thus the number of sequences required for training) by *tying* those parameters together, forcing them to be the same. On the other hand, since in CML method the emission probabilities are set to maximize the conditional probability of the annotation given the sequence, rather than likelihood of the sequence, it is not clear that the emission probabilities in these two states should be similar, even if the sequences attributed to these states are similar.

Conditional random fields (Lafferty et al., 2001) further continue in the direction of CML training, abandoning the probabilistic interpretation of emission and transition probabilities, replacing them with undirected potentials that do not need to be normalized to 1. They were applied in bioinformatics for recognizing protein structure motifs (Liu et al., 2006) and for finding genes (Culotta et al., 2005).

Some recent extensions abolish the probabilistic interpretation of HMMs altogether. Instead, they consider the following problem directly: set the parameters of the model (without normalization restrictions) so that the model discriminates well

between correct and incorrect annotations. These models, such as hidden Markov support vector machines (Altun et al., 2003) and convex hidden Markov models (Xu et al., 2006b), are inspired by maximum margin training and kernel methods in support vector machines (Boser et al., 1992), which are a very successful method for the classification problem.

1.7 CONCLUSION

On our tour through HMMs and their use in biological sequence annotation, we have seen both the most traditional HMM algorithms and their most exotic extensions. We have seen extensions to the decoding algorithms to handle many cases where multiple different paths through the HMM correspond to the same semantic meaning, and algorithms to handle generalized HMMs, in which the lengths of features may come from complex, non-geometric, distributions. We have seen many ways in which HMMs can operate on multiple sequences, and in all these cases we have argued why these extensions are useful in modeling and annotating biological sequences.

Many of these extensions rely upon the conceptual simplicity of the basic HMM framework: unlike the parameters of a neural network or of a support vector machine, the parameters of a hidden Markov model trained for maximum likelihood are extremely simple to understand. Even for their more complex extensions (such as phylogenetic HMMs or pair HMMs), one can quickly determine the semantic meaning of the parameters, and imagine ways to make them estimated more accurately, or to change the topology of the HMM to more closely model reality (though, of course, our discussion of the multiple path problem in Section 1.3.2 shows that this may not be entirely wise). Even the more complex decoding approaches to handle external information, such as those of Section 1.5.2 can be seen as a way of mathematically encoding sensible intuitive concepts.

Perhaps the most important question for the future of HMMs, then, is whether increasingly sophisticated HMM modeling, training, and decoding procedures can continue to maintain this conceptual simplicity while still allowing the use of ever more and more complex forms of sequence data. Can we incorporate a useful understanding of the 3-dimensional geometry of molecules into HMM analysis? Can we usefully train HMMs to understand the evolutionary relationships among thousands of sequences? Can we annotate features and subfeatures of biological sequences that are allowed to overlap each other in complex ways, and where a feature is not simply a contiguous segment of DNA? These questions, and numerous others, will be the subject of the research of the next many years in HMM analysis.

Acknowledgments

The work of all three authors has been supported by the Natural Sciences and Engineering Research Council of Canada. We are grateful to many colleagues for

interesting discussions over the past several years, including Ming Li, Burkhard Morgenstern, Dale Schuurmans, Gregory Kucherov, Ian Korf, and Dan Gusfield.

REFERENCES

Abe, N. and Warmuth, M. K. (1992). On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9(2-3):205–260.

Alexandersson, M., Cawley, S., and Pachter, L. (2003). SLAM: cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Research*, 13(3):496–502.

Allen, J. E. and Salzberg, S. L. (2006). A phylogenetic generalized hidden Markov model for predicting alternatively spliced exons. *Algorithms for Molecular Biology*, 1(1):14.

Altun, Y., Tsochantaridis, I., and Hofmann, T. (2003). Hidden Markov support vector machines. In *ICML 2003: 20th International Conference on Machine Learning*, pages 3–10. AAAI Press.

Asmussen, S., Nerman, O., and Olsson, M. (1996). Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23(4):419–441.

Aydin, Z., Altunbasak, Y., and Borodovsky, M. (2006). Protein secondary structure prediction for a single-sequence using hidden semi-Markov models. *BMC Bioinformatics*, 7:178.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities III, Proceeding of the Third Symposium*, pages 1–8. Academic Press, New York.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73:360–363.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., and Wheeler, D. L. (2000). GenBank. *Nucleic Acids Research*, 28(1):15–18.

Birney, E., Clamp, M., and Durbin, R. (2004). GeneWise and Genomewise. *Genome Research*, 14(5):988–995.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT 1992: 5th Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press.

Brejová, B., Brown, D. G., Li, M., and Vinař, T. (2005). Exonhunter: A comprehensive approach to gene finding. *Bioinformatics*, 21(S1):i57–i65.

Brejová, B., Brown, D. G., and Vinař , T. (2004). The most probable labeling problem in HMMs and its applications to bio informatics. In *WABI 2004: Algorithms in Bioinformatics*, volume 3240 of *Lecture Notes in Bioinformatics*, pages 426–437, Bergen, Norway. Springer.

Brejová, B. and Vinař, T. (2002). A better method for length distribution modeling in HMMs and its application to gene finding. In *CPM 2002: Combinatorial Pattern Matching, 13th Annual Symposium*, volume 2373 of *Lecture Notes in Computer Science*, pages 190–202, Fukuoka, Japan. Springer.

Burge, C. and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94.

Bystroff, C., Thorsson, V., and Baker, D. (2000). HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301(1):173–180.

Cawley, S. L. and Pachter, L. (2003). HMM sampling and applications to gene finding and alternative splicing. *Bioinformatics*, 19(S2):II36–II41.

Chatterji, S. and Pachter, L. (2005). Large multiple organism gene finding by collapsed Gibbs sampling. *Journal of Computational Biology*, 12(6):599–608.

Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd conference on Applied natural language processing*, pages 136–143, Morristown, NJ, USA. Association for Computational Linguistics.

Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51(1):79–94.

Commault, C. and Mocanu, S. (2003). Phase-type distributions and representations: some resuts and open problems for system theory. *International Journal of Control*, 76(6):566–580.

Culotta, A., Kulp, D., and McCallum, A. (2005). Gene prediction with conditional random fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst.

Delorenzi, M. and Speed, T. (2002). An HMM model for coiled-coil domains and a comparison with PSSM-based predictions. *Bioinformatics*, 18(4):617–625.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statististical Society Series B*, 39(1):1–38.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Fariselli, P., Martelli, P. L., and Casadio, R. (2005). A new decoding algorithm for hidden Markov models improves the prediction of the topology of all-beta membrane proteins. *BMC Bioinformatics*, 6(S4):S12.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376.

Felsenstein, J. and Churchill, G. A. (1996). A hidden Markov model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution*, 13(1):93–104.

Finn, R. D., Mistry, J., Schuster-Bockler, B., Griffiths-Jones, S., Hollich, V., Lassmann, T., Moxon, S., Marshall, M., Khanna, A., Durbin, R., Eddy, S. R., Sonnhammer, E. L., and Bateman, A. (2006). Pfam: clans, web tools and services. *Nucleic Acids Research*, 34(Database issue):D247–251.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278.

Gillman, D. and Sipser, M. (1994). Inference and minimization of hidden Markov chains. In *COLT 1994: Proceedings of the 7th Annual Conference on Computational Learning Theory*, pages 147–158. ACM Press.

Gilson, P. R., Nebl, T., Vukcevic, D., Moritz, R. L., Sargeant, T., Speed, T. P., Schofield, L., and Crabb, B. S. (2006). Identification and stoichiometry of glycosylphosphatidylinositol-anchored membrane proteins of the human malaria parasite Plasmodium falciparum. *Molecular and Cellular Proteomics*, 5(7):1286–1289.

Goldman, N., Thorne, J. L., and Jones, D. T. (1996). Using evolutionary trees in protein secondary structure prediction and other comparative sequence analyses. *Journal of Molecular Biology*, 263(2):196–208.

Gross, S. S. and Brent, M. R. (2005). Using multiple alignments to improve gene prediction. In *RECOMB 2005: Proceedings of the 9th Annual International Conference on Research in Computational Molecular Biology*, volume 3500 of *Lecture Notes in Computer Science*, pages 374–388. Springer.

Johnson, M. T. (2005). Capacity and complexity of HMM duration modeling techniques. *IEEE Signal Processing Letters*, 12(5):407–410.

Jukes, T. H. and Cantor, C. (1969). *Evolution of protein molecules*, pages 21–132. Academic Press.

Käll, L., Krogh, A., and Sonnhammer, E. L. L. (2005). An HMM posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, 21(S1):i251–257.

Klein, D. and Manning, C. D. (2002). Conditional structure versus conditional estimation in NLP models. In *EMNLP 2002: Conference on Empirical Methods in Natural Language Processing*, pages 9–16. Association for Computational Linguistics.

Korf, I. (2004). Gene finding in novel genomes. *BMC Bioinformatics*, 5:59.

Korf, I., Flicek, P., Duan, D., and Brent, M. R. (2001). Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17(S1):S140–148. ISMB 2001.

Krogh, A. (1997). Two methods for improving performance of an HMM and their application for gene finding. In *ISMB 1997: Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, pages 179–186.

Krogh, A. (2000). Using database matches with for HMMGene for automated gene detection in Drosophila. *Genome Research*, 10(4):523–528.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology. applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1501.

Krogh, A., Larsson, B., von Heijne, G., and Sonnhammer, E. L. (2001). Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *Journal of Molecular Biology*, 305(3):567–570.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001: 18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Larsen, T. S. and Krogh, A. (2003). EasyGene–a prokaryotic gene finder that ranks ORFs by statistical significance. *BMC Bioinformatics*, 4:21.

Liu, Y., Carbonell, J., Weigele, P., and Gopalakrishnan, V. (2006). Protein fold recognition using segmentation conditional random fields (SCRFs). *Journal of Computational Biology*, 13(2):394–406.

Lomsadze, A., Ter-Hovhannisyan, V., Chernoff, Y. O., and Borodovsky, M. (2005). Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, 33(20):6494–6496.

Martelli, P. L., Fariselli, P., Krogh, A., and Casadio, R. (2002). A sequence-profile-based HMM for predicting and discriminating beta barrel membrane proteins. *Bioinformatics*, 18(S1):S46–53.

McAuliffe, J. D., Pachter, L., and Jordan, M. I. (2004). Multiple-sequence functional annotation and the generalized hidden Markov phylogeny. *Bioinformatics*, 20(12):1850–1850.

McCauley, S. and Hein, J. (2006). Using hidden Markov models and observed evolution to annotate viral genomes. *Bioinformatics*, 22(11):1308–1316.

Meyer, I. M. and Durbin, R. (2002). Comparative ab initio prediction of gene structures using pair HMMs. *Bioinformatics*, 18(10):1309–1318.

Mitchell, C., Harper, M., and Jamieson, L. (1995). On the complexity of explicit duration HMMs. *IEEE Transactions on Speech and Audio Processing*, 3(3):213–217.

Müller, H. (2001). New approach to fire detection algorithms based on the hidden Markov model. In *AUBE 2001: 12th International Conference on Automatic Fire Detection*, pages 129–138. National Institute of Standards and Technology.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS 2002: Advances in Neural Information Processing Systems*, pages 841–848. MIT Press.

Nielsen, H. and Krogh, A. (1998). Prediction of signal peptides and signal anchors by a hidden Markov model. In *ISMB 1998: Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology*, pages 122–130. AAAI Press.

Ohler, U., Niemann, H., and Rubin, G. M. (2001). Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. *Bioinformatics*, 17(S1):S199–206.

Pavlović, V., Garg, A., and Kasif, S. (2002). A Bayesian framework for combining gene predictions. *Bioinformatics*, 18(1):19–27.

Pedersen, J. S. and Hein, J. (2003). Gene finding with a hidden Markov model of genome structure and evolution. *Bioinformatics*, 19(2):219–227.

Pollastri, E. and Simoncelli, G. (2001). Classification of melodies by composer with hidden Markov models. In *WEDELMUSIC 2001: The 1st International Conference on WEB Delivering of Music*, pages 88–95. IEEE Computer Society.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Rastas, P., Koivisto, M., Mannila, H., and Ukkonen, E. (2005). A hidden Markov technique for haplotype reconstruction. In *WABI 2005: Algorithms in Bioinformatics*, volume 3692 of *Lecture Notes in Computer Science*, pages 140–151. Springer.

Rodriguez, F., Oliver, J. L., Marin, A., and Medina, J. R. (1990). The general stochastic model of nucleotide substitution. *Journal of Theoretical Biology*, 142(4):485–501.

Schultz, A.-K., Zhang, M., Leitner, T., Kuiken, C., Korber, B., Morgenstern, B., and Stanke, M. (2006). A jumping profile hidden Markov model and applications to recombination sites in HIV and HCV genomes. *BMC Bioinformatics*, 7:265.

Schwartz, R. and Chow, Y.-L. (1990). The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses. In *ICASSP 1990: Acoustics, Speech, and Signal Processing*, pages 81–84, vol. 1.

Seymore, K., McCallum, A., and Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*.

Siepel, A., Bejerano, G., Pedersen, J. S., Hinrichs, A. S., Hou, M., Rosenbloom, K., Clawson, H., Spieth, J., Hillier, L. W., Richards, S., Weinstock, G. M., Wilson, R. K., Gibbs, R. A., Kent, W. J., Miller, W., and Haussler, D. (2005). Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Research*, 15(8):1034–1040.

Siepel, A. and Haussler, D. (2003). Combining phylogenetic and hidden Markov models in biosequence analysis. In *RECOMB '03: Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*, pages 277–286, New York, NY, USA. ACM Press.

Siepel, A. and Haussler, D. (2004). Computational identification of evolutionarily conserved exons. In *RECOMB 2004: Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology*, pages 177–186. ACM Press.

Stanke, M., Keller, O., Gunduz, I., Hayes, A., Waack, S., and Morgenstern, B. (2006a). AUGUSTUS: ab initio prediction of alternative transcripts. *Nucleic Acids Research*, 34(W):W435–439.

Stanke, M., Schoffmann, O., Morgenstern, B., and Waack, S. (2006b). Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, 7:62.

Stanke, M. and Waack, S. (2003). Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19(S2):II215–II225.

Tusnády, G. E. and Simon, I. (1998). Principles governing amino acid composition of integral membrane proteins: application to topology prediction. *Journal of Molecular Biology*, 283(2):489–506.

Viklund, H. and Elofsson, A. (2004). Best alpha-helical transmembrane protein topology predictions are achieved using hidden Markov models and evolutionary information. *Protein Science*, 13(7):1908–1917.

Vinař, T. (2005). *Enhancements to Hidden Markov Models for Gene Finding and Other Biological Applications*. PhD thesis, University of Waterloo.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–267.

Xu, E. W., Kearney, P., and Brown, D. G. (2006a). The use of functional domains to improve transmembrane protein topology prediction. *Journal of Bioinformatics and Computational Biology*, 4(1):109–113.

Xu, L., Wilkinson, D., Southey, F., and Schuurmans, D. (2006b). Discriminative unsupervised learning of structured predictors. In *ICML 2006: International Conference on Machine Learning*.

Yamron, J., Carp, I., Gillick, L., Lowe, S., and van Mulbregt, P. (1998). A hidden Markov model approach to text segmentation and event tracking. In *ICASSP 1998: IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 333–336.

Yang, Z. (1995). A space-time process model for the evolution of DNA sequences. *Genetics*, 139(2):993–1005.

Yeh, R. F., Lim, L. P., and Burge, C. B. (2001). Computational inference of homologous gene structures in the human genome. *Genome Research*, 11(5):803–806.

Zhu, J., Liu, J. S., and Lawrence, C. E. (1998). Bayesian adaptive sequence alignment algorithms. *Bioinformatics*, 14(1):25–39.