SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

01.112 Machine Learning, Fall 2018
Lecture Notes for Week 10

16. Hidden Markov Models (III)

Last update: Thursday 8th November, 2018 21:37

1 Unsupervised Learning

We have already discussed how to do supervised learning of parameters for HMM in week 8. The procedure for learning HMMs with supervision was very simple and straightforward. However, what if we would like to learn the HMM in an unsupervised manner? In other words, what if we are only given the unlabelled observation sequences $\mathbf{x}$s and a collection of tags but not the tag sequences $\mathbf{y}$s for each instance during the training phase?

When we no longer have the tags, we must resort to other ways of estimating the HMMs. It is not trivial to construct a model that agrees with the observations except in very simple scenarios. Here is one:

- We have an HMM with states $\mathcal{Y} = \{A, B, C\}, \Sigma = \{a, b, c\}$
- We see the following output sequences in training data: $(a, b), (a, c), (a, b)$

Discussions How would you choose the parameter values for $a_{i,j}$ and $b_i(o)$? Can you think of a reasonable choice?

Certainly we should not estimate the parameters by manually designing one. In general, how do we learn the parameters? What would be the algorithm to use if we want to learn the model parameters in an unsupervised manner? Yes, we can use EM!

1. Hard EM

One simple approach to learning the hidden Markov model in an unsupervised manner is the so-called *hard-EM* approach. Different from standard (soft-)EM, the hard EM runs in a way that is very similar to k-means for clustering. Suppose now that we have multiple observed sequences of outputs (no observed tags). We will denote these sequences with superscripts, *i.e.*, $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$. Unlike soft-EM, at the E-step of the hard-EM algorithm each data point $\mathbf{x}$ is strictly assigned to a single $\mathbf{y}$ (*i.e.*, there is no partial membership). In HMM, this means at the E-step each observation sequence (word sequence) is strictly assigned a single state sequence (tag sequence). Remember at the E-step, we are given model parameters and the observation sequences, and would need to find for each observation sequence the corresponding state sequence. How do

we figure out the most probable state sequence for a given observation sequence, if we know the model's parameters? The answer is to use the Viterbi algorithm. This leads to the following E-step for our hard-EM algorithm for HMM:

E-step

For each observation sequence $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$, we use the Viterbi algorithm to find the most probable state sequence $\mathbf{y}^{(i)^*} = (y_1^{(i)^*}, y_2^{(i)^*}, \ldots, y_n^{(i)^*})$. Next, we can easily compute the following counts from our data set (since now the dataset becomes fully annotated after running the Viterbi algorithm):

$$\text{Count}(u; v), \text{Count}(u)$$

for any $u, v \in \mathcal{Y} = \{0, 1, 2, \ldots, N, N+1\}$, and

$$\text{Count}(u \to o), \text{Count}(u)$$

for each $u \in \mathcal{Y}, o \in \Sigma$.

M-step

The M-step is essentially the same as what we do for parameter estimation using MLE when we have labeled data (in this case the labels are predicted from the E-step above).

$$a_{u,v} = \frac{\text{Count}(u; v)}{\text{Count}(u)}$$

$$b_u(o) = \frac{\text{Count}(u \to o)}{\text{Count}(u)}$$

Okay. That's the hard-EM algorithm. Let us move to the discussions on the soft-EM next.

2. (Soft) EM

For the soft EM, in the context of each sequence, we must evaluate a posterior probability over possible tag sequences. For estimation, we only need expected counts that are used in the re-estimation step (M-step). To this end, let $\text{Count}(\mathbf{x}^{(i)}, \mathbf{y}, u \to v)$ be the number of times a transition from state u to state v occurs in a tag sequence $\mathbf{y}$ corresponding to observation sequence $\mathbf{x}^{(i)}$. We now discuss how to do unsupervised learning of an HMM using the EM algorithm. We will only show here the derivations for transition probabilities probabilities for now.

E-step

Re-assign the partial memberships for each $\mathbf{x}^{(i)}$. In this case we would like to calculate *expected* counts, added across sequences:

$$\text{Count}(u; v) = \sum_{i=1}^{m} \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(i)}) \text{Count}(\mathbf{x}^{(i)}, \mathbf{y}, u \to v) \tag{1}$$

M-step

Re-estimate transition probabilities based on the expected counts:

$$a_{u,v} = \frac{\text{Count}(u; v)}{\sum_{v'} \text{Count}(u; v')} \tag{2}$$

where the denominator ensures that

$$\sum_{v'=1}^{N+1} a_{u,v'} = 1$$

The main problem in running the EM algorithm is calculating the sum over the possible tag sequences in the E-step:

$$\sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(i)}) \text{Count}(\mathbf{x}^{(i)}, \mathbf{y}, u \to v) \tag{3}$$

The sum is over an exponential number of possible hidden state sequences $\mathbf{y}$. How do we calculate such terms efficiently?

Suppose we could efficiently calculate marginal posterior probabilities

$$p(y_j = u, y_{j+1} = v|\mathbf{x}; \theta) = \sum_{\mathbf{y}: y_j = u, y_{j+1} = v} p(\mathbf{y}|\mathbf{x}; \theta) \tag{4}$$

for any $u, v \in 0, \ldots, N+1, j \in 1, \ldots, n$. These are the posterior probabilities that the state in position j was u and we transitioned into v at the next step. The probability is conditioned on the observed sequence $\mathbf{x}$ and the current setting of the model parameters θ. Now, under this assumption, we could rewrite Equation (3) as:

$$\sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(i)}; \theta) \text{Count}(\mathbf{x}^{(i)}, \mathbf{y}, u \to v) = \sum_{j=0}^{n} p(y_j = u, y_{j+1} = v|\mathbf{x}^{(i)}; \theta) \tag{5}$$

The key remaining question is how to calculate these posterior marginals efficiently. In other words, our goal is to evaluate $p(y_j = u, y_{j+1} = v|\mathbf{x}^{(i)}; \theta)$.

2 Inference

Let us start with a simpler question. How do we efficiently compute the following marginal posterior probabilities?

$$p(y_j = u|\mathbf{x};\theta) = \sum_{\mathbf{y}:y_j=u} p(\mathbf{y}|\mathbf{x};\theta) \tag{6}$$

for any $u \in 1,\ldots,N, j \in 1,\ldots,n$. These are the posterior probabilities that the state in position j is u. The probability is conditioned on the observed sequence $\mathbf{x}$ and the current setting of the model parameters θ.

$$p(y_j = u|\mathbf{x};\theta) = \frac{p(y_j = u, \mathbf{x};\theta)}{p(\mathbf{x};\theta)} = \frac{p(x_1, x_2, \ldots, x_{j-1}, y_j = u, x_j, x_{j+1}, \ldots, x_n;\theta)}{p(x_1, x_2, \ldots, x_n;\theta)} \tag{7}$$

Now let's look at the joint probability at the numerator. It can be rewritten as follows:

$$p(x_1, x_2, \ldots, x_{j-1}, y_j = u, x_j, x_{j+1}, \ldots, x_n;\theta) \tag{8}$$

$$= p(x_1, x_2, \ldots, x_{j-1}, y_j = u;\theta) \cdot p(x_j, x_{j+1}, \ldots, x_n|x_1, x_2, \ldots, x_{j-1}, y_j = u;\theta) \tag{9}$$

However, in HMM, given y_j, we can observe that $x_j, x_{j+1}, \ldots, x_n$ are conditionally independent of $x_1, x_2, \ldots, x_{j-1}$. Thus we have:

$$p(x_1, x_2, \ldots, x_{j-1}, y_j = u, x_j, x_{j+1}, \ldots, x_n;\theta) \tag{10}$$

$$= p(x_1, x_2, \ldots, x_{j-1}, y_j = u;\theta) \cdot p(x_j, x_{j+1}, \ldots, x_n|y_j = u;\theta) \tag{11}$$

Let us define the following forward probabilities:

$$\alpha_u(j) = p(x_1, x_2, \ldots, x_{j-1}, y_j = u;\theta) \tag{12}$$

$\alpha_u(j)$ is the probability of emitting the symbols $x_1, \ldots, x_{j-1}$ and ending in state u in position j without (yet) emitting the corresponding output symbol. These are analogous to the $\pi(k, v)$ probabilities in the Viterbi algorithm with the exception that $\pi(k, v)$ included generating the corresponding observation in position k. Note that, unlike before, we are summing over all the possible sequences of states that could give rise to the observations $x_1, \ldots, x_{j-1}$. In the Viterbi algorithm, we maximized over the tag sequences.

Similarly, we define the following backward probabilities:

$$\beta_u(j) = p(x_j, x_{j+1}, \ldots, x_n|y_j = u;\theta) \tag{13}$$

We have:

$$p(x_1, x_2, \ldots, x_{j-1}, y_j = u, x_j, x_{j+1}, \ldots, x_n;\theta) = \alpha_u(j) \cdot \beta_u(j) \tag{14}$$

for all $j \in 1,\ldots,n$, for all $u \in 1,\ldots,N$. $\beta_u(j)$ is the probability of emitting symbols $x_j, \ldots, x_n$ and transitioning into the final (`STOP`) state, given that we begun in state u in position

j. Again, this definition involves summing over all the tag sequences that could have generated the observations from x_j onwards, provided that the tag at j is u.

Why are these two definitions useful? Suppose we had been able to evaluate α and β probabilities *efficiently*. Then the following marginal probability we were after could be calculated as:

$$p(y_j = u, y_{j+1} = v|\mathbf{x}; \theta)$$
$$= \frac{p(x_1, x_2, \ldots, x_{j-1}, y_j = u; \theta) \cdot p(y_{j+1} = v|y_j = u) \cdot p(x_j|y_j = u) \cdot p(x_{j+1}, \ldots, x_n|y_{j+1} = v; \theta)}{p(\mathbf{x}; \theta)}$$
$$= \frac{\alpha_u(j) \cdot a_{u,v} \cdot b_u(x_j) \cdot \beta_v(j+1)}{p(\mathbf{x}; \theta)} \quad (15)$$

Now what about $p(\mathbf{x}; \theta)$?

$$p(\mathbf{x}; \theta) = p(x_1, x_2, \ldots, x_n; \theta) = \sum_u p(x_1, x_2, x_{i-1}, y_i = u, x_i, \ldots, x_n; \theta)$$
$$= \sum_{u'} p(x_1, x_2, \ldots, x_{i-1}, y_i = u'; \theta) \cdot p(x_i, x_{i+1}, \ldots, x_n|y_i = u'; \theta) = \sum_{u'} \alpha_{u'}(i)\beta_{u'}(i) \quad (16)$$

Discussions Interesting! Why does the right hand side involve i, but not the left hand side of the above equation?

Now we arrive at the following:

$$p(y_j = u, y_{j+1} = v|\mathbf{x}; \theta) = \frac{\alpha_u(j) \cdot a_{u,v} \cdot b_u(x_j) \cdot \beta_v(j+1)}{\sum_{u'} \alpha_{u'}(i)\beta_{u'}(i)} \quad (17)$$

for any i.

This is just the sum over all possible tag sequences that include the transition $y_j = u$ and $y_{j+1} = v$ and generates the observations, divided by the sum over all tag sequences that generate the observations. As a result, we obtain the relative probability of the transition, relative to all the alternatives given the observations, i.e., the posterior probability. Note that $\alpha_u(j)$ involves all the summations over tags $y_0, y_1, \ldots, y_{j-1}$, and $\beta_v(j+1)$ involves all the summations over the tags $y_{j+2}, \ldots, y_n, y_{n+1}$.

This means that if we can efficiently compute the α and β terms, the expected counts in Equation (3) can also be computed efficiently. The next question is: how do we compute the forward and backward scores efficiently? We will introduce an efficient dynamic programming algorithm for computing the forward and backward probabilities efficiently in the next class.

Learning Objectives

You need to know:

1. How to do unsupervised learning of an HMM using the hard EM

2. What is the difference between the soft and hard EM in unsupervised learning of an HMM
3. What do the forward and backward probabilities stand for
4. How to compute the expected counts using the forward and backward probabilities