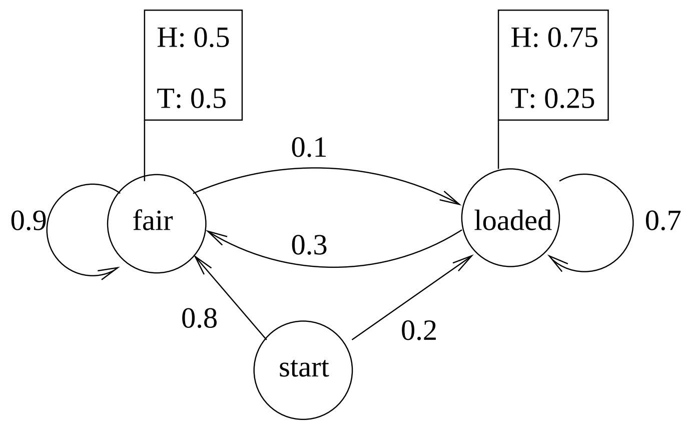
Hidden Markov Models

It is natural to combine the i.i.d. model (which has emission probabilities for the various symbols of the underlying alphabet) with a Markov chain (which has transition probabilities for each edge). A probabilistic sequence model with both emission and transition probabilities is called a *hidden Markov model* (HMM). For example, consider the following probabilistic model for generating a sequence of `H`'s ("heads") and `T`'s ("tails"). The person generating the sequence has two coins, one fair (where `H` and `T` have equal likelihood) and one loaded (which comes up `H` 75% of the time). Just after flipping the fair coin, the person picks the next coin to use, switching to the loaded coin 10% of the time. After a flip of the loaded coin, she changes back to the fair coin 30% of the time. To complete the model, let's assume that the process starts with the fair coin 80% of the time. An observer sees only the sequence of `H` and `T`; the underlying sequences of states (fair or loaded) is hidden. Here is the picture:



Given the picture it would be straightforward to write a computer program that randomly generates sequences according to the model. But how can we score a given sequence, i.e., determine the probability that the sequence would be generated by the model? The sticky point is that we can't say which sequence of states was followed to generate the observed sequence of `H` and `T`.

Motivation from the gene-prediction problem. We'll think of a genomic DNA sequence as generated by a probabilistic sequence model that has states for introns, for exons, for splice signals, for poly-A signals, etc. Any piece of DNA, such as `ACACAC`, could be generated by either intron states or exon states, though not with equal likelihood. Similarly, perfectly good poly-A signals can be generated in an exon or intron state. That is, given the generated sequence we cannot determine with certainty the underlying states. But estimating the sequence of states is precisely the gene-prediction problem. Given a DNA sequence to analyze, a natural goal is to find the most likely state path such that the model picked the path and generated the observed DNA sequence. For instance, let's intuitively try to "decode" the followng sequence, generated by the above head/tail HMM.

`TTHTHTTHTHTTHHTHTHHTHHHHHTHHTHHTTHTTHTHTTHTTTHTT`

It looks as if the fair coin was used near the ends of the sequence, given the density of `T`. But near the middle there is a high density of `H`, suggesting that the loaded coin was used in that region. One can imagine assigning to each letter in the sequence the probability that it was generated in the loaded state, just as GenScan reports a probability that a predicted exon was, indeed, generated from exon states in its underlying probabilistic sequence model. With the above head/tail sequence, the probability of being emitted in the loaded state looks highest at the middle `H` in the run of five consecutive `H`'s, but the precise points where coin-shifts were made aren't clear. In what follows, we'll see how determine the most probable state path for a given observed sequence. We'll also see how to efficiently score a given sequence, by in effect summing, over all state paths, the probability of picking that path and emitting the sequence.

A toy example. Consider the sequence HHT. It (or any other head/tail sequence of length 3) can be generated from any sequence of three states (not counting the start state). We'll use F and L to denote the fair and loaded states, respectively. The probability that the model picks the path FFF and generates HHT is $(0.8)(0.5)(0.9)(0.5)(0.9)(0.5) = 0.081$, while the probability that it picks the path LLL and generates HHT is $(0.2)(0.75)(0.7)(0.75)(0.7)(0.25) = 0.0137\cdots$. To get the probability that HHT is generated by the model, we need to sum, over all 8 paths of length 3, the probability of picking that path and generating HHT.

Let π denote an arbitrary path of length 3 in the "coin" HMM. Fix S as the observed sequence HHT. Let $P(\pi, S)$ denote the joint probability of π and S, i.e., the probability of picking π and generating S.

π	$P(\pi, S)$
FFF	0.081
FFL	0.0045
FLF	0.0045
FLL	0.00525
LFF	0.010125
LFL	$0.000562\cdots$
LLF	$0.011812\cdots$
LLL	$0.013781\cdots$
total	$0.131449\cdots$

Thus, the probability of S (given the model) is $P(S) = 0.131449\cdots$.

Given that the observed sequence HHT was generated, what is the "posterior" probability, $P(\pi|S)$, that a given state-path π was taken? We use the formula

$$P(\pi|S) = P(\pi, S)/P(S)$$

π	$P(\pi, S)$	$P(\pi|S)$
FFF	0.081	$0.6158\cdots$
FFL	0.0045	$0.0342\cdots$
FLF	0.0045	$0.0342\cdots$
FLL	0.00525	$0.0399\cdots$
LFF	0.010125	$0.0769\cdots$
LFL	$0.000562\cdots$	$0.0427\cdots$
LLF	$0.011812\cdots$	$0.0898\cdots$
LLL	$0.013781\cdots$	$0.1047\cdots$

Thus the most probable path is FFF.

What is the probability that the second H in HHT was generated in the F state?
$0.6158 + 0.0342 + 0.0769 + 0.0427 \approx 0.77$

Fundamental methods for HMMs. In what follows, consider a fixed hidden Markov model, $\mathcal{M}$, with start state s_0 and n other states $s_1, s_2, \ldots, s_n$. Let $t_{j,k}$ denote the *transition probability* of going from s_j to s_k, and let $e_j(a)$ be the *emission probability* of symbol a in state s_j. (State s_0 doesn't emit symbols.) Thus $\sum_{k=1}^{n} t_{j,k} = 1$ for $j = 0, 1, \ldots, n$, while if $j > 0$, then $\sum_a e_j(a) = 1$, summing over all possible observed symbols a. Also, fix an observed sequence $x = x_1 x_2 \ldots x_m$ consisting of m symbols.

Consider a path π from s_0 and of length m (the same length as x). That is, π is a connected chain of m edges in $\mathcal{M}$ that starts at s_0. Let $P(x|\pi)$ denote the *conditional probability of x given π*. Then $P(x|\pi)$ is the product of the probabilities of emitting x_i at the ith state along π for all i with $1 \le i \le m$. More succinctly, $P(x|\pi) = \prod_{i=1}^{m} e_{p_i}(x_i)$, where s_{p_i} is the ith state on π. Also, let $P(x, \pi)$ denote the *joint probability of sequence x and path π*, i.e., the probability of both picking path π and generating x along that path. The probability of picking π is $P(\pi) = \prod_{i=1}^{m} t_{p_{i-1},p_i}$ (with p_i as above), and $P(x,\pi) = P(\pi)P(x|\pi)$.

Example. Suppose that a loaded die emits 1 with probability 0.5, and emits 2 through 6 each with probability 0.1. Suppose that the F-to-L (fair to loaded) transition has probability 0.1 and L-to-F has probability 0.2. Finally, with probability 0.9 we start with the fair die. Consider the observed sequence $x = 1214641$ and the hidden path $\pi =$ LFFFFFL. Then $P(x|\pi) = (0.5)(1/6)^5(0.5)$, $P(\pi) = (0.1)(0.2)(0.9)^4(0.1)$, and $P(x,\pi) = P(x|\pi)P(\pi)$. Note that in this model with these probabilities, $P(x,\pi) > 0$ whenever x and π have the same length.

Computing the probability that a given sequence is generated by the model. Let $P(x)$ denote the probability of generating x from $\mathcal{M}$. Thus $P(x)$ is $\sum_\pi P(x,\pi)$, summing over all paths π from s_0. (Only paths with exactly m edges contribute to the sum.) $P(x)$ can be computed by the so-called *Forward algorithm*, which is quite similar to the dynamic programming algorithm for aligning two sequences. For $i = 0, 1, \ldots, m$ (denoting a position in x) and $j = 0, 1, \ldots, n$ (denoting a state in $\mathcal{M}$), define $f_j(i)$ to be $\sum_p P(x_1x_2 \ldots x_i, p)$ over all paths p from s_0 to s_j. In other words, $f_j(i)$ is the probability of generating $x_1x_2 \ldots x_i$ and ending in state s_j. You can think of the f-values as forming a table with $m+1$ rows and $n+1$ columns. We will fill in the table by rows.

Row 0 consists of values $f_j(0)$, corresponding to paths from s_0 to s_j that "spell out" the first 0 symbols of x. Clearly $f_0(0) = 1$, and if $j > 0$ then $f_j(0) = 0$. For any later row, say row i, suppose that the f-values have been determined for row $i-1$, and fix s_j. Then $x_1x_2 \ldots x_i$ is generated by a path ending at s_j if and only if some state s_k satisfies (1) $x_1x_2 \ldots x_{i-1}$ is generated ending in s_k (probability $f_k(i-1)$), (2) the transition from s_k to s_j is chosen (probability $t_{k,j}$), and (3) x_i is emitted (probability $e_j(x_i)$). Summing over all possible s_k, we get the recurrence relation $f_j(i) = e_j(x_i)\sum_{k=0}^{n} t_{k,j} f_k(i-1)$. This gives the following algorithm.

$f_0(0) \leftarrow 1$; $f_j(0) \leftarrow 0$ for $j = 1, 2, \ldots, n$
for $i = 1$ to m do
    for $j = 1$ to n do
        $f_j(i) \leftarrow e_j(x_i)\sum_{k=0}^{n} t_{k,j} f_k(i-1)$
$P(x) \leftarrow \sum_{j=1}^{n} f_j(m)$

The Forward algorithm for HMMs.

Computing the most probable state path generating a given observed sequence. Given observed sequence x, we want the path π that maximizes $P(x,\pi)$. (This corresponds to GenScan's prediction of the most probably set of genes in a given genomic sequence.) Of course, several paths may tie for the most probable path, in which case the method will pick one of them. In essence, an optimal path can be found simply by replacing the sum operation in the Forward algorithm by a maximization. To see that this is justified, we reason as follows. For $i = 0, 1, \ldots, m$ and $j = 0, 1, \ldots, n$, define $v_j(i)$ to be the maximum $P(x_1x_2 \ldots x_i, p)$ over all paths p from s_0 to s_j. If p is restricted so that its last edge starts at s_k, then the best we can do is to optimally spell $x_1x_2 \ldots x_{i-1}$ with a path ending at s_k (probability $v_k(i-1)$), add the edge to s_j (probability $t_{k,j}$), and emit x_i (probability $e_j(x_i)$). This recurrence relation immediately gives the following algorithm for computing the number $\max_\pi P(x,\pi)$.

To explicitly determine an optimizing path π, one can save back-pointers. That is, each time a $v_j(i)$ is computed, one can determine and save *backpointer*$_j(i)$, defined as the k (or one of them, in case of a tie) such that s_k immediately precedes s_j on an optimal path spelling $x_1x_2 \ldots x_i$ and ending at s_j (i.e., the k that maximizes the expression used to define $v_j(i)$ in the above pseudo-code). These edges can be used to trace out an optimal path in reverse order.

$v_0(0) \leftarrow 1$; $v_j(0) \leftarrow 0$ for $j = 1, 2, \ldots, n$
for $i = 1$ to m do
    for $j = 1$ to n do
        $v_j(i) \leftarrow e_j(x_i) \max_{k=0}^{n} t_{k,j} v_k(i-1)$
$\max_\pi P(x, \pi)$ is $\max_{j=0}^{n} v_j(m)$

The Viterbi algorithm for HMMs.

Computing the probability that a given observed symbol was generated by a given state. Fix i where $1 \le i \le m$, which selects element x_i of the observed sequence x. For some or all states s_j of $\mathcal{M}$ we want to compute the probability that x_i is emitted in s_j, given that x is emitted by the full path. This value can be denoted as $P(\pi_i = s_j | x)$, using π_i to denote the ith state on π. It is analogous to the probability that a certain genomic segment corresponds to an exon state of GenScan. More precisely, for fixed j we want to sum $P(x, \pi)$ over all paths π whose ith state is s_j. Dividing this value by $P(x)$ gives $P(\pi_i = s_j | x)$.

Recall that $f_j(i)$, as computed by the Forward algorithm, is the probability of emitting $x_1 x_2 \ldots x_i$ and ending in state s_j (i.e., it equals $\sum P(x_1 x_2 \ldots x_i, p)$ over all paths p from s_0 to s_j). We need to multiply this by $b_j(i)$, defined as the probability of emitting $x_{i+1} x_{i+2} \ldots x_m$, given s_j as the starting point (and not emitting anything until after a state transition). In symbols, $b_j(i) = P(x_{i+1} x_{i+2} \ldots x_m | \pi_i = s_j)$. The values $b_j(h)$ can be computed in backwards order (i.e., decreasing h) beginning with $h = m$. The desired recurrence relation follows from the observation that $b_j(h)$ is the sum over all s_k of the probability of a transition from s_j to s_k (namely $t_{j,k}$) times the probability of emitting x_{h+1} in state s_k (namely $e_k(h+1)$) times the probability of emitting $x_{h+2} x_{h+3} \ldots x_m$, starting at s_k (namely $b_k(h+1)$).

Compute $P(x)$ and values $f_j(i)$ for all j using the Forward algorithm.
$b_j(m) \leftarrow 1$ for $j = 1, 2, \ldots, n$
for $h = m - 1$ down to i do
    for $j = 1$ to n do
        $b_j(h) \leftarrow \sum_{k=1}^{n} t_{j,k} e_k(h+1) b_k(h+1)$
for $j = 1$ to n do
    $P(\pi_i = s_j | x)$ is $f_j(i) b_j(i) / P(x)$

The Forward/Backward algorithm for HMMs.

Hidden semi-Markov models (as in GenScan). Suppose that the probability of picking observed length ℓ in state s_j is $L_j(\ell)$ and that the probability of emitting a string y of length ℓ in state s_j is $E_{j,\ell}(y)$. Let $v_j(i)$ denote the maximum joint probability of picking a state path π from s_0 to s_j and emitting $x_1 x_2 \ldots x_i$. If the last edge on π is from s_k to s_j and if $x_{h+1} x_{h+2} \ldots x_i$ is emitted in state s_j, then the relevant value is the probability of emitting $x_1 x_2 \ldots x_h$ and ending in state s_k (namely $v_k(h)$) time the probability of a transition to s_j (namely $t_{k,j}$) time the probability of picking emitted sequence length $i - h$ (namely $L_j(i-h)$) times the probability of emitting $x_{h+1} x_{h+2} \ldots x_i$ (namely $E_{j,i-h}(x_{h+1} x_{h+2} \ldots x_i)$). This gives following recurrence relation.

$$v_j(i) = \max_k [\max_{h<i} E_{j,i-h}(x_{h+1} x_{h+2} \ldots x_i) L_j(i-h) t_{k,j} v_k(h)]$$

Reasoning of this sort gives the appropriate variants of the Forward, Viterbi and Forward/Backward algorithms for hidden semi-Markov models.