Exercises on Hidden Markov Models

Statistical Processing of Natural Language

Data Mining and Knowledge Management

Fall 2011

1 HMM Parameters

Question 1. Compute the parameters of a model

Given the following sequences of pairs `(state,emission)`:

```
(D,the) (N,wine) (V,ages) (A,alone)
(D,the) (N,wine) (N,waits) (V,last) (N,ages)
(D,some) (N,flies) (V,dove) (P,into) (D,the) (N,wine)
(D,the) (N,dove) (V,flies) (P,for) (D,some) (N,flies)
(D,the) (A,last) (N,dove) (V,waits) (A,alone)
```

Question 1.1. Draw the graph of the resulting *bigram* HMM, and list all non-zero model parameters that we can obtain via MLE from this data.

Question 1.2. Draw the graph of the resulting *trigram* HMM, and list all non-zero model parameters that we can obtain via MLE from this data.

Question 1.3. Compute the probability of the following sequence according to each of the two previous models:
`(D,the) (N,dove) (V,waits) (P,for) (DT,some) (A,last) (N,wine)`

2 The Viterbi Algorithm

In this exercise we will modify the Viterbi algorithm used to compute the most likely state sequence in HMMs. Recall that we can specify an HMM model as $\mu = (A, B, \pi)$, where A is a matrix of transition parameters, B is a matrix of emission parameters, and π is the initial state distribution. Let's first recall the computations behind Viterbi. Given an observation sequence $O = O_1 \dots O_T$ the algorithm computes:

$$\begin{aligned}\operatorname*{argmax}_{X=X_1\dots X_T} P_\mu(X \mid O) &= \operatorname*{argmax}_{X} \frac{P_\mu(X, O)}{P_\mu(O)} \\ &= \operatorname*{argmax}_{X} P_\mu(X, O)\end{aligned}$$

The algorithm proceeds by defining quantities $\delta_j(t)$, which keep track of the most likely way of being at state X_j after emmitting $O_1 \dots O_t$. In parallel, the algorithm also computes variable $\psi_j(t)$, which store the transitions that lead to the most likely state sequence. The computations are as follows:

1. Initialization: $\forall j = 1 \dots N$
   $\delta_j(1) = \pi_j b_{j,o_1}$
   $\psi_j(1) = 0$

2. Induction: $\forall t : 1 \leq t < T$
   $\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) a_{i,j} b_{j,o_{t+1}} \quad \forall j = 1 \dots N$
   $\psi_j(t+1) = \arg\max_{1 \leq i \leq N} \delta_i(t) a_{i,j} \quad \forall j = 1 \dots N$

3. Termination: backwards path readout.
   - $\hat{X}_T = \arg\max_{1 \leq i \leq N} \delta_i(T)$
   - $\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$
   - $P(\hat{X}) = \max_{1 \leq i \leq N} \delta_i(T)$

Question 2. Viterbi in log-space.

We are now interested in working in the logarithmic space. That is, we want to redefine the computations so that Viterbi computes $\log(P_\mu(X,O))$ instead of $P_\mu(X,O)$. This extension is important in tasks where probabilities of individual emissions or transitions may be very small, such as in NLP applications where the number of symbols is usually very very large. If individual probabilities are small then products of such probabilities will be very very very small. In such cases, our machines may run out of precision, hence yielding unreliable computations. Working in the log space is a common "trick" that solves the problem. We still want the most likely sequence under μ, but we will compute it as:

$$\begin{aligned}
\operatorname*{argmax}_{X=X_1\ldots X_T} P_\mu(X \mid O) &= \operatorname*{argmax}_{X} \log(P_\mu(X \mid O)) \\
&= \operatorname*{argmax}_{X} \log\left(\frac{P_\mu(X,O)}{P_\mu(O)}\right) \\
&= \operatorname*{argmax}_{x} \log(P_\mu(X,O)) - \log(P_\mu(O)) \\
&= \operatorname*{argmax}_{x} \log(P_\mu(X,O))
\end{aligned}$$

First of all, note that even if we compute log probabilities instead of normal probabilities, the most likely sequence will be the same. This is because the log function preserves the value of the state sequence attaining the maximum probability. Second, as before, we can drop the term $\log(P_\mu(O))$ because O is fixed, and it does not affect the maximum.

Let's assume that the HMM is given in the log-space. That is, $\mu' = (A', B', \pi')$ where

- For any states i and j: $a'_{i,j} = \log a_{i,j} = \log P(X_{t+1} = s_j \mid X_t = s_i)$
- For any state i and symbol j: $b'_{i,j} = \log b_{i,j} = \log P(O_t = j \mid X_t = s_i)$
- For any state i : $\pi'_i = \log \pi_i = \log P(X_1 = i)$

Question 2.1. Write $\log(P_\mu(X,O))$ in terms of μ'.

Question 2.2. Rewrite the recursive expressions for δ and ψ to work with log probabilities.

Question 3. Error-augmented Viterbi.

In this question we will modify Viterbi to account for a notion of Hamming error of state sequences. In future lectures we will see applications of this algorithm.

Let X and X' be two state sequences of length T. We define an error function that counts the number of different states (also known as Hamming error):

$$\text{error}(X, X') = \sum_{i=1}^{T} I[X_i \neq X'_i]$$

where the function $I[p]$ is an indicator function that returns 1 if predicate p is true and 0 otherwise. For example, error("*abc*","*acb*") = 2.

For this problem, the input will consist of an observation sequence O together with its *correct* state sequence X^*. The goal is to find the *most-erroneous* sequence under an HMM model specified by μ. That is, we are interested in finding a state sequence that has high probability under our model and also has high error. More formally, we would like to find:

$$\underset{X}{\text{argmax}}\ \log(P(X, O)) + \lambda \cdot \text{error}(X^*, X)$$

where the parameter λ controls the trade-off between the two terms (high values give more importance to the error).

Question 3. Modify the Viterbi algorithm to solve this problem.

HINT. Note that the error function decomposes in a similar fashion to the computations behind an HMM. Then think of the optimality conditions behind the design of Viterbi that allow us to compute δ and ψ recursively.