Lecture 10: The EM Algorithm: Training Models with Hidden Data

Earlier in this course we considered a range of techniques for estimating probability distributions from data. In these cases, the training data always included all the information we needed to compute the probabilities. For instance, if we were training a tagging model, we had data that was tagged with the correct answers. In many applications, however, the data we have is missing critical information, which is called the **hidden data**. Today we'll explore a general technique for estimating distributions in which some of the key information is hidden. While the exact nature of the algorithm will vary from application to application, the general technique is called the EM algorithm, which stands for expectation maximization.

One of the key ideas of any EM algorithm is that you start with some distribution of the data, and perform an iteration step that produces a new distribution that is either better or the same as the initial one (in the sense that it increases the probability of the training corpus). This does not mean that the algorithm could eventually produce the optimal distribution, It is what is often called a **hill climbing** algorithm, and it may converge on some local maximum that is not the best one that could be found. In addition, we must remember that optimizing the probability of the training corpus is never a goal in itself. We actually want model that perform well on new data. So we must be careful not to **overtrain** the model by running too many iterations.

1. An Intuitive EM Algorithm for HMMs

The general problem of training HMMs is that we don't know which states correspond to which output, thus we cannot do a simple counting estimate. As an example, consider the HMM λ that models a simplified 2 bucket version of the ball game from a couple of assignments ago. There are two output producing nodes, S1 and S2, and the output vocabulary is R,W, and B (red, white and blue balls). The possible non-zero arcs are shown in Figure 1.

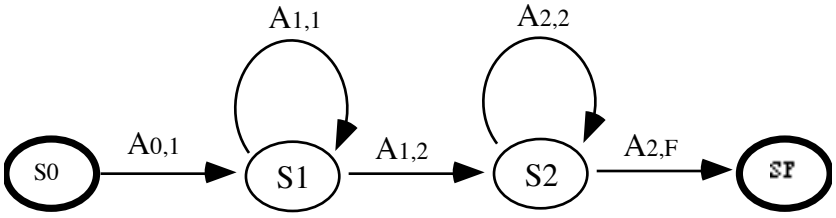

Figure 1: An example HMM for Training

Say the observation sequence we want to model is RWBB. We want to find values for the transition probabilities and output probabilities so as to maximize Prob(RWBB | λ). If we had a tagged corpus, we could estimate these probabilities by simple counting on the training corpus. But in this case we don't know what the state sequence was. In fact, there are three sequences that could have produced RWBB with the HMM in Figure 1.

P1: S0-S1-S1-S1-S2-SF

P2: S0-S1-S1-S2-S2-SF

P3: S0-S1-S2-S2-S2-SF

We are going to train the HMM by adapting the counting technique to do a weighted count over all possible sequences. To do this, we need some initial values for the transition and output probabilities so we can compute an initial probability for each of the possible sequence. Lets just assume everything is equally likely, so the probability of $A_{1,1}$ and $A_{1,2}$ are both .5, and the output probability distribution for each node assigns each symbol a probability of 1/3. These values are shown in Figure 2.

Transition Pr	S 1	S 2	S F
S 0	1.000	0.000	0.000
S 1	0.5	0.5	0
S 2	0	0.5	0.5
Output Pr	R	W	B
S 1	0.33	0.33	0.33
S 2	0.33	0.33	0.33

Figure 2: The initial guess at the HMM parameters

Given this HMM, we see that the probability of each of the three paths is equally likely. To see this, consider the probability observing RWBB from path P1 is

Prob(RWBB & P1)

$$= A_{0,1} * Pr(R|S1) * A_{1,1} * Pr(W|S1) * A_{1,1} * Pr(B|S1) * A_{1,2} * Pr(B|S2) * A_{2,F}$$

$$= 7.4 * 10^{-4}$$

The probability of RWBB for the other paths will be the same value since the transition and output probabilities are identical for each arc and node in the network. Given the probability of observing the sequence for each path, we can compute the probability of each path given the observation sequence:

$$Prob(P1 \mid RWBB)$$

$$= Prob(P1 \ \& \ RWBB) / Prob(RWBB)$$

We saw how to compute the numerator above. How do we find the value of Prob(RWBB) given the HMM? Well, there are only three paths that can generate this sequence, so

$$Prob(RWBB) = \sum_i Prob(P_i \ \& \ RWBB)$$

In our example above, $Prob(RWBB) = 7.4 * 10^{-4} + 7.4 * 10^{-4} + 7.4 * 10^{-4} = 2.22 * 10^{-3}$. Combining these two, we get

$$Prob(Pi \mid RWBB) = \frac{Prob(Pi \ \& \ RWBB)}{\sum_j Prob(Pj \ \& \ RWBB)}$$

With our initial parameters, the formula gives a probability of 1/3 to each path.

We can now do our counting for each path. Figure 3 show the counts of each transition for each path, and then the weighted sum. For instance, the sum for S1-S1 includes 2 from path P1, weighted at .333, giving .666, and one from path P2 with weight .333 again, giving .333, producing a total of 1. Due to the structure of the network and the fact that each path is equally likely, each of these sums comes to one this time. When we use these counts to compute the new transition probabilities,we find they haven't changed this time. For instance, there is a weighted

sum of 2 transitions from S1, thus the Prob(S1-S1 being taken from S1) = 1/2 = .5 and Prob(S1-S2 being taken from S1) = 1/2 = .5, both as before.

Path	Prob	Nodes	S0-S1	S1-S1	S1-S2	S2-S2	S2-SF
P1	0.333	S0-S1-S1-S1-S2-SF	1	2	1	0	1
P2	0.333	S0-S1-S1-S2-S2-SF	1	1	1	1	1
P3	0.333	S0-S1-S2-S2-S2-SF	1	0	1	2	1
Weighted Count			1	1	1	1	1

Figure 3: Computing the Weighted sum of transitions

For the output probabilities, we can obtain more interesting data as shown in Figure 4.

Path	Prob	Nodes	R/S1	W/S1	B/S1	R/S2	W/S2	B/S2
P1	0.333	S0-S1-S1-S1-S2-SF	1	1	1	0	0	1
P2	0.333	S0-S1-S1-S2-S2-SF	1	1	0	0	0	2
P3	0.333	S0-S1-S2-S2-S2-SF	1	0	0	0	1	2
Weighted Count			1.000	0.667	0.333	0.000	0.333	1.667

Figure 4: Computing the weighted sum of outputs from each node

Using the counts in Figure 4, we can compute the new output probabilities using

$$\text{Prob}(R \mid S1) = \text{Count}(R, S1) / \text{Count}(S1)$$

The weighted count for node S1 over all output symbols is 2. Thus we have

$$\text{Prob}(R \mid S1) = 1/2 = .5,\ \text{Prob}(W \mid S1) = .667 / 2 = .333 \text{ and } \text{Prob}(B \mid S1) = .333/2 = .167$$

Putting it all together, we have a new set of parameters as shown in Figure 5.

Transition Pr	S1	S2	SF
S0	1.000	0.000	0.000
S1	0.5	0.5	0
S2	0	0.5	0.5
Output Pr	**R**	**W**	**B**
S1	0.5	0.333	0.167
S2	0	0.167	0.833

Figure 5: Revised set of HMM parameters

Note that the probability of RWBB given this new set of parameters, computed using the formula

$$\text{Prob}(\text{RWBB}) = \sum_i \text{Prob}(P_i \,\&\, \text{RWBB})$$

is $1.2*10^{-2}$, considerably higher than the $2.22*10^{-3}$ with the original parameters. Baum (1972) proved that by performing this reestimation procedure on an HMM, the new parameters will yield a higher or equal probability to the sequence than the original parameters. We use the reestimation method repeatedly until the probability of the observation doesn't improve significantly.

If we repeat this process another time using the new parameters, we get the following probabilities for each path:

Prob(P1 & RWBB) = $1.45 * 10^{-3}$

Prob(P2 & RWBB) = $7.22 * 10^{-3}$

Prob(P3 & RWBB) = $3.62 * 10^{-3}$

With these weights, the total is $1.23*10^{-2}$. From this, we can compute the path probabilities and obtain the new counts shown in Figure 6.

Path	Prob	Nodes	S0-S1	S1-S1	S1-S2	S2-S2	S2-SF
P1	0.118	S0-S1-S1-S1-S2-SF	1	2	1	0	1
P2	0.587	S0-S1-S1-S2-S2-SF	1	1	1	1	1
P3	0.294	S0-S1-S2-S2-S2-SF	1	0	1	2	1
Weighted Count			1	0.823	1	1.175	1

Path	Prob	Nodes	RIS1	WIS1	BIS1	RIS2	WIS2	BIS2
P1	0.118	S0-S1-S1-S1-S2-SF	1	1	1	0	0	1
P2	0.587	S0-S1-S1-S2-S2-SF	1	1	0	0	0	2
P3	0.294	S0-S1-S2-S2-S2-SF	1	0	0	0	1	2
Weighted Count			1.000	0.705	0.118	0.000	0.294	1.880

Figure 6: The new weighted counts for the second iteration

This produces the new estimates of the parameters shown in the column for the second iteration shown in Figure 7. The probability of the output sequence given these new parameters is $1.45 * 10^{-2}$, another improvement. In fact, if we decide to continue the iterations until there is no change in the 10^{-5} digit, we will continue for eight iterations as shown in Figure 7. Note that the main change in the iterations is that the probability of S1 outputting B is slowly sinking to zero. If we are solely interested in optimizing the probability of RWBB then this continues to improve the score. But if we are using this to estimate parameters for a recognition system, we may not want to continue for too long or we will **overtrain** on this input and assign poor scores to slight variants (say an observation that starts with a B).

Iteratio	0	1	2	3	4	5	6	7	8
Transition Probabilities									
S1-S1	0.500	0.500	0.452	0.432	0.424	0.420	0.419	0.418	0.417
S1-S2	0.500	0.500	0.548	0.568	0.576	0.580	0.581	0.582	0.583
S2-S2	0.500	0.500	0.541	0.554	0.558	0.560	0.561	0.562	0.563
S2-SF	0.500	0.500	0.459	0.446	0.442	0.440	0.439	0.438	0.437
Output Probabilities									
S1/R	0.33	0.5	0.548	0.568	0.576	0.579	0.581	0.582	0.583
S1/W	0.33	0.33	0.387	0.407	0.415	0.417	0.417	0.417	0.416
S1/B	0.33	0.167	0.0645	0.024	0.008	0.003	0.001	0.0004	0.0001
S2/R	0.33	0	0	0	0	0	0	0	0
S2/W	0.33	0.167	0.135	0.126	0.123	0.123	0.123	0.124	0.125
S2/B	0.33	0.833	0.865	0.874	0.876	0.876	0.876	0.876	0.875
Probability of Observation									
	0.00222	0.01230	0.01446	0.01521	0.01549	0.01558	0.01562	0.01563	0.01563

Figure 7: A Summary of Eight Iterations of the Training Algorithm