A Revealing Introduction to Hidden Markov Models

Mark Stamp*
Department of Computer Science
San Jose State University

October 17, 2018

1 A simple example

Suppose we want to determine the average annual temperature at a particular location on earth over a series of years. To make it interesting, suppose the years we are concerned with lie in the distant past, before thermometers were invented. Since we can't go back in time, we instead look for indirect evidence of the temperature.

To simplify the problem, we only consider two annual temperatures, "hot" and "cold". Suppose that modern evidence indicates that the probability of a hot year followed by another hot year is 0.7 and the probability that a cold year is followed by another cold year is 0.6. We'll assume that these probabilities held in the distant past as well. The information so far can be summarized as

$$\begin{array}{c} \\ H \\ C \end{array} \begin{array}{c} \begin{array}{cc} H & C \end{array} \\ \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \end{array} \tag{1}$$

where H is "hot" and C is "cold".

Also suppose that current research indicates a correlation between the size of tree growth rings and temperature. For simplicity, we only consider three different tree ring sizes, small, medium and large, or S, M and L, respectively. Finally, suppose that based on available evidence, the probabilistic relationship between annual temperature and tree ring sizes is given by

$$\begin{array}{c} \\ H \\ C \end{array} \begin{array}{c} \begin{array}{ccc} S & M & L \end{array} \\ \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \end{array}. \tag{2}$$

*Email: mark.stamp@sjsu.edu. This tutorial was originally published online in 2004. Minor corrections and additions have been made over time, with new (and improved!) exercises added. This current version is suspiciously similar to Chapter 2 of my book, *Introduction to Machine Learning with Applications in Information Security* [5].

For this system, the *state* is the average annual temperature—either H or C. The transition from one state to the next is a *Markov process* (of order one[1]), since the next state depends only on the current state and the fixed probabilities in (1). However, the actual states are "hidden" since we can't directly observe the temperature in the past.

Although we can't observe the state (temperature) in the past, we can observe the size of tree rings. From (2), tree rings provide us with probabilistic information regarding the temperature. Since the states are hidden, this type of system is known as a *Hidden Markov Model* (HMM). Our goal is to make effective and efficient use of the observable information so as to gain insight into various aspects of the Markov process.

The state transition matrix

$$A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \tag{3}$$

comes from (1) and the observation matrix

$$B = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}. \tag{4}$$

is from (2). In this example, suppose that the initial state distribution, denoted by π, is

$$\pi = \begin{bmatrix} 0.6 & 0.4 \end{bmatrix}. \tag{5}$$

The matrices π, A and B are *row stochastic*, meaning that each element is a probability and the elements of each row sum to 1, that is, each row is a probability distribution.

Now consider a particular four-year period of interest from the distant past, for which we observe the series of tree rings S, M, S, L. Letting 0 represent S, 1 represent M and 2 represent L, this observation sequence is

$$\mathcal{O} = (0, 1, 0, 2). \tag{6}$$

We might want to determine the most likely state sequence of the Markov process given the observations (6). That is, we might want to know the most likely average annual temperatures over the four-year period of interest. This is not quite as clear-cut as it seems, since there are different possible interpretations of "most likely." On the one hand, we could reasonably define "most likely" as the state sequence with the highest probability from among all possible state sequences of length four. Dynamic programming (DP) can be used to efficiently finds this particular solution. On the other hand, we might reasonably define "most likely" as the state sequence that maximizes the expected number of correct states. HMMs can be used to find this sequence.

The DP solution and the HMM solution are not necessarily the same. For example, the DP solution must have valid state transitions, while this is not necessarily the case for the HMMs. And even if all state transitions are valid, the HMM solution can still differ from the DP solution—as illustrated in the example below.

Next, we present one of the most challenging aspects of HMMs, namely, the notation. Then we discuss the three fundamental problems related to HMMs and give algorithms

[1]A Markov process of order two would depend on the two preceding states, a Markov process of order three would depend on the three preceding states, and so on. In any case, the "memory" is finite.

for their efficient solutions. We also consider some critical computational issue that must be addressed when writing any HMM computer program. We conclude with a substantial example that does not require any specialized knowledge, yet nicely illustrates the strength of the HMM approach. Rabiner [3] is the best source for further introductory information on HMMs.

2 Notation

Let

T = length of the observation sequence
N = number of states in the model
M = number of observation symbols
Q = $\{q_0, q_1, \ldots, q_{N-1}\}$ = distinct states of the Markov process
V = $\{0, 1, \ldots, M-1\}$ = set of possible observations
A = state transition probabilities
B = observation probability matrix
π = initial state distribution
$\mathcal{O}$ = $(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$ = observation sequence.

The observations are always denoted by $\{0, 1, \ldots, M-1\}$, since this simplifies the notation with no loss of generality. That is, $\mathcal{O}_i \in V$ for $i = 0, 1, \ldots, T-1$.

A generic hidden Markov model is illustrated in Figure 1, where the X_i represent the hidden state sequence and all other notation is as given above. The Markov process—which is hidden behind the dashed line—is determined by the current state and the A matrix. We are only able to observe the $\mathcal{O}_i$, which are related to the (hidden) states of the Markov process by the matrix B.

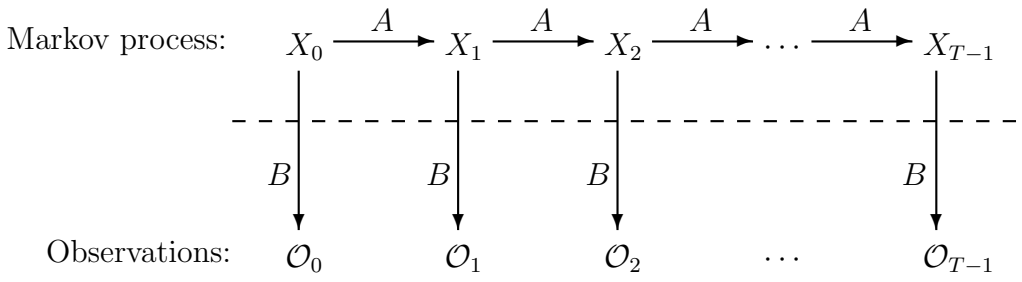

Figure 1: Hidden Markov Model

For the temperature example of the previous section—with the observations sequence given in (6)—we have $T = 4$, $N = 2$, $M = 3$, $Q = \{H, C\}$, $V = \{0, 1, 2\}$ (where we let $0, 1, 2$ represent "small", "medium" and "large" tree rings, respectively). In this case, the matrices A, B and π are given by (3), (4) and (5), respectively.

The matrix $A = \{a_{ij}\}$ is $N \times N$ with

$$a_{ij} = P(\text{state } q_j \text{ at } t+1 \,|\, \text{state } q_i \text{ at } t)$$

and A is row stochastic. Note that the probabilities a_{ij} are independent of t. The matrix $B = \{b_j(k)\}$ is an $N \times M$ with

$$b_j(k) = P(\text{observation } k \text{ at } t \,|\, \text{state } q_j \text{ at } t).$$

As with A, the matrix B is row stochastic and the probabilities $b_j(k)$ are independent of t. The unusual notation $b_j(k)$ is standard in the HMM world.

An HMM is defined by A, B and π (and, implicitly, by the dimensions N and M). The HMM is denoted by $\lambda = (A, B, \pi)$.

Consider a generic state sequence of length four

$$X = (x_0, x_1, x_2, x_3)$$

with corresponding observations

$$\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3).$$

Then π_{x_0} is the probability of starting in state x_0. Also, $b_{x_0}(\mathcal{O}_0)$ is the probability of initially observing $\mathcal{O}_0$ and a_{x_0,x_1} is the probability of transiting from state x_0 to state x_1. Continuing, we see that the probability of the state sequence X is given by

$$P(X, \mathcal{O}) = \pi_{x_0} b_{x_0}(\mathcal{O}_0) a_{x_0,x_1} b_{x_1}(\mathcal{O}_1) a_{x_1,x_2} b_{x_2}(\mathcal{O}_2) a_{x_2,x_3} b_{x_3}(\mathcal{O}_3). \tag{7}$$

Consider again the temperature example in Section 1 with observation sequence $\mathcal{O} = (0, 1, 0, 2)$, as given in (6). Using (7) we can compute, say,

$$P(HHCC) = 0.6(0.1)(0.7)(0.4)(0.3)(0.7)(0.6)(0.1) = 0.000212.$$

Similarly, we can directly compute the probability of each possible state sequence of length four, assuming the given observation sequence (6). We have listed these results in Table 1, where the probabilities in the last column are normalized so that they sum to 1.

To find the optimal state sequence in the dynamic programming (DP) sense, we simply choose the sequence with the highest probability, namely, $CCCH$. To find the optimal sequence in the HMM sense, we choose the most probable symbol at each position. To this end we sum the probabilities in Table 1 that have an H in the first position. Doing so, we find the (normalized) probability of H in the first position is 0.18817 and hence the probability of C in the first position is 0.81183. The HMM therefore chooses the first element of the optimal sequence to be C. We repeat this for each element of the sequence, obtaining the probabilities in Table 2.

From Table 2 we find that the optimal sequence—in the HMM sense—is $CHCH$. Note that in this example, the optimal DP sequence differs from the optimal HMM sequence and all state transitions are valid.

3 The three problems

There are three fundamental problems that we can solve using HMMs. Here, we briefly describe these three problems, and in the next section we give efficient algorithms for their solution.

state	probability	normalized probability
$HHHH$	.000412	.042787
$HHHC$	.000035	.003635
$HHCH$	.000706	.073320
$HHCC$	.000212	.022017
$HCHH$	.000050	.005193
$HCHC$	.000004	.000415
$HCCH$	.000302	.031364
$HCCC$	.000091	.009451
$CHHH$	.001098	.114031
$CHHC$	.000094	.009762
$CHCH$	.001882	.195451
$CHCC$	.000564	.058573
$CCHH$	.000470	.048811
$CCHC$	.000040	.004154
$CCCH$	.002822	.293073
$CCCC$	.000847	.087963

Table 1: State sequence probabilities

	element			
	0	1	2	3
$P(H)$	0.188182	0.519576	0.228788	0.804029
$P(C)$	0.811818	0.480424	0.771212	0.195971

Table 2: HMM probabilities

3.1 Problem 1

Given the model $\lambda = (A, B, \pi)$ and a sequence of observations $\mathcal{O}$, find $P(\mathcal{O} \mid \lambda)$. Here, we want to determine a score for the observed sequence $\mathcal{O}$ with respect to the given model λ.

3.2 Problem 2

Given $\lambda = (A, B, \pi)$ and an observation sequence $\mathcal{O}$, find an optimal state sequence for the underlying Markov process. In other words, we want to uncover the hidden part of the Hidden Markov Model. This type of problem is discussed in some detail in Section 1, above.

3.3 Problem 3

Given an observation sequence $\mathcal{O}$ and the dimensions N and M, find the model $\lambda = (A, B, \pi)$ that maximizes the probability of $\mathcal{O}$. This can be viewed as training a model to best fit the

observed data. Alternatively, we can view this as a (discrete) hill climb on the parameter space represented by A, B and π.

3.4 Discussion

Consider the problem of speech recognition (which just happens to be one of the best-known applications of HMMs). We can use the solution to Problem 3 to train an HMM, say, λ_0 to recognize the spoken word "no" and train another HMM, say, λ_1 to recognize the spoken word "yes". Then given an unknown spoken word, we can use the solution to Problem 1 to score this word against λ_0 and also against λ_1 to determine whether it is more likely "no," "yes," or neither. In this case, we don't need to solve Problem 2, but it is possible that such a solution—which uncovers the hidden states—might provide additional insight into the underlying speech model.

4 The three solutions

4.1 Solution to Problem 1

Let $\lambda = (A, B, \pi)$ be a given model and let $\mathcal{O} = (O_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$ be a series of observations. We want to find $P(\mathcal{O} \mid \lambda)$.

Let $X = (x_0, x_1, \ldots, x_{T-1})$ be a state sequence. Then by the definition of B we have

$$P(\mathcal{O} \mid X, \lambda) = b_{x_0}(\mathcal{O}_0) b_{x_1}(\mathcal{O}_1) \cdots b_{x_{T-1}}(\mathcal{O}_{T-1})$$

and by the definition of π and A it follows that

$$P(X \mid \lambda) = \pi_{x_0} a_{x_0,x_1} a_{x_1,x_2} \cdots a_{x_{T-2},x_{T-1}}.$$

Since

$$P(\mathcal{O}, X \mid \lambda) = \frac{P(\mathcal{O} \cap X \cap \lambda)}{P(\lambda)}$$

and

$$P(\mathcal{O} \mid X, \lambda) P(X \mid \lambda) = \frac{P(\mathcal{O} \cap X \cap \lambda)}{P(X \cap \lambda)} \cdot \frac{P(X \cap \lambda)}{P(\lambda)} = \frac{P(\mathcal{O} \cap X \cap \lambda)}{P(\lambda)}$$

we have

$$P(\mathcal{O}, X \mid \lambda) = P(\mathcal{O} \mid X, \lambda) P(X \mid \lambda).$$

By summing over all possible state sequences we obtain

$$\begin{aligned} P(\mathcal{O} \mid \lambda) &= \sum_X P(\mathcal{O}, X \mid \lambda) \\ &= \sum_X P(\mathcal{O} \mid X, \lambda) P(X \mid \lambda) \\ &= \sum_X \pi_{x_0} b_{x_0}(\mathcal{O}_0) a_{x_0,x_1} b_{x_1}(\mathcal{O}_1) \cdots a_{x_{T-2},x_{T-1}} b_{x_{T-1}}(\mathcal{O}_{T-1}). \end{aligned}$$

However, this direct computation is generally infeasible, since it requires about $2TN^T$ multiplications. The strength of HMMs derives largely from the fact that there exists an efficient algorithm to achieve the same result.

To find $P(\mathcal{O} \mid \lambda)$, the so-called *forward algorithm*, or α-*pass*, is used. For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, define

$$\alpha_t(i) = P(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_t, x_t = q_i \mid \lambda). \tag{8}$$

Then $\alpha_t(i)$ is the probability of the partial observation sequence up to time t, where the underlying Markov process is in state q_i at time t.

The crucial insight here is that the $\alpha_t(i)$ can be computed recursively as follows.

1. Let $\alpha_0(i) = \pi_i b_i(\mathcal{O}_0)$, for $i = 0, 1, \ldots, N-1$
2. For $t = 1, 2, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, compute
$$\alpha_t(i) = \left[\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji}\right] b_i(\mathcal{O}_t)$$
3. Then from (8) it is clear that
$$P(\mathcal{O} \mid \lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i).$$

The forward algorithm only requires about N^2T multiplications, as opposed to more than $2TN^T$ for the naïve approach.

4.2 Solution to Problem 2

Given the model $\lambda = (A, B, \pi)$ and a sequence of observations $\mathcal{O}$, our goal is to find the most likely state sequence. As mentioned above, there are different possible interpretations of "most likely." For HMMs we want to maximize the expected number of correct states. In contrast, a dynamic program finds the highest scoring overall path. As we have seen, these solutions are not necessarily the same.

First, we define the *backward algorithm*, or β-*pass*. This is analogous to the α-pass discussed above, except that it starts at the end and works back toward the beginning.

For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, define

$$\beta_t(i) = P(\mathcal{O}_{t+1}, \mathcal{O}_{t+2}, \ldots, \mathcal{O}_{T-1} \mid x_t = q_i, \lambda).$$

Then the $\beta_t(i)$ can be computed recursively (and efficiently) as follows.

1. Let $\beta_{T-1}(i) = 1$, for $i = 0, 1, \ldots, N-1$.
2. For $t = T-2, T-3, \ldots, 0$ and $i = 0, 1, \ldots, N-1$ compute
$$\beta_t(i) = \sum_{j=0}^{N-1} a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j).$$

For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, define

$$\gamma_t(i) = P(x_t = q_i \,|\, \mathcal{O}, \lambda).$$

Since $\alpha_t(i)$ measures the relevant probability up to time t and $\beta_t(i)$ measures the relevant probability after time t,

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathcal{O} \,|\, \lambda)}.$$

Recall that the denominator $P(\mathcal{O} \,|\, \lambda)$ is obtained by summing $\alpha_{T-1}(i)$ over i. From the definition of $\gamma_t(i)$ it follows that the most likely state at time t is the state q_i for which $\gamma_t(i)$ is maximum, where the maximum is taken over the index i.

4.3 Solution to Problem 3

Here we want to adjust the model parameters to best fit the observations. The sizes of the matrices (N and M) are fixed but the elements of A, B and π are to be determined, subject to the row stochastic condition. The fact that we can efficiently re-estimate the model itself is one of the more amazing aspects of HMMs.

For $t = 0, 1, \ldots, T-2$ and $i, j \in \{0, 1, \ldots, N-1\}$, define "di-gammas" as

$$\gamma_t(i, j) = P(x_t = q_i, x_{t+1} = q_j \,|\, \mathcal{O}, \lambda).$$

Then $\gamma_t(i, j)$ is the probability of being in state q_i at time t and transiting to state q_j at time $t+1$. The di-gammas can be written in terms of α, β, A and B as

$$\gamma_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)}{P(\mathcal{O} \,|\, \lambda)}.$$

For $t = 0, 1, \ldots, T-2$,the $\gamma_t(i)$ and $\gamma_t(i, j)$ are related by

$$\gamma_t(i) = \sum_{j=0}^{N-1} \gamma_t(i, j).$$

Given the γ and di-gamma we verify below that the model $\lambda = (A, B, \pi)$ can be re-estimated as follows.

1. For $i = 0, 1, \ldots, N-1$, let
$$\pi_i = \gamma_0(i) \tag{9}$$

2. For $i = 0, 1, \ldots, N-1$ and $j = 0, 1, \ldots, N-1$, compute
$$a_{ij} = \sum_{t=0}^{T-2} \gamma_t(i, j) \Bigg/ \sum_{t=0}^{T-2} \gamma_t(i) \,. \tag{10}$$

3. For $j = 0, 1, \ldots, N-1$ and $k = 0, 1, \ldots, M-1$, compute
$$b_j(k) = \sum_{\substack{t \in \{0,1,\ldots,T-1\} \\ \mathcal{O}_t = k}} \gamma_t(j) \Bigg/ \sum_{t=0}^{T-1} \gamma_t(j) \,. \tag{11}$$

The numerator of the re-estimated a_{ij} can be seen to give the expected number of transitions from state q_i to state q_j, while the denominator is the expected number of transitions from q_i to any state. Then the ratio is the probability of transiting from state q_i to state q_j, which is the desired value of a_{ij}.

The numerator of the re-estimated $b_j(k)$ is the expected number of times the model is in state q_j with observation k, while the denominator is the expected number of times the model is in state q_j. The ratio is the probability of observing symbol k, given that the model is in state q_j, which is the desired value of $b_j(k)$.

Re-estimation is an iterative process. First, we initialize $\lambda = (A, B, \pi)$ with a best guess or, if no reasonable guess is available, we choose random values such that $\pi_i \approx 1/N$ and $a_{ij} \approx 1/N$ and $b_j(k) \approx 1/M$. It's critical that A, B and π be randomized, since exactly uniform values will result in a local maximum from which the model cannot climb. As always, π, A and B must be row stochastic.

The solution to Problem 3 can be summarized as follows.

1. Initialize, $\lambda = (A, B, \pi)$.
2. Compute $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i, j)$ and $\gamma_t(i)$.
3. Re-estimate the model $\lambda = (A, B, \pi)$.
4. If $P(\mathcal{O} \,|\, \lambda)$ increases, goto 2.

Of course, it might be desirable to stop if $P(\mathcal{O} \,|\, \lambda)$ does not increase by at least some predetermined threshold and/or to set a maximum number of iterations.

5 Dynamic programming

Before completing our discussion of the elementary aspects of HMMs, we make a brief detour to show the relationship between dynamic programming (DP) and HMMs. The executive summary is that a DP can be viewed as an α-pass where "sum" is replaced by "max." More precisely, for π, A and B as above, the dynamic programming algorithm can be stated as follows.

1. Let $\delta_0(i) = \pi_i b_i(\mathcal{O}_0)$, for $i = 0, 1, \ldots, N-1$.
2. For $t = 1, 2, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, compute

$$\delta_t(i) = \max_{j \in \{0,1,\ldots,N-1\}} [\delta_{t-1}(j) a_{ji} b_i(\mathcal{O}_t)]$$

At each successive t, the DP determines the probability of the best path ending at each of the states $i = 0, 1, \ldots, N-1$. Consequently, the probability of the best overall path is

$$\max_{j \in \{0,1,\ldots,N-1\}} [\delta_{T-1}(j)]. \tag{12}$$

Be sure to note that (12) only gives the optimal probability, not the optimal path itself. By keeping track of each preceding state, the DP procedure given here can be augmented to recover the optimal path by tracing back from the highest-scoring final state.

Consider the example of Section 1. The initial probabilities are

$$P(H) = \pi_0 b_0(0) = 0.6(0.1) = 0.06 \text{ and } P(C) = \pi_1 b_1(0) = 0.4(0.7) = 0.28.$$

The probabilities of the paths of length two are

$$\begin{aligned} P(HH) &= 0.06(0.7)(0.4) = 0.0168 \\ P(HC) &= 0.06(0.3)(0.2) = 0.0036 \\ P(CH) &= 0.28(0.4)(0.4) = 0.0448 \\ P(CC) &= 0.28(0.6)(0.2) = 0.0336 \end{aligned}$$

and hence the best (most probable) path of length two ending with H is CH while the best path of length two ending with C is CC. Continuing, we construct the the diagram in Figure 2 one "level" at a time, where each arrow points to the preceding element in the optimal path up to that particular state. Note that at each stage, the dynamic programming algorithm only needs to maintain the highest-scoring paths at each possible state—not a list of all possible paths. This is the key to the efficiency of the DP algorithm.

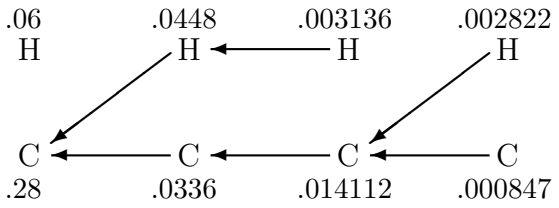

Figure 2: Dynamic programming

In Figure 2, the maximum final probability is 0.002822 which occurs at the final state H. We can use the arrows to trace back from H to find the optimal path $CCCH$. Note that this agrees with the brute force calculation in Table 1.

Underflow is a concern with a dynamic programming problem of this form, since we compute products of probabilities. Fortunately, underflow is easily avoided by simply taking logarithms. The Following is the underflow-resistant version of the DP algorithm.

1. Let $\hat{\delta}_0(i) = \log[\pi_i b_i(\mathcal{O}_0)]$, for $i = 0, 1, \ldots, N-1$.
2. For $t = 1, 2, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, compute

$$\hat{\delta}_t(i) = \max_{j \in \{0,1,\ldots,N-1\}} \left\{ \hat{\delta}_{t-1}(j) + \log[a_{ji}] + \log[b_i(\mathcal{O}_t)] \right\}.$$

In this case, the optimal score is

$$\max_{j \in \{0,1,\ldots,N-1\}} [\hat{\delta}_{T-1}(j)].$$

Of course, additional bookkeeping is still required to find the optimal path.

6 HMM Scaling

The three HMM solutions in Section 4 all require computations involving products of probabilities. It is easy to see, for example, that $\alpha_t(i)$ tends to 0 exponentially as T increases. Therefore, any attempt to implement the formulae as given above will inevitably result in underflow. The solution to this underflow problem is to scale the numbers. However, care must be taken to insure that, for example, the re-estimation formulae remain valid.

First, consider the computation of $\alpha_t(i)$. The basic recurrence is

$$\alpha_t(i) = \sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji} b_i(\mathcal{O}_t).$$

It seems sensible to normalize each $\alpha_t(i)$ by dividing by the sum (over j) of $\alpha_t(j)$. However, we must verify that the re-estimation formulae hold.

For $t = 0$, let $\tilde{\alpha}_0(i) = \alpha_0(i)$ for $i = 0, 1, \ldots, N-1$. Then let $c_0 = 1/\sum_{j=0}^{N-1} \tilde{\alpha}_0(j)$ and, finally, $\hat{\alpha}_0(i) = c_0 \tilde{\alpha}_0(i)$ for $i = 0, 1, \ldots, N-1$. Then for each $t = 1, 2, \ldots, T-1$ do the following.

1. For $i = 0, 1, \ldots, N-1$, compute

$$\tilde{\alpha}_t(i) = \sum_{j=0}^{N-1} \hat{\alpha}_{t-1}(j) a_{ji} b_i(\mathcal{O}_t).$$

2. Let

$$c_t = \frac{1}{\sum_{j=0}^{N-1} \tilde{\alpha}_t(j)}.$$

3. For $i = 0, 1, \ldots, N-1$, compute

$$\hat{\alpha}_t(i) = c_t \tilde{\alpha}_t(i).$$

Apparently $\hat{\alpha}_0(i) = c_0 \alpha_0(i)$. Suppose that

$$\hat{\alpha}_t(i) = c_0 c_1 \cdots c_t \alpha_t(i). \tag{13}$$

Then

$$\begin{aligned}
\hat{\alpha}_{t+1}(i) &= c_{t+1} \tilde{\alpha}_{t+1}(i) \\
&= c_{t+1} \sum_{j=0}^{N-1} \hat{\alpha}_t(j) a_{ji} b_i(\mathcal{O}_{t+1}) \\
&= c_0 c_1 \cdots c_t c_{t+1} \sum_{j=0}^{N-1} \alpha_t(j) a_{ji} b_i(\mathcal{O}_{t+1}) \\
&= c_0 c_1 \cdots c_{t+1} \alpha_{t+1}(i)
\end{aligned}$$

and hence (13) holds, by induction, for all t.

From (13) and the definitions of $\tilde{\alpha}$ and $\hat{\alpha}$ it follows that

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=0}^{N-1} \alpha_t(j)} \tag{14}$$

and hence $\hat{\alpha}_t(i)$ are the desired scaled values of $\alpha_t(i)$ for all t.

As a consequence of (14),

$$\sum_{j=0}^{N-1} \hat{\alpha}_{T-1}(j) = 1.$$

Also, from (13) we have

$$\begin{aligned} \sum_{j=0}^{N-1} \hat{\alpha}_{T-1}(j) &= c_0 c_1 \cdots c_{T-1} \sum_{j=0}^{N-1} \alpha_{T-1}(j) \\ &= c_0 c_1 \cdots c_{T-1} P(\mathcal{O} \,|\, \lambda). \end{aligned}$$

Combining these results gives

$$P(\mathcal{O} \,|\, \lambda) = \frac{1}{\prod_{j=0}^{T-1} c_j}.$$

To avoid underflow, we instead compute

$$\log[P(\mathcal{O} \,|\, \lambda)] = -\sum_{j=0}^{T-1} \log c_j. \tag{15}$$

The same scale factor is used for $\beta_t(i)$ as was used for $\alpha_t(i)$, namely c_t, so that we have $\hat{\beta}_t(i) = c_t \beta_t(i)$. We then compute $\gamma_t(i,j)$ and $\gamma_t(i)$ using the formulae of the previous section with $\hat{\alpha}_t(i)$ and $\hat{\beta}_t(i)$ in place of $\alpha_t(i)$ and $\beta_t(i)$, respectively. The resulting gammas and di-gammas are then used to re-estimate π, A and B.

By writing the original re-estimation formulae (9) and (10) and (11) directly in terms of $\alpha_t(i)$ and $\beta_t(i)$, it is an easy exercise to show that the re-estimated π and A and B are exact when $\hat{\alpha}_t(i)$ and $\hat{\beta}_t(i)$ are used in place of $\alpha_t(i)$ and $\beta_t(i)$. Furthermore, $P(\mathcal{O} \,|\, \lambda)$ isn't required in the re-estimation formulae, since in each case it cancels in the numerator and denominator. Therefore, (15) can be used to verify that $P(\mathcal{O} \,|\, \lambda)$ is increasing at each iteration. Fortunately, we have no need to directly calculate the value of $P(\mathcal{O} \,|\, \lambda)$.

7 Putting it all together

Here we give complete pseudo-code for solving Problem 3, including scaling. This pseudo-code also provides everything needed to solve Problems 1 and 2.

The values N and M are fixed and the T observations

$$\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$$

are assumed known.

1. Initialization:

   Select initial values for the matrices A, B and π, where π is $1 \times N$, while $A = \{a_{ij}\}$ is $N \times N$ and $B = \{b_j(k)\}$ is $N \times M$, and all three matrices are row-stochastic. If known, use reasonable approximations for the matrix values, otherwise let $\pi_i \approx 1/N$ and $a_{ij} \approx 1/N$ and $b_j(k) \approx 1/M$. Be sure that each row sums to 1 and the elements of each matrix are *not* uniform. Then initialize

   `maxIters` = maximum number of re-estimation iterations
   `iters` = 0
   `oldLogProb` = $-\infty$.

2. The α-pass

```
// compute α0(i)
c0 = 0
for i = 0 to N − 1
    α0(i) = πi bi(O0)
    c0 = c0 + α0(i)
next i

// scale the α0(i)
c0 = 1/c0
for i = 0 to N − 1
    α0(i) = c0 α0(i)
next i

// compute αt(i)
for t = 1 to T − 1
    ct = 0
    for i = 0 to N − 1
        αt(i) = 0
        for j = 0 to N − 1
            αt(i) = αt(i) + αt−1(j) aji
        next j
        αt(i) = αt(i) bi(Ot)
        ct = ct + αt(i)
    next i
```

// scale $\alpha_t(i)$
$c_t = 1/c_t$
for $i = 0$ to $N - 1$
$\alpha_t(i) = c_t \alpha_t(i)$
next i
next t

3. The β-pass

// Let $\beta_{T-1}(i) = 1$, scaled by c_{T-1}
for $i = 0$ to $N - 1$
$\beta_{T-1}(i) = c_{T-1}$
next i

// β-pass
for $t = T - 2$ to 0 by -1
for $i = 0$ to $N - 1$
$\beta_t(i) = 0$
for $j = 0$ to $N - 1$
$\beta_t(i) = \beta_t(i) + a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)$
next j
// scale $\beta_t(i)$ with same scale factor as $\alpha_t(i)$
$\beta_t(i) = c_t \beta_t(i)$
next i
next t

4. Compute $\gamma_t(i, j)$ and $\gamma_t(i)$

// No need to normalize $\gamma_t(i, j)$ since using scaled α and β
for $t = 0$ to $T - 2$
for $i = 0$ to $N - 1$
$\gamma_t(i) = 0$
for $j = 0$ to $N - 1$
$\gamma_t(i, j) = (\alpha_t(i) a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j))$
$\gamma_t(i) = \gamma_t(i) + \gamma_t(i, j)$
next j
next i
next t

// Special case for $\gamma_{T-1}(i)$ (as above, no need to normalize)
for $i = 0$ to $N - 1$
$\gamma_{T-1}(i) = \alpha_{T-1}(i)$
next i

5. Re-estimate A, B and π

```
// re-estimate π
for i = 0 to N − 1
    π_i = γ_0(i)
next i

// re-estimate A
for i = 0 to N − 1
    denom = 0
    for t = 0 to T − 2
        denom = denom + γ_t(i)
    next t
    for j = 0 to N − 1
        numer = 0
        for t = 0 to T − 2
            numer = numer + γ_t(i, j)
        next t
        a_ij = numer/denom
    next j
next i

// re-estimate B
for i = 0 to N − 1
    denom = 0
    for t = 0 to T − 1
        denom = denom + γ_t(i)
    next t
    for j = 0 to M − 1
        numer = 0
        for t = 0 to T − 1
            if(O_t == j) then
                numer = numer + γ_t(i)
            end if
        next t
        b_i(j) = numer/denom
    next j
next i
```

6. Compute $\log[P(\mathcal{O} \mid \lambda)]$

```
logProb = 0
for i = 0 to T − 1
    logProb = logProb + log(c_i)
next i
logProb = −logProb
```

7. To iterate or not to iterate, that is the question...

```
iters = iters + 1
if (iters < maxIters and logProb > oldLogProb) then
    oldLogProb = logProb
    goto 2
else
    output λ = (π, A, B)
end if
```

8 A not-so-simple example

In this section we discuss a classic application of Hidden Markov Models, which appears to have orginated with Cave and Neuwirth [2]. This application nicely illustrates the strength of HMMs and has the additional advantage that it requires no background in any specialized field such as speech processing.

Suppose Marvin the Martian obtains a large body of English text, such as the "Brown Corpus," [1] which has about 1,000,000 words. Marvin, who has a working knowledge of HMMs, but no knowledge of English, would like to determine some basic properties of this mysterious writing system. A reasonable question he might ask is whether the characters can be partitioned into sets so that each set is "different" in a statistically significant way.

Marvin might consider attempting the following. First, remove all punctuation, numbers, etc., and convert all letters to lower case. This leaves 26 distinct letters and word space, for a total of 27 symbols. He could then test the hypothesis that there is an underlying Markov process (of order one) with two states. For each of these two hidden states, he assume that the 27 symbols are observed according to fixed probability distributions.

This defines an HMM with $N = 2$ and $M = 27$, where the state transition probabilities of the A matrix and the observation probabilities—the B matrix—are unknown, while the the observations are the series of characters found in the text. To find the most probable A and B matrices, Marvin must solve Problem 3 of Section 7.

We programmed this experiment, using the first $T = 50{,}000$ observations (letters—converted to lower case—and word spaces) from the "Brown Corpus" [1]. We initialized each element of π and A randomly to approximately $1/2$. The precise values used were

$$\pi = [\ 0.51316 \quad 0.48684\ ]$$

and

$$A = \begin{bmatrix} 0.47468 & 0.52532 \\ 0.51656 & 0.48344 \end{bmatrix}.$$

Each element of B was initialized to approximately $1/27$. The precise values in the initial B matrix (actually, the transpose of B) appear in the second and third columns of Figure 3.

	Initial		Final	
a	0.03735	0.03909	0.13845	0.00075
b	0.03408	0.03537	0.00000	0.02311
c	0.03455	0.03537	0.00062	0.05614
d	0.03828	0.03909	0.00000	0.06937
e	0.03782	0.03583	0.21404	0.00000
f	0.03922	0.03630	0.00000	0.03559
g	0.03688	0.04048	0.00081	0.02724
h	0.03408	0.03537	0.00066	0.07278
i	0.03875	0.03816	0.12275	0.00000
j	0.04062	0.03909	0.00000	0.00365
k	0.03735	0.03490	0.00182	0.00703
l	0.03968	0.03723	0.00049	0.07231
m	0.03548	0.03537	0.00000	0.03889
n	0.03735	0.03909	0.00000	0.11461
o	0.04062	0.03397	0.13156	0.00000
p	0.03595	0.03397	0.00040	0.03674
q	0.03641	0.03816	0.00000	0.00153
r	0.03408	0.03676	0.00000	0.10225
s	0.04062	0.04048	0.00000	0.11042
t	0.03548	0.03443	0.01102	0.14392
u	0.03922	0.03537	0.04508	0.00000
v	0.04062	0.03955	0.00000	0.01621
w	0.03455	0.03816	0.00000	0.02303
x	0.03595	0.03723	0.00000	0.00447
y	0.03408	0.03769	0.00019	0.02587
z	0.03408	0.03955	0.00000	0.00110
space	0.03688	0.03397	0.33211	0.01298

Figure 3: Initial and final B transpose

After the initial iteration, we have

$$\log[P(\mathcal{O}\,|\lambda)] = -165097.29$$

and after 100 iterations,

$$\log[P(\mathcal{O}\,|\,\lambda)] = -137305.28.$$

This shows that the model has been improved significantly.

After 100 iterations, the model $\lambda = (A, B, \pi)$ has converged to

$$\pi = \begin{bmatrix} 0.00000 & 1.00000 \end{bmatrix}$$

and

$$A = \begin{bmatrix} 0.25596 & 0.74404 \\ 0.71571 & 0.28429 \end{bmatrix}$$

with the transpose of B appearing in the last two columns of Figure 3.

The most interesting part of this result is the B matrix. Without having made any assumption about the two hidden states, the B matrix tells us that one hidden state contains the vowels while the other hidden state contains the consonants. Curiously, word-space is more like a vowel, while y is not even sometimes a vowel. Of course, anyone familiar with English would not be too surprised that there is a clear distinction between vowels and consonants. But the HMM result show us that this distinction is a statistically significant feature inherent in the language. And, thanks to HMMs, this could easily be deduced by Marvin the Martian, who has no background knowledge of the language.

Cave and Neuwirth [2] obtain further interesting results by allowing more than two hidden states. In fact, they are able to obtain and sensibly interpret the results for models with up to 12 hidden states.

9 Exercises

1. Suppose that you have trained an HMM and you obtain the model $\lambda = (A, B, \pi)$ where

$$A = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}, \quad B = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{pmatrix}, \quad \pi = \begin{pmatrix} 0.0 & 1.0 \end{pmatrix}.$$

   Furthermore, suppose the hidden states correspond to H and C, respectively, and the observations are S, M, and L, respectively. In this problem, we consider the observation sequence $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2) = (M, S, L)$.

   a) Directly compute $P(\mathcal{O} \mid \lambda)$. Since

$$P(\mathcal{O} \mid \lambda) = \sum_X P(\mathcal{O}, X \mid \lambda)$$

   we use the probabilities in $\lambda = (A, B, \pi)$ to compute each of the following for given observation sequence:

$$\begin{aligned}
P(\mathcal{O}, X = HHH) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = HHC) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = HCH) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = HCC) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = CHH) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = CHC) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____ \\
P(\mathcal{O}, X = CCH) &= 1.0 \cdot 0.2 \cdot 0.6 \cdot 0.7 \cdot 0.4 \cdot 0.5 = _____ \\
P(\mathcal{O}, X = CCC) &= ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ \cdot ____ = _____
\end{aligned}$$

   The desired probability is the sum of these 8 probabilities.

b) Compute $P(\mathcal{O}|\lambda)$ using the α pass. That is, compute

$$\begin{aligned}
\alpha_0(0) &= ____ \cdot ____ = _____ \\
\alpha_0(1) &= \underline{\ 1.0\ } \cdot \underline{\ 0.2\ } = _____ \\
\alpha_1(0) &= (____ \cdot ____ + ____ \cdot ____) \cdot ____ = _____ \\
\alpha_1(1) &= (____ \cdot ____ + ____ \cdot ____) \cdot ____ = _____ \\
\alpha_2(0) &= (____ \cdot ____ + ____ \cdot ____) \cdot ____ = _____ \\
\alpha_2(1) &= (____ \cdot ____ + ____ \cdot ____) \cdot ____ = _____
\end{aligned}$$

where the recurrence for $\alpha_t(i)$ is

$$\alpha_0(i) = \pi_i b_i(\mathcal{O}_0), \text{ for } i = 0, 1, \ldots, N-1$$

and

$$\alpha_t(i) = \left(\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji} \right) b_i(\mathcal{O}_t)$$

for $t = 1, 2, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$. The desired probability is given by

$$P(\mathcal{O} \mid \lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i).$$

c) Explain the results you obtained for parts a) and b). Be sure to explain why you obtained the results you did.

d) In terms of N and T, and counting only multiplications, what is the work factor for the method in part a)? The method in part b)?

2. For this problem, use the same model λ and observation sequence $\mathcal{O}$ given in Problem 1.

a) Determine the "best" hidden state sequence (X_0, X_1, X_2) in the dynamic programming sense.

b) Determine the "best" hidden state sequence (X_0, X_1, X_2) in the HMM sense.

3. Summing the numbers in the "probability" column of Table 1, we find

$$P(\mathcal{O} = (0, 1, 0, 2)) = 0.009629.$$

a) By a similar direct calculation, compute $P(\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3))$, where each $\mathcal{O}_i \in \{0, 1, 2\}$, and verify that $\sum P(\mathcal{O}) = 1$. You will use the probabilities for A, B and π given in equations (3), (4) and (5) in Section 1, respectively.

b) Use the forward algorithm to compute each $P(\mathcal{O})$ and verify that you obtain the same results as in part a).

4. Write the re-estimation formulae (9), (10) and (11) directly in terms of α and β.

5. In the re-estimation formulae obtained in the previous problem, substitute $\hat{\alpha}$ and $\hat{\beta}$ for α and β, respectively, and show that the resulting re-estimation formulae are exact.

6. Instead of using c_t to scale the $\beta_t(i)$, scale each $\beta_t(i)$ by

$$d_t = 1 \Big/ \sum_{j=0}^{N-1} \tilde{\beta}_t(j)$$

where the definition of $\tilde{\beta}$ is similar to that of $\tilde{\alpha}$.

   a) Using the scaling factors c_t and d_t show that the re-estimation formulae obtained in Exercise 1 are exact with $\hat{\alpha}$ and $\hat{\beta}$ in place of α and β.

   b) Write $\log\big(P(\mathcal{O} \,|\, \lambda)\big)$ in terms of c_t and d_t.

7. Consider an HMM where for each t the state transition matrix is time dependent. Then for each t, there is an $N \times N$ row-stochastic $A_t = \{a_{ij}^t\}$ that is used in place of A in the HMM computations. For such an HMM,

   a) Give pseudo-code to solve Problem 1.

   b) Give pseudo-code to solve Problem 2.

8. Consider an HMM of order two, that is, an HMM where the underlying Markov process is of order two. Then the the state at time t depends on the states at time $t-1$ and $t-2$ and a fixed set of probabilities. For an HMM of order two,

   a) Give pseudo-code to solve Problem 1.

   b) Give pseudo-code to solve Problem 2.

   c) Give pseudo-code to solve Problem 3.

9. Write an HMM program to solve the English language problem discussed in Section 8 with

   a) Two hidden states. Explain your results.

   b) Three hidden states. Explain your results.

   c) Four hidden states. Explain your results.

10. A ciphertext message is can be found here. This message was encrypted with a simple substitution.

   a) Use an HMM to determine the ciphertext letters that correspond to consonants and those that correspond to vowels.

   b) Use an HMM to solve this simple substitution. Hint: Let A be a 26×26 matrix that contains English digraph (relative) frequencies. Let B be the analogous matrix for the ciphertext. Train the model, holding A fixed. Analyze the final B matrix to determine the key.

11. Write an HMM program to solve the problem discussed in Section 8, replacing English text with

    a) French.

    b) Russian.

    c) Chinese.

12. Perform an HMM analysis similar to that discussed in Section 8, replacing English with "Hamptonese"; see

    http://www.cs.sjsu.edu/faculty/stamp/Hampton/hampton.html

    for information on Hamptonese.

References

[1] The Brown Corpus of standard American English, http://www.cs.toronto.edu/~gpenn/csc401/a1res.html

[2] R. L. Cave and L. P. Neuwirth, Hidden Markov models for English, in J. D. Ferguson, editor, *Hidden Markov Models for Speech*, IDA-CRD, Princeton, NJ, October 1980. https://www.cs.sjsu.edu/~stamp/RUA/CaveNeuwirth/index.html

[3] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, Vol. 77, No. 2, February 1989, https://www.cs.sjsu.edu/~stamp/RUA/Rabiner.pdf

[4] M. Stamp, Reference implementation of HMM in C (includes Brown Corpus), https://www.cs.sjsu.edu/~stamp/RUA/HMM_ref.zip

[5] M. Stamp, *Introduction to Machine Learning with Applications in Information Security*, Chapman & Hall/CRC Press, 2017.