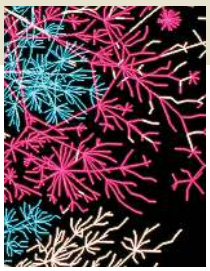

What is a hidden Markov model?

Sean R Eddy

Statistical models called hidden Markov models are a recurring theme in computational biology. What are hidden Markov models, and why are they so useful for so many different problems?

Often, biological sequence analysis is 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 frustrating issues will always recur. One is that we want to incorporate heterogeneous sources of information. A genefinder, for instance, ought to combine splice-site consensus, codon bias, exon/intron length preferences and open reading frame analysis into one scoring system. How should these parameters be set? How should different kinds of information be weighted? A second issue is 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 scoring answer 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. 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 for building complex models just by drawing an intuitive picture. They are at the heart of a diverse range of programs, including genefinding, profile searches, multiple sequence alignment and regulatory site identification. HMMs are the Legos of computational sequence analysis.

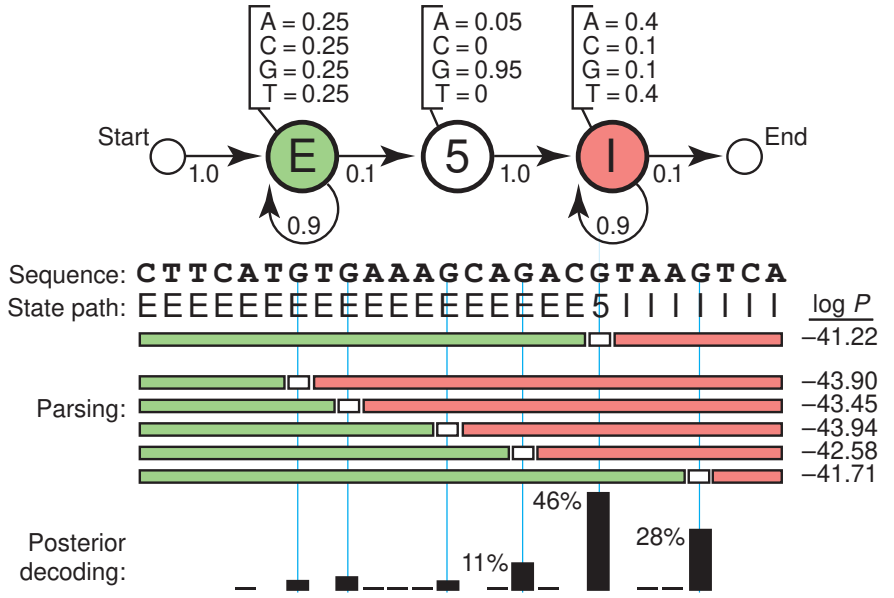

Figure 1 A toy HMM for 5′ splice site recognition. See text for explanation.

A toy HMM: 5′ splice site recognition

As a simple example, 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 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 guess intelligently, the sequences of exons, splice sites and introns must have different statistical properties. Let's imagine some simple differences: say that 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).

Starting from this information, we can draw an HMM (**Fig. 1**). The HMM invokes three *states*, one for each of the three labels we might assign to a nucleotide: E (exon), 5 (5′SS) and I (intron). Each state has its own *emission probabilities* (shown above the states), which model the base composition of exons, introns and the consensus G at the 5′SS. Each state also has *transition probabilities* (arrows), the probabilities of moving from this state to a new state. The transition

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

probabilities describe the linear order in which we expect the states to occur: one or more Es, one 5, one or more Is.

So, what's hidden?

It's useful to imagine an HMM generating a sequence. When we visit a state, we emit a residue from 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 being emitted from one state in the state path.

The state path is a Markov chain, meaning that what state we go to next depends only on what state we're in. Since 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 $P(S,\pi|\text{HMM},\theta)$ 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 $\log P(S,\pi|\text{HMM},\theta) = -41.22$.

An HMM is a *full probabilistic model*—the model parameters and the overall sequence 'scores' are all probabilities. Therefore, we can use Bayesian probability theory to manipulate these numbers in standard, powerful ways, including optimizing parameters and interpreting the significance of scores.

Finding the best state path

In an analysis problem, we're given a 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 14 possible paths that have nonzero probability, since the 5′SS must fall on one of 14 internal As or Gs. **Figure 1** enumerates the six highest-scoring paths (those with G at the 5′SS). The best one has a log probability of –41.22, which infers that the most likely 5′SS position is at the fifth G.

For most problems, there are so many possible state sequences that we could not afford to enumerate them. The efficient 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 versus –41.22). How confident are we that the fifth G is the right choice?

This is an example of an advantage of probabilistic modeling: 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 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 correct and 28% that the sixth G position is correct (**Fig. 1**, bottom). This is called *posterior decoding*. For larger problems, posterior decoding uses two dynamic programming algorithms called Forward and Backward, which are essentially like Viterbi, but they sum over possible paths instead of choosing the best.

Making more realistic models

Making an HMM means specifying four things: (i) the symbol alphabet, K different symbols (e.g., ACGT, $K = 4$); (ii) the number of states in the model, M; (iii) emission probabilities $e_i(x)$ for each state i, that sum to one over K symbols x, $\sum_x e_i(x) = 1$; and (iv) 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, $\sum_j t_i(j) = 1$. Any model that has these properties is an HMM.

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 want to add a more realistic six-nucleotide consensus GTRAGT at the 5′ splice site. We can put a row of six HMM states in place of '5' state, to model a six-base ungapped consensus motif, parameterizing the emission probabilities on known 5′ splice sites. And maybe we want to model a complete intron, including a 3′ splice site; we just add a row of states for the 3′SS consensus, and add a 3′ exon state to let the observed sequence end in an exon instead of an intron. Then maybe we want to build a complete gene model…whatever we add, it's just a matter of drawing what we want.

The catch

HMMs don't deal well with correlations between residues, because they assume that each residue depends only on one underlying state. An example where HMMs are usually inappropriate is RNA secondary structure analysis. Conserved RNA base pairs induce long-range pairwise correlations; one position might be any residue, but the base-paired partner must be complementary. An HMM state path has no way of 'remembering' what a distant state generated.

Sometimes, one can bend the rules of HMMs without breaking the algorithms. For instance, in genefinding, one wants to emit a correlated triplet codon instead of three independent residues; HMM algorithms can readily be extended to triplet-emitting states. However, the basic HMM toolkit can only be stretched so far. Beyond HMMs, there are more powerful (though less efficient) classes of probabilistic models for sequence analysis.

1. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77**, 257–286 (1989).
2. Durbin, R., Eddy, S.R., Krogh, A. & Mitchison, G.J. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge University Press, Cambridge UK, 1998).