Notes on the EM Algorithm

Michael Collins, September 24th 2005

1 Hidden Markov Models

A hidden Markov model (N, Σ, Θ) consists of the following elements:

- N is a positive integer specifying the number of states in the model. Without loss of generality, we will take the N'th state to be a special state, the *final* or *stop* state.
- Σ is a set of output symbols, for example $\Sigma = \{a, b\}$
- Θ is a vector of parameters. It contains three types of parameters:
  - π_j for $j = 1 \ldots N$ is the probability of choosing state j as an initial state. Note that $\sum_{j=1}^{N} \pi_j = 1$.
  - $a_{j,k}$ for $j = 1 \ldots (N-1)$, $k = 1 \ldots N$, is the probability of transitioning from state j to state k. Note that for all j, $\sum_{k=1}^{N} a_{j,k} = 1$.
  - $b_j(o)$ for $j = 1 \ldots (N-1)$, and $o \in \Sigma$, is the probability of emitting symbol o from state j. Note that for all j, $\sum_{o \in \Sigma} b_j(o) = 1$.

  Thus it can be seen that Θ is a vector of $N + (N-1)N + (N-1)|\Sigma|$ parameters.

An HMM specifies a probability for each possible (x, y) pair, where x is a sequence of symbols drawn from Σ, and y is a sequence of states drawn from the integers $1 \ldots (N-1)$. The sequences x and y are restricted to have the same length. As an example, say we have an HMM with $N = 3$, $\Sigma = \{a, b\}$, and with some choice of the parameters Θ. Take $x = \langle a, a, b, b \rangle$ and $y = \langle 1, 2, 2, 1 \rangle$. Then in this case,

$$P(x, y|\Theta) = \pi_1 \; a_{1,2} \; a_{2,2} \; a_{2,1} \; a_{1,3} \; b_1(a) \; b_2(a) \; b_2(b) \; b_1(b)$$

Thus we have a product of terms: the probability π_1 of starting in state 1; the probabilities $a_{1,2}$, $a_{2,2}$, $a_{2,1}$ $a_{1,3}$ specifying a series of transitions which terminate in the stop state 3; and emission probabilities $b_1(a), b_2(a), \ldots$ specifying the probability of emitting each symbol from its associated state.

In general, if we have the sequence $x = x_1, x_2, \ldots x_n$ where each $x_j \in \Sigma$, and the sequence $y = y_1, y_2, \ldots y_n$ where each $y_j \in 1 \ldots (N-1)$, then

$$P(x, y|\Theta) = \pi_{y_1} a_{y_n, N} \prod_{j=2}^{n} a_{y_{j-1}, y_j} \prod_{j=1}^{n} b_{y_j}(x_j)$$

Thus we see that $P(x, y|\Theta)$ is a simple function of the parameters Θ.

2 The basic setting in EM

We assume the following set-up:

- We have some data points—a "sample"—$x^1, x^2, \ldots x^m$. For example, each x^i might be a sentence such as "the dog slept": this will be the case in EM applied to hidden Markov models (HMMs) or probabilistic context-free-grammars (PCFGs). (Note that in this case each x^i is a *sequence*, which we will sometimes write $x^i_1, x^i_2, \ldots x^i_{n_i}$ where n_i is the length of the sequence.) Or in the three coins example (see the lecture notes), each x_i might be a sequence of three coin tosses, such as HHH, THT, or TTT.

- We have a parameter vector Θ. For example, see the description of HMMs in the previous section. As another example, in a PCFG, Θ would contain the probability $P(\alpha \rightarrow \beta|\alpha)$ for every rule expansion $\alpha \rightarrow \beta$ in the context-free grammar within the PCFG.

- We have a model $P(x, y|\Theta)$. This is essentially a function that for any x, y, Θ triple returns a probability, which is the probability of seeing x and y together. For example, see the description of HMMs in the previous section. Note that this model defines a *joint* distribution over x and y, but that we can also derive a *marginal* distribution over x alone, defined as

$$P(x|\Theta) = \sum_y P(x, y|\Theta)$$

  Thus $P(x|\Theta)$ is derived by summing over all possibilities for y. In the case of HMMs, if x is a sequence of length n, then we would sum over all state sequences of length n.

- Given the sample $x^1, x^2, \ldots x^m$, we define the *likelihood* as

$$L'(\Theta) = \prod_{i=1}^{m} P(x^i|\Theta) = \prod_{i=1}^{m} \sum_y P(x^i, y|\Theta)$$

  and we define the *log-likelihood* as

$$L(\Theta) = \log L'(\Theta) = \sum_{i=1}^{m} \log P(x^i|\Theta) = \sum_{i=1}^{m} \log \sum_y P(x^i, y|\Theta)$$

- The *maximum-likelihood estimation problem* is to find

$$\Theta_{ML} = \arg\max_{\Theta \in \Omega} L(\Theta)$$

  where Ω is a *parameter space* specifying the set of allowable parameter settings. In the HMM example, Ω would enforce the restrictions that all parameter values were ≥ 0; that $\sum_{j=1}^{N} \pi_j = 1$; that for all $j = 1 \ldots (N-1)$, $\sum_{k=1}^{N} a_{j,k} = 1$; and that for all $j = 1 \ldots (N-1)$, $\sum_{o \in \Sigma} b_j(o) = 1$.

To illustrate these definitions, say we would like to infer the parameters of an HMM from some data. For the HMM we'll assume $N = 3$, and $\Sigma = \{e, f, g, h\}$. These choices are fixed in the HMM. The parameter vector, Θ, is the one thing we'll learn from data. Say we now observe the following "sample" of 4 sequences, $x_1, x_2, \ldots x_4$:

e	g
e	h
f	h
f	g

Intuitively, a good setting for the parameters of the HMM would be:

$\pi_1 = 1.0$, $\pi_2 = \pi_3 = 0$
$b_1(e) = b_1(f) = 0.5$, $b_1(g) = b_1(h) = 0$
$b_2(e) = b_2(f) = 0$, $b_2(g) = b_2(h) = 0.5$
$a_{1,2} = 1.0$, $a_{1,1} = a_{1,3} = 0$
$a_{2,3} = 1.0$, $a_{2,1} = a_{2,2} = 0$

Under these definitions, the HMM always starts in state 1, and then transitions to state 2 followed by state 3, the final state. State 1 has a 50% chance of emitting either e or f, while state 2 has a 50% chance of emitting either g or h. These parameter settings appear to fit the sample of 4 sequences quite well.

The log-likelihood function $L(\Theta)$ in this case gives us a formal measure of how well a particular parameter setting Θ fits the observed sample. Note that $L(\Theta)$ is a function of both the parameters Θ and the data $x^1, x^2, \ldots x^4$. The higher $L(\Theta)$ is, the higher the probability assigned under the model to the observations $x^1, x^2, \ldots x^4$. In fact, if we could efficiently search for $\Theta_{ML} = \arg\max L(\Theta)$, in this case this would result in parameter settings such as the "intuitively" correct parameters shown above. Thus we now have a well motivated way of setting the parameters in the model given some observed data, i.e., the maximum likelihood estimates.

Note that this HMM example is a classic case of a situation with "hidden" or "latent" information. Each sample point x^i contains a sequence of symbols such as `e g`, but does **not** contain an underlying sequence of states, such as `1 2`. We can imagine that the data points $x^1, x^2, \ldots$ have been created in a process where in a first step an HMM is used to generate output sequences paired with underlying state sequences; but in the second step the state sequences are discarded. In this sense the state sequences are "hidden" or "latent" information.

3 Products of Multinomial (PM) Models

We now describe a class of models $P(x, y|\Theta)$ that is very important in NLP, and actually includes the three coins example as well as HMMs and PCFGs. This class of models uses *products of multinomial parameters*. We will refer to them as *PM models*. In the next section we'll describe the EM algorithm for this class of model.

Recall that in a PCFG, each sample point x is a sentence, and each y is a possible parse tree for that sentence. We have

$$P(x, y|\Theta) = \prod_{i=1}^{n} P(\alpha_i \rightarrow \beta_i | \alpha_i)$$

assuming that (x, y) contains the n context-free rules $\alpha_i \rightarrow \beta_i$ for $i = 1 \ldots n$. For example, if (x, y) contains the rules $\texttt{S} \rightarrow \texttt{NP VP}$, $\texttt{NP} \rightarrow \texttt{Jim}$, and $\texttt{VP} \rightarrow \texttt{sleeps}$, then

$$P(x, y|\Theta) = P(\texttt{S} \rightarrow \texttt{NP VP}|\texttt{S}) \times P(\texttt{NP} \rightarrow \texttt{Jim}|\texttt{NP}) \times P(\texttt{VP} \rightarrow \texttt{sleeps}|\texttt{VP})$$

Note that $P(x, y|\Theta)$ is a product of parameters, where each parameter is a member of a different multinomial distribution. In a PCFG, for each non-terminal α there is a different multinomial distribution $P(\alpha \rightarrow \beta|\alpha)$ for each non-terminal α in the grammar.

HMMs define a model with a similar form. Recall the example in the section on HMMs, where we had the following probability for a particular (x, y) pair:

$$P(x, y|\Theta) = \pi_1 \ a_{1,2} \ a_{2,2} \ a_{2,1} \ a_{1,3} \ b_1(a) \ b_2(a) \ b_2(b) \ b_1(b)$$

Again, notice that $P(x, y|\Theta)$ is a product of parameters, where each parameter is a member of some multinomial distribution.

In both HMMs and PCFGs, the model can be written in the following form

$$P(x, y|\Theta) = \prod_{r=1\ldots|\Theta|} \Theta_r^{Count(x,y,r)} \tag{1}$$

Here:

- Θ_r for $r = 1 \ldots |\Theta|$ is the r'th parameter in the model. Each parameter is the member of some multinomial distribution.
- $Count(x, y, r)$ for $r = 1 \ldots |\Theta|$ is a count corresponding to how many times Θ_r is seen in the expression for $P(x, y|\Theta)$.

We will refer to any model that can be written in the is form as a *product of multinomials* (PM) model. This class of model is important for a couple of reasons. First, it includes many models that we will come across in NLP. Second, as we will see in the next section, the EM algorithm—a method for finding the maximum likelihood estimates Θ_{ML}—takes a relatively simple form for PM models.

4 The EM Algorithm for PM Models

Figure 1 shows the EM algorithm for PM models. It is an iterative algorithm; we will use Θ^t to denote the parameter values at the t'th iteration of the algorithm. In the initialization step, some choice for initial parameter settings Θ^0 is made. The algorithm then defines an iterative sequence of parameters $\Theta^0, \Theta^1, \ldots, \Theta^T$, before returning Θ^T as the final parameter settings. In theory, it can be shown that as $T \rightarrow \infty$, Θ^T will converge to a point that is either a local maximum or saddle point of the log-likelihood function, $L(\Theta)$. In practice, EM is often quite quick to converge, perhaps taking a handful of iterations.

Note that at each iteration of the algorithm, two steps are taken. In the first step, *expected counts* $\overline{Count}(r)$ are calculated for each parameter Θ_r in the model. It can be verified that at the t'th iteration,

$$\overline{Count}(r) = \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, r)$$

For example, say we are estimating the parameters of a PCFG using the EM algorithm. Take a particular rule, such as $S \rightarrow NP \ VP$. Then the expected count for this rule at the t'th iteration will be

$$\overline{Count}(S \rightarrow NP \ VP)) = \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, S \rightarrow NP \ VP)$$

Note that we sum over all training examples $i = 1 \ldots m$, and we sum over all parse trees for each sample x^i. $Count(x^i, y, S \rightarrow NP \ VP)$ is the number of times that $S \rightarrow NP \ VP$ is seen in tree y for sentence x^i. The factor $P(y|x^i, \Theta^{t-1})$ in the sum means that each parse tree y for x^i makes a contribution of $P(y|x^i, \Theta^{t-1}) \times Count(x^i, y, S \rightarrow NP\ VP)$ to the expected count.

In the second step, we calculate the updated parameters Θ^t. These are calculated as simple functions of the expected counts. For example, we would re-estimate

$$P(S \rightarrow NP\ VP|S) = \frac{\overline{Count}(S \rightarrow NP\ VP)}{\sum_{S \rightarrow \beta \in R} \overline{Count}(S \rightarrow \beta)}$$

Note that the denominator in this term involves a summation over all rules of the form $S \rightarrow \beta$ in the grammar. This term ensures that $\sum_{S \rightarrow \beta \in R} P(S \rightarrow \beta|S) = 1$, the usual constraint on rule probabilities in PCFGs.

As another example, consider the EM algorithm applied to HMMs. Recall that there are three types of parameters in an HMM: initial state parameters such as π_1; transition parameters such as $a_{1,2}$; and emission parameters such as $b_1(e)$. Each of these parameters will have an associated expected count under the model. For example, define $Count(x^i, y, 1 \rightarrow 2)$ to be the number of times a transition from state 1 to state 2 is seen in y, and define $\overline{Count}(1 \rightarrow 2)$ to be the expected count in the training set of this transition, assuming the parameters θ^{t-1} at the t'th iteration. Then the following quantity will be calculated in the first step of the algorithm:

$$\overline{Count}(1 \rightarrow 2) = \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, 1 \rightarrow 2)$$

Moreover, in the second step the transition parameter $a_{1,2}$ will be re-estimated as

$$a_{1,2} = \frac{\overline{Count}(1 \rightarrow 2)}{\sum_{k=1}^{N} \overline{Count}(1 \rightarrow k)}$$

where in this case the denominator ensures that $\sum_{k=1}^{N} a_{1,k} = 1$. Similar calculations will be performed for other transition parameters, as well as the initial state parameters and emission parameters.

5 The Forward-Backward Algorithm for HMMs

5.1 Background

There is clearly a major problem for the algorithm in figure 1, at least when applied to HMMs (or PCFGs). For each training example x^i, the algorithm requires a "brute force" summation over all possible values for y. For example, with an HMM where $N = 3$, and an input sequence of length n, we need to sum over all possible state sequences of length n. There are 2^n possible state sequences in this case, an intractable number as n grows large.

Fortunately, there is a way of avoiding this brute force strategy with HMMs, using a dynamic programming algorithm called *the forward-backward algorithm*. Say that we could efficiently calculate the following quantities for any x of length n, for any $j \in 1 \ldots n$, and for any $p \in 1 \ldots (N-1)$ and $q \in 1 \ldots N$:

$$P(y_j = p, y_{j+1} = q|x, \Theta) = \sum_{y: y_j = p, y_{j+1} = q} P(y|x, \Theta) \tag{2}$$

This is the conditional probability of being in state p at time j, and at state q at time $(j+1)$, given an input x and some parameter settings Θ. It involves a summation over all possible state sequences with $y_j = p$

Inputs: A sample of m points, $x^1, x^2, \ldots, x^m$. A model $P(x, y|\Theta)$ which takes the following form:

$$P(x, y|\Theta) = \prod_{r=1\ldots|\Theta|} \Theta_r^{Count(x,y,r)}$$

Goal: To find the maximum-likelihood estimates,

$$\Theta_{ML} = \arg\max_{\Theta} L(\Theta) = \arg\max_{\Theta} \sum_{i=1}^{m} \log \sum_{y} P(x^i, y|\Theta)$$

Initialization: Choose some initial value for the parameters, call this Θ^0.

Algorithm: For $t = 1 \ldots T$,

- For $r = 1 \ldots |\Theta|$, set $\overline{Count}(r) = 0$
- For $i = 1 \ldots m$,
  - For all y, calculate $t_y = P(x^i, y|\Theta^{t-1})$
  - Set $sum = \sum_y t_y$
  - For all y, set $u_y = t_y / sum$ (note that $u_y = P(y|x^i, \Theta^{t-1})$)
  - For all $r = 1 \ldots |\Theta|$, set

    $$\overline{Count}(r) = \overline{Count}(r) + \sum_y u_y Count(x^i, y, r)$$
- For all $r = 1 \ldots |\Theta|$, set

  $$\Theta_r^t = \frac{\overline{Count}(r)}{Z}$$

  where Z is a normalization constant that ensures that the multinomial distribution of which Θ_r^t is a member sums to 1.

Output: Return parameter values Θ^T

Figure 1: The EM Algorithm for PM Models

and $y_{j+1} = q$. Say we could also efficiently compute the following quantity for any x of length n, and any $j \in 1 \ldots n$ and $p \in 1 \ldots (N-1)$:

$$P(y_j = p|x, \Theta) = \sum_{y: y_j = p} P(y|x, \Theta) \tag{3}$$

This is the probability of being in state p at time j, given some input x and parameter settings Θ.

Recall that in the EM algorithm, in order to re-estimate transition parameters, we needed to calculate expected counts defined as the following for any $p \in 1 \ldots N-1$ and $q \in 1 \ldots N$

$$\overline{Count}(p \to q) \quad = \quad \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, p \to q)$$

The inner sum can now be re-written using terms such as that in Eq. 2, as

$$\sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, p \to q) = \sum_{j=1}^{n_i} P(y_j = p, y_{j+1} = q|x, \Theta^{t-1})$$

Similarly, suppose we need to calculate estimated counts corresponding to initial state parameters. We will write $s_1 = p$ to denote the initial state being state p. Then we need to calculate

$$\overline{Count}(s_1 = p) \quad = \quad \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, s_1 = p)$$

for any $p \in 1 \ldots N$. In this case the inner sum can be re-written in terms of the formula in Eq. 3, as

$$\sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, s_1 = p) = P(y_1 = p|x^i, \Theta^{t-1})$$

Finally, suppose we need to calculate estimated counts corresponding to emission parameters. We will write $p \uparrow o$ to denote state p emitting the symbol o. Then we need to calculate

$$\overline{Count}(p \uparrow o) \quad = \quad \sum_{i=1}^{m} \sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, p \uparrow o)$$

for any $p \in 1 \ldots (N-1)$. In this case the inner sum can be re-written in terms of the formula in Eq. 3, as

$$\sum_{y} P(y|x^i, \Theta^{t-1}) Count(x^i, y, p \uparrow o) = \sum_{j: x_j = o} P(y_j = p|x^i, \Theta^{t-1})$$

In summary, if we can calculate the quantities in Equations 2 and 3, then we can calculate all expected counts required in the EM algorithm for HMMs.

5.2 The Algorithm

We will now describe how to calculate the quantities in Eq. 2 and Eq. 3 using the forward–backward algorithm.

Given an input sequence $x_1 \ldots x_n$, we will define the *forward probabilities* as being

$$\alpha_p(j) \quad = \quad P(x_1 \ldots x_{j-1}, y_j = p \mid \Theta)$$

for all $j \in 1 \ldots n$, for all $p \in 1 \ldots N-1$. The forward probability $\alpha_p(j)$ is then the probability of the HMM emitting the output symbols $x_1 \ldots x_{j-1}$, and then ending up in state p. Note that this term involves a summation over all possible state sequences underlying $x_1 \ldots x_{j-1}$.

Given an input sequence $x_1 \ldots x_n$, we will define the *backward probabilities* as being

$$\beta_p(j) \quad = \quad P(x_j \ldots x_n \mid y_j = p, \Theta)$$

for all $j \in 1 \ldots n$, for all $p \in 1 \ldots N-1$. This is the probability of emitting symbols $x_j \ldots x_n$, then ending up in the final state, given that we begin in state p.

The forward and backward probabilities can be calculated efficiently using the recursive definitions in figure 2. We will give more justification for these definitions in the next section.

Given the forward and backward probabilities, the first thing we can calculate is the following:

$$Z = P(x_1, x_2, \ldots x_n | \Theta) = \sum_p \alpha_p(j)\beta_p(j)$$

for any $j \in 1 \ldots n$. Thus we can calculate the probability of the sequence $x_1, x_2, \ldots x_n$ being emitted by the HMM.

We can also calculate the probability of state p underlying observation x_j, one of the quantities introduced in the previous section:

$$P(y_j = p | x, \Theta) = \frac{\alpha_p(j)\beta_p(j)}{Z}$$

for any p, j. Finally, we can calculate the probability of each possible state transition, as follows:

$$P(y_j = p, y_{j+1} = q | x, \Theta) = \frac{\alpha_p(j) a_{p,q} b_p(o_j) \beta_q(j+1)}{Z}$$

for any p, q, j.

5.3 Justification for the Algorithm

To understand the recursive definitions for the forward and backward probabilities, we will make use of a particular directed graph. The graph is associated with a particular input sequence $x_1, x_2, \ldots x_n$, and parameter vector Θ, and has the following vertices:

- A “source” vertex, which we will label s.
- A “final” vertex, which we will label f.
- For all $j \in 1 \ldots n$, for all $p \in 1 \ldots N-1$, there is an associated vertex which we will label $\langle j, p \rangle$.

Given this set of vertices, we define the following directed edges between pairs of vertices (note that each edge has a an associated weight, or probability):

- There is an edge from s to each vertex $\langle 1, p\rangle$ for $p = 1 \ldots N-1$. Each such edge has a weight equal to π_p.
- For any $j \in 1 \ldots n-1$, and $p, q \in 1 \ldots N-1$, there is an edge from vertex $\langle j, p\rangle$ to vertex $\langle (j+1), q\rangle$. This edge has weight equal to $a_{p,q}\ b_p(x_j)$.
- There is an edge from each vertex $\langle n, p\rangle$ for $p = 1 \ldots N-1$ to the final vertex f. Each such edge has a weight equal to $a_{p,N}\ b_p(x_n)$

The resulting graph has a large number of paths from the source s to the final state f; each path goes through a number of intermediate vertices. The weight of an entire path will be taken as the product of weights on the edges in the path. You should be able to convince yourself that:

1. For every state sequence $y_1, y_2, \ldots y_n$ in the original HMM, there is a path through with graph that has the sequence of states $s, \langle 1, y_1\rangle, \ldots, \langle n, y_n\rangle, f$
2. The path associated with state sequence $y_1, y_2, \ldots y_n$ has weight equal to $P(x, y|\Theta)$

We can now interpret the forward and backward probabilities as following:

- $\alpha_p(j)$ is the sum of weights of all paths from s to the state $\langle j, p\rangle$
- $\beta_p(j)$ is the sum of weights of all paths from state $\langle j, p\rangle$ to the final state f

If you construct this graph, you should be able to convince yourself that the recursive definitions for the forward and backward probabilities are correct.

- Given an input sequence $x_1 \ldots x_n$, for any $p \in 1 \ldots N$, $j \in 1 \ldots n$,

$$\alpha_p(j) \quad = \quad P(x_1 \ldots x_{j-1}, y_j = p \mid \Theta) \quad \text{forward probabilities}$$

- **Base case:**

$$\alpha_p(1) = \pi_p \quad \text{for all } p$$

- **Recursive case:**

$$\alpha_p(j+1) = \sum_q \alpha_q(j) a_{q,p} b_q(x_j) \quad \text{for all } p = 1 \ldots N-1 \text{ and } j = 1 \ldots n-1$$

- Given an input sequence $x_1 \ldots x_n$:

$$\beta_p(j) \quad = \quad P(x_j \ldots x_n \mid y_j = p, \Theta) \quad \text{backward probabilities}$$

- **Base case:**

$$\beta_p(n) = a_{p,N} b_p(x_n) \quad \text{for all } p = 1 \ldots N-1$$

- **Recursive case:**

$$\beta_p(j) = \sum_q a_{p,q} b_p(x_j) \beta_q(j+1) \quad \text{for all } p = 1 \ldots N-1 \text{ and } j = 1 \ldots n-1$$

Figure 2: Recursive definitions of the forward and backward probabilities