CSE 620: Fall 2011

Project 7: Hidden Markov Models

Due: 2:30 PM, Dec. 6, 2013

Introduction: You will implement and use the Viterbi algorithm to perform a segmentation analysis of the Bacteriophage lambda, a virus that infects the *e. coli* bacteria. This was one of the first viral genomes to be completely sequenced, and shows some interesting segmentation properties. Specifically, sections of the genome show a very high GC content (that is, a large portion of the bases are Gs and Cs), while others sections show a much smaller GC content. The effect is distinct enough that is almost certainly biologically significant, but finding the exact boarders of the segments can be tricky.

Part 1: Write a Python program that takes, as input, the description of an HMM (in a format of your choice) and a series of observations, and returns the most likely state path through the HMM given that set of observations (i.e. implement the Viterbi algorithm, associated with the δ function in the slides).

Note: you will need to work with *log* probabilities; working directly with probabilities will result in significant rounding error. (That is, everything will round to zero.) Your algorithm should return the best path and the associated *log* probability.

Part 2: Augment your program with *Viterbi* training, which takes an HMM and then adjusts the parameters to better match the provided observation list. Specifically, it takes an HMM with a (very) rough estimates of the the parameters, an observation list, and modifies the parameter estimates to *maximize the Viterbi* score of the sequence. We can formally describe the Viterbi training problem as follows:

- Input:
  - N: the number of states in the HMM.
  - Σ: The set of possible observations.
  - π: The start-state distribution for the HMM.
  - O: A sequence over Σ.
  - a: An NxN matrix containing initial estimates of the transition probabilities.
  - b: An $Nx|\Sigma|$ matrix containing initial estimates of the emission probabilities.
- Output:
  - Updated values for **a** values for **b**.
- Goal: Update the **a** and **b** matrices to maximize $max_{0\leq i<N}\, \delta_i(|O|)$, where δ is the Viterbi function, used on the observation sequence O and the HMM described by $\lambda=(N,M,\pi,a,b)$.

The algorithm works as follows. Given the observation sequence **O** and your current estimates of **a** and **b**:

1. Use the Viterbi algorithm to find the most likely path **Q** of **O** through the HMM. (That is, find $max_{0 \leq i < N}\, \delta_i(|O|)$, and the corresponding path **Q**.)
2. Update the HMM parameters as follows:
   - For each state pair *i* and *j*, update the transition probability from *i* to *j* with the probability T_{ij}/C_i, where T_{ij} is the number of times **Q** transitioned from state *i* to state *j*, and C_i is the number of times **Q** entered state *i.*
   - For state *i* and observation *j*, update the emission probability of state *i* emitting observation *j* to E_{ij}/C_i, where E_{ij} is the number of times that **Q** requires state *i* to emit observation *b* and C_i is as before.

We iterate over these steps, noticing that after updating *a* and *b* in step (2), we will should see a better Viterbi score in step 1. We halt the algorithm when the score fails to improve (or the improvement is to trivial to continue).

Part 3: As stated above, the Bacteriophage lambda virus is notable for its segmentation by GC-content. We would like to find the segments. That is: find those segments of the genome that have higher GC-content values, find the content of the genome that have lower GC-content values, and find the *break points* – the points at which they switch.

Design a *two-state* HMM to identify segments, use the lambda phage genome and Viterbi training to set the parameters[1], and return a list of break points.

Submission:
Python code for two functions:
- The Viterbi algorithm.
- The Viterbi training algorithm.

Two python programs that can be run from the command line:
- The first takes is its command line parameter a file containing an HMM description and a file containing an observation sequence, and prints (to standard output) the best path and associated log probability.
- The second takes a file containing an HMM description and a file containing an observation sequence, and prints a description of the trained HMM in the format you are using to represent HMMs.

Sample input files for each executable that I can modify for testing.

A file called README.txt describing your HMM input format and how to run each executable.

Files containing your initial two-state HMM and your final two-state HMM.

A text file containing the number of break points, and a list of all break coordinates, for the lambda phage genome.

[1] Assume $\pi = (½, ½)$