What is a hidden Markov model?

Sean R. Eddy
Howard Hughes Medical Institute & Department of Genetics,
Washington University School of Medicine
4444 Forest Park Blvd., Box 8510
Saint Louis, Missouri 63108 USA
eddy@genetics.wustl.edu

August 31, 2004

Often, problems in biological sequence analysis are just a matter of putting the right label on each residue. In gene identification, we want to label nucleotides as exons, introns, or intergenic sequence. In sequence alignment, we want to associate residues in a query sequence with homologous residues in a target database sequence. We can always write an *ad hoc* program for any given problem, but the same potentially frustrating issues will always recur. One issue is that we often want to incorporate multiple heterogenous sources of information. A genefinder, for instance, ought to combine splice site consenses, codon bias, exon/intron length preferences, and open reading frame analysis all in one scoring system. How should all those parameters be set? How should different kinds of information be weighted? A second issue is being able to interpret results probabilistically. Finding a best scoring answer is one thing, but what does the score mean, and how confident are we that the best answer, or any given part of it, is correct? A third issue is extensibility. The moment we perfect our *ad hoc* genefinder, we wish we had also modeled translational initiation consensus, alternative splicing, and a polyadenylation signal. All too often, piling more reality onto a fragile *ad hoc* program makes it collapse under its own weight.

Hidden Markov models (HMMs) are a formal foundation for making probabilistic models of linear sequence "labeling" problems [1, 2]. They provide a conceptual toolkit that allows building a model of almost any complexity, just by drawing an intuitive picture. They are at the heart of a diverse range of programs, including genefinding, consensus profile searches, multiple sequence alignment, and regulatory site identification. HMMs are the Legos of computational sequence analysis.

A toy HMM: 5' splice site recognition.

As a simple example HMM, imagine the following caricature of a 5' splice site recognition problem. Assume we are given a DNA sequence that begins in an exon, contains one and only one 5' splice site, and ends in an intron. The problem is to identify where the switch from exon to intron occurred – where the 5' splice site (5'SS) is.

For us to be able to guess intelligently, the sequences of exons, splice sites, and introns must have different statistical properties. Let's imagine some simple statistical differences: a base composition difference between introns and exons, and the presence of a single consensus G at the 5' splice site. Let's say exons have a uniform base composition on average (25% each base), introns are A/T rich (say, 40% each for A/T, 10% each for C/G), and the 5'SS consensus nucleotide is almost always a G (say, 95% G and 5% A).

This information is sufficient to draw a hidden Markov model, as shown in Figure 1. The HMM invokes three *states*, one for each of the three labels we might assign to a nucleotide: E (exon), 5 (5' splice site), and I (intron). Each state (shown as circles) has its own *emission probabilities* (shown above the states), stating the probability that it generates an A,C,G, or T. The emission probabilities describe the statistical features of the DNA residues: the base composition of exons and introns, and the consensus G at the 5'SS. Each state also has *transition probabilities* (shown as arrows), stating the probability of moving from this state to a new state. The transition probabilities describe the linear order that we expect the states to occur in: one or more E's, one 5, one or more I's.

So, what's hidden?

It's useful to imagine an HMM *generating* a sequence. When we visit a state, we emit a residue according to the state's emission probability distribution. Then, we choose which state to visit next according to the state's transition probability distribution. The model thus generates *two* strings of information. One is the underlying *state path* (the labels), as we transition from state to state. The other is the *observed sequence* (the DNA), each residue of which is emitted from one state in the state path.

The state path is a Markov chain, which means that what state we go to next depends only on what state we're in. But, if we're only given the observed sequence, this underlying state path is *hidden* - these are the residue labels that we'd like to infer. The state path is a *hidden Markov chain*.

The probability $P(S, \pi \mid \text{HMM}, \theta)$ that an HMM with parameters θ generates a state path π and an observed sequence S is the product of all the emission probabilities and transition probabilities that were used. For example, consider the 26 nucleotide sequence and state path in the middle of Figure 1, where there are 27 transitions and 26 emissions to tote up; multiply all 53 probabilities together (and take the log, since these are small numbers) and you'll calculate $\log_e P(S, \pi \mid \text{HMM}, \theta) = -41.22$.

An HMM is a *full probabilistic model* – the model parameters θ and the overall sequence "scores" $P(S, \pi \mid \text{HMM}, \theta)$ are all probabilities. As a result, we can use standard Bayesian probability theory to manipulate these numbers in powerful ways, including optimizing parameters, calculating confidence in predictions, and interpreting the statistical significance of scores.

Finding the best state path

In a sequence analysis problem, we're only given the observed sequence, and we want to infer the hidden state path. There are potentially many state paths that could generate the same sequence. We want to find the one with the highest probability.

For example, if we were given the HMM and the 26 nucleotide sequence in Figure 1, there are only 14 possible paths that have non-zero probability; the 5'SS can only fall on one of the 14 internal A's or G's in the sequence. Figure 1 enumerates the six highest-scoring paths (those with G at the 5'SS) and their log probabilities. The best one has a log probability of -41.22, which places the most likely 5'SS position at the fifth G in the sequence.

For most real problems, there are so many possible state sequences that we could not afford to enumerate them. An efficient algorithm called the Viterbi algorithm is guaranteed to find the most probable state path given a sequence and an HMM. The Viterbi algorithm is a dynamic programming algorithm quite similar to those used for standard sequence alignment.

Beyond best scoring alignments

Figure 1 shows that one alternative state path differs only slightly in score from putting the 5'SS at the fifth G (log probabilities of -41.71 vs. -41.22). How confident are we that the fifth G is the right choice?

This is just one example of an advantage of probabilistic modeling. Because we have a probabilistic model, we can calculate our confidence directly. The probability that residue i was emitted by state k is the sum of the probabilities of all the state paths that use state k to generate residue i (that is, $\pi_i = k$ in the state path π), normalized by the unconditional sum over all possible state paths. In our toy model, this is just one state path in the numerator and a sum over 14 state paths in the denominator. We get a probability of 46% that the best-scoring fifth G is the correct guess, and the probability that the sixth G position is correct is 28% (bottom of Figure 1). This is called *posterior decoding*. For larger problems, posterior decoding uses two more dynamic programming algorithms called the Forward and Backward algorithms, which are essentially like Viterbi, but they sum over possible paths instead of choosing the best.

Making more realistic models

Creating an HMM just means specifying four things: 1) the symbol alphabet, K different symbols (e.g. ACGT, $K = 4$); 2) the number of states in the model, M; 3) emission probabilities $e_i(x)$ for each state i, that sum to one over K symbols x, $\sum_x e_i(x) = 1$; and 4) transition probabilities $t_i(j)$ for each state i going to any other state j (including itself), that sum to one over the M states j ($\sum_j t_{ij} = 1$). *Any* model that has these properties is a hidden Markov model.

Essentially, this means that one can make a new HMM just by drawing a picture corresponding to the problem at hand, like Figure 1. This graphical simplicity lets one focus clearly on the biological definition of a problem.

For example, in our toy splice site model, maybe we're not happy with our discrimination power; maybe we decide to model a more realistic six-nucleotide consensus GTRAGT at the 5' splice site. We can just put a row of six HMM states in place of “5” state, as a model of an ungapped consensus motif of length 6, and parameterize the emission probabilities on known 5’ splice sites. Then maybe we want to make a more complete intron model, including a 3’ splice site; we just add a row of states for the 3’SS consensus, and do something to let the observed sequence end in an exon instead of an intron (such as, add a second E state at the end, after the I state and the 3’ SS). Then maybe we want to build a complete gene model... it’s just a matter of drawing what we’re trying to model.

The catch

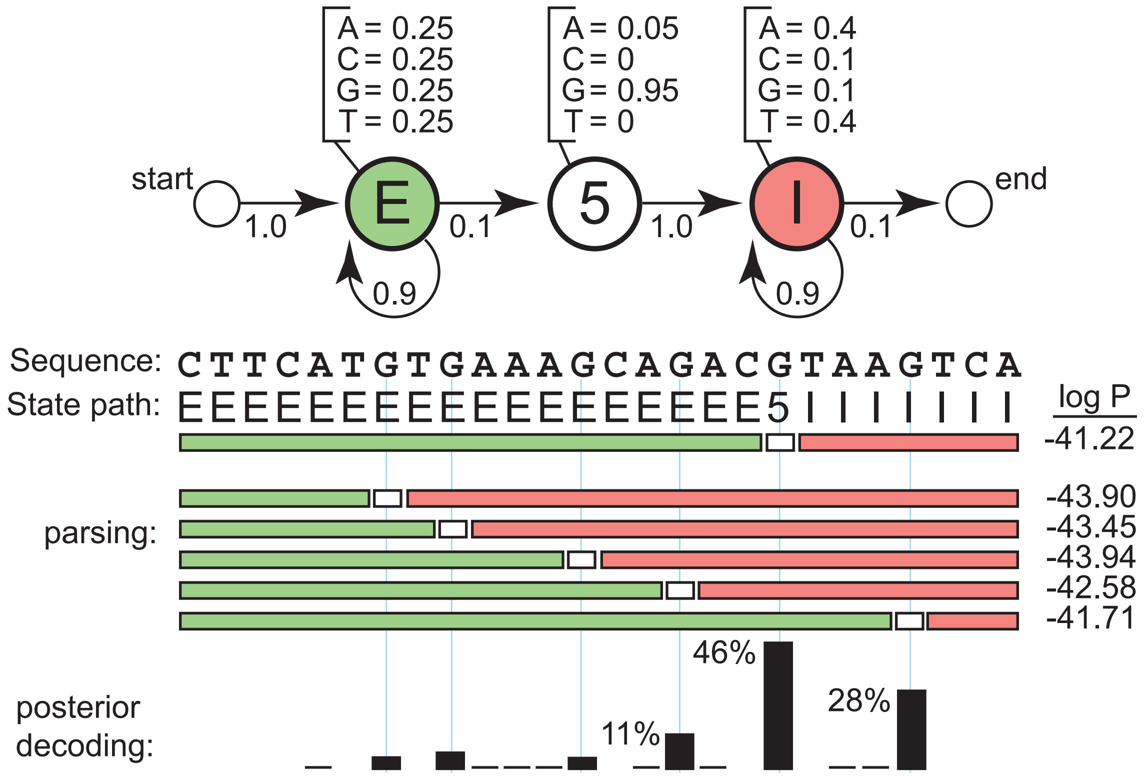
HMMs are reasonable models of linear sequence problems, but they don’t deal well with correlations between residues or states, especially long-range correlations. HMMs assume that each residue depends only on one underlying state, and each state in the state path depends only on one previous state; otherwise, residues and states are independent of each other. One example where HMMs are usually inappropriate is RNA secondary structure analysis. Conserved RNA base pairs induce pairwise correlations between distant residues (one position might be anything, but whatever it is, the base-paired partner must be complementary). The state path of an HMM has no way of “remembering” what a distant state generated when a second state generates its residue.

In some cases, one can bend the rules of HMMs and deal with short-range correlations without breaking the algorithms. For instance, in genefinding, it’s much better to model codons as a correlated triplet, instead of three independent residues. It’s no big deal to make a state that generates a triplet from an emission distribution over 64 possible codons. Trickery like this, especially common in genefinding applications, extends the basic toolkit of HMM-based techniques to even more powerful classes of probabilistic models called “generalized HMMs” or “semi-HMMs”.

References

[1] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[2] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK, 1998.


A = 0.25
C = 0.25
G = 0.25
T = 0.25
A = 0.05
C = 0
G = 0.95
T = 0
A = 0.4
C = 0.1
G = 0.1
T = 0.4
start
1.0
E
0.1
5
1.0
I
0.1
end
0.9
0.9
Sequence: CTTCATGTGAAAGCAGACGTAAGTCA
State path: EEEEEEEEEEEEEEEEEE5IIIIIII
log P
-41.22
parsing:
-43.90
-43.45
-43.94
-42.58
-41.71
posterior decoding:
46%
28%
11%