6.864: Lecture 6 (September 27th, 2005)

The EM Algorithm Part II

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.

- 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.
  - $a_{j,k}$ for $j = 1 \ldots (N-1)$, $k = 1 \ldots N$, is the probability of transitioning from state j to state k.
  - $b_j(o)$ for $j = 1 \ldots (N-1)$, and $o \in \Sigma$, is the probability of emitting symbol o from state j.

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

- Note that we have the following constraints:
  - $\sum_{j=1}^{N} \pi_j = 1$.
  - for all j, $\sum_{k=1}^{N} a_{j,k} = 1$.
  - for all j, $\sum_{o \in \Sigma} b_j(o) = 1$.

Hidden Markov Models

- 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.
- E.g., 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,

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

Hidden Markov Models

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

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

EM: the Basic 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 $P(x, y|\Theta)$: A function that for any x, y, Θ triple returns a probability, which is the probability of seeing x and y together given parameter settings Θ.
- This model defines a *joint* distribution over x and y, but that we can also derive a *marginal* distribution over x alone, defined as

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

- 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 $\sum_{j=1}^{N} \pi_j = 1$, for all $j = 1 \ldots (N-1)$, $\sum_{k=1}^{N} a_{j,k} = 1$, and for all $j = 1 \ldots (N-1)$, $\sum_{o \in \Sigma} b_j(o) = 1$.

Products of Multinomial (PM) Models

- 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

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

Products of Multinomial (PM) Models

- 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:

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

Products of Multinomial (PM) Models

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

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

  Here:

  - Θ_r for $r = 1 \ldots |\Theta|$ is the r'th parameter in the model
  - $Count(x, y, r)$ 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 this form as a *product of multinomials* (PM) model.

The EM Algorithm for PM Models

- 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.
- Crucial detail: deriving Θ^t from Θ^{t-1}

The EM Algorithm for PM Models: Step 1

- At each iteration of the algorithm, two steps are taken.
- **Step 1:** *expected counts* $\overline{Count}(r)$ are calculated for each parameter Θ_r in the model.

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

The EM Algorithm for PM Models: Step 1

- 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 at the t'th iteration,

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

The EM Algorithm for PM Models: Step 2

- **Step 2:** Calculate the updated parameters Θ^t. 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.

The EM Algorithm for HMMs: Step 1

- Define $Count(x^i, y, p \rightarrow q)$ to be the number of times a transition from state p to state q is seen in y.
- **Step 1:** Calculate expected counts such as

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

- (Note: similar counts will be calculated for emission and initial-state parameters)

The EM Algorithm for HMMs: Step 2

- **Step 2:** Re-estimate transition parameter 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.

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

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

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

The Forward-Backward Algorithm for HMMs

- Define $Count(x^i, y, p \rightarrow q)$ to be the number of times a transition from state p to state q is seen in y.
- **Step 1:** Calculate expected counts such as

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

- A problem: the inner sum

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

The Forward-Backward Algorithm for HMMs

- 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 $q \in 1 \ldots N$:

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

- The inner sum can now be re-written using terms such as this:

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

The Forward Probabilities

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 Probabilities

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 $p \in 1 \ldots N - 1$.

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 calculate the probability of being in any state at any position:

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

for any p, j.

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.

- 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:**

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

Justification for the Algorithm

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 $\langle j, p \rangle$.

Justification for the Algorithm

Given this set of vertices, we define the following directed edges:

- 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 $a_{p,N}\ b_p(x_n)$

Justification for the Algorithm

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 $P(x, y|\Theta)$

Justification for the Algorithm

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

Another Application of EM in NLP: Topic Modeling

- Say we have a collection of m documents
- Each document x^i for $i = 1 \ldots m$ is a sequence of words $x_1^i, x_2^i, \ldots x_n^i$
- E.g., we might have a few thousand articles from the New York Times

Another Application of EM in NLP: Topic Modeling

- We'll assume that y^i for $i = 1 \ldots m$ is a **hidden** "topic variable". y^i can take on any of the values $1, 2, \ldots K$

- For any document x^i, and topic variable y, we write

$$P(x^i, y|\Theta) = P(y) \prod_{j=1}^{n} P(x^i_j|y)$$

- Θ contains two types of parameters:
  - $P(y)$ for $y \in 1 \ldots K$ is the probability of selecting topic y
  - $P(w|y)$ for $y \in 1 \ldots K$, w in some vocabulary of possible words, is the probability of generating the word w given topic y

- As before, we can use EM to find

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

- Result: for each of the K topics, we have a different distribution over words, $P(w|y)$

Results from Hofmann, SIGIR 1999

- Applied the method to 15,000 documents, using $k = 128$ topics

- Examples of 6 topics (in each case, table shows the 10 words for which $P(w|y)$ is maximized):

plane, airport, crash, flight, safety, aircraft, air, passenger, board, airline
space, shuttle, mission, astronauts, launch, station, crew, nasa, satellite, earth
home, family, like, love, kids, mother, life, happy, friends, cnn
fi lm, movie, music, new, best, hollywood, love, actor, entertainment, star
un, bosnian, serbs, bosnia, serb, sarajevo, nato, peacekeepers, nations, peace
refugees, aid, rwanda, relief, people, camps, zaire, camp, food, rwandan

Another Application of EM in NLP: Word Clustering

- Say we have a collection of m bigrams
- Each bigram consists of a word pair w_1^i, w_2^i where w_2^i follows w_1^i
- We'd like to build a model of $P(w_2|w_1)$

Another Application of EM in NLP: Word Clustering

- We'll assume that y^i for $i = 1 \ldots m$ is a **hidden** "cluster variable". y^i can take on any of the values $1, 2, \ldots K$

- For any bigram w_1^i, w_2^i, we write

$$P(w_2^i | w_1^i, \Theta) = \sum_y P(w_2^i | y) P(y | w_1^i)$$

- Θ contains two types of parameters:
  - $P(y|w_1)$ for $y \in 1 \ldots K$ is the probability of selecting cluster y given that the first word in the bigram is w_1
  - $P(w_2|y)$ for $y \in 1 \ldots K$, w_2 in some vocabulary of possible words, is the probability of selecting w_2, given cluster y

- As before, we can use EM to find

$$\Theta_{ML} = \arg\max_{\Theta} L(\Theta) = \arg\max_{\Theta} \sum_i \log \sum_y P(w_2^i|y)P(y|w_1^i)$$

- Result: for each of the K clusters, we have the distributions $P(w_2|y)$ and $P(y|w_1)$

- See Saul and Pereira, 1997, for more details