6.047/6.878 Fall 2008 - Problem Set 3

Due: Monday October 6, 2008 at 8pm

1. **Hidden Markov models and protein structure**

   One biological application of hidden Markov models is to determine the secondary structure (i.e. the general three dimensional shape) of a protein. This general shape is made up of alpha helices, beta sheets, turns and other structures. In this problem we will assume that the amino acid composition of these regions is governed by an HMM.

   In order to keep this problem relatively simple, we do not use actual transition values or emission probabilities. We will use the state transition probabilities of:

	Alpha Helix	Beta Sheet	Turn	Other
Alpha Helix	0.7	0.1	0.2	0.0
Beta Sheet	0.2	0.6	0.0	0.2
Turn	0.3	0.3	0.1	0.3
Other	0.3	0.3	0.0	0.4

   where, for example, $P(\text{Alpha Helix} \rightarrow \text{Turn}) = 0.2$. The start state is always Other and the emission probabilities are:

Amino Acid	Alpha Helix	Beta Sheet	Turn	Other
M	0.35	0.10	0.00	0.05
L	0.30	0.05	0.15	0.15
N	0.15	0.30	0.10	0.20
E	0.10	0.40	0.10	0.15
A	0.05	0.00	0.35	0.20
G	0.05	0.15	0.30	0.25

   (a) Draw this HMM. Include the state transition probabilities and the emission probabilities for each state.

   (b) What is the probability of the second emitted character being an A?

   (c) Give the most likely state transition path for the amino acid sequence $MLAN$ using the Viterbi algorithm. What is $P(X, Y)$ for $X = MLAN$ and $Y =$ (the Viterbi path)? (Include the Viterbi algorithm table in your solution.)

   (d) How many other paths could give rise to the same sequence $MLAN$? Compute the total probability $P(X)$ for $X = MLAN$, defined as $P(X) = \sum_Y P(X, Y)$ (although this is not how you should compute it!). Compare this to $P(X, Y)$ for the Viterbi path in part (c). What does this say about the reliability of the Viterbi path?

2. **Markov chains for CpG classification**

   In this problem, we will simulate and test the use of Markov chains for classifying DNA sequences based on nucleotide frequencies. In these Markov chains, we have four states, one for each nucleotide, and transition probabilities between the states. This is a degenerate case of an HMM in which the state fully determines the emission (thus removing the "hiddenness").

   Page 50 of Durbin gives the following transition probabilities for the models of being inside a CpG island (M1) and outside a CpG island (M2):

M1	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.367	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

M2	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

where, for example, $P_{M1}(\mathrm{A} \rightarrow \mathrm{G}) = 0.426$. We will additionally assume a uniform distribution among nucleotides in the initial state (that is, the chains start in each state with probability 0.25).

Write code to (1) generate random sequences from Markov chains defined by M1 and M2 and (2) find the probability of a given sequence being generated by M1 or M2.

(a) Simulate 10000 sequences of length 6 from M1. Of the sequences simulated, how often is the probability of the sequence being generated by M2 higher than the probability of it being generated by M1?

(b) Now, simulate 10000 sequences of length 6 from M2. Of the sequences simulated, how often is the probability of the sequence being generated by M1 higher than the probability of it being generated by M2?

(c) What do the two parts above tell you about the potential for error when classifying sequences into regions of CpG islands using a simple likelihood estimate?

(d) If you had a prior on how often a sequence was part of a CpG island, how would you use this information to better classify sequences?

In addition to your answers above, include a printout the code you used.

3. **HMMs, state durations, and GENSCAN**

An important use of HMMs is to decode or parse a genome into its biological components, such as exons, introns, CpG islands, etc. In this problem, we will examine how the accuracy of HMM predictions is affected by certain inherent properties of the model.

In lecture, we developed an HMM to detect CpG islands that requires eight states in order to capture the CpG dinucleotide bias. To simplify this problem, instead of searching for CpG islands, we will simply search for High-GC (on average 60% G or C) and Low-GC regions (on average 60% A or T). Thus, our model requires only two states. We have provided a program, `viterbi.py`, which you will complete and use to decode several artificial genomes, and then compare the resulting predictions of High-GC and Low-GC regions to a provided (correct) annotation. More details about this program are included at the end of the problem.

(a) In most HMMs, the self-loop transition probabilities a_{kk} are large, while the transition probabilities between different states a_{kl} are small. Once a Markov chain with these transition probabilities enters state k, it tends to stay in state k for a while. The *state duration* is the total number of steps for which the Markov chain stays in the same state, before switching to another state. What is the expected (mean) state duration of state k as a function of the transition probability a_{kk}? What is the distribution of state durations $P(D_k = d)$?

(b) Complete the implementation of the Viterbi algorithm in `viterbi.py`. Based on the HMM parameters hard-coded into the program, what are the expected state durations for High-GC and Low-GC regions? Apply the finished program to the data file `hmmgen`, which was generated using the same HMM, and verify that your program achieves ~83% accuracy.

(c) Now apply your program to the files `mystery1`, `mystery2`, and `mystery3`. How do the (correct) state duration distributions in the mystery sequences differ and what do they have in common? What accuracy levels does your HMM achieve on these sequences? How does each Viterbi-predicted state duration distribution differ from the correct distribution? (You don't need to include the plots in your solutions.)

(d) Would re-training the HMM parameters according to the procedure described in lecture, using the correct annotations as training data, improve the accuracy of the Viterbi annotation for the mystery sequences? Why or why not?

(optional) Try to make the decoder perform better by adjusting the hard-coded model parameters. If you succeed, can you explain why?

(e) As you are now aware, the length distribution of genomic elements can strongly affect the predictive accuracy of an HMM used to decode them. Unfortunately, most elements in real genomes do not follow the length distribution you derived in part (a). By reading the following paper (or any other sources), describe how the gene finder GENSCAN addresses this issue. How is it possible, algorithmically, to use state duration distributions that differ from the one you derived in part (a)?

Burge C, Karlin S. Prediction of complete gene structures in human genomic DNA. *J Mol Bio* 268(1):78-94, 1997.

Details about `viterbi.py`

The nearly complete program `viterbi.py` performs the following:

- Reads in a data file containing a DNA sequence and an authoritative (correct) annotation, consisting of a string of pluses and minuses, specifying where the High-GC and Low-GC regions are, respectively.
- Calculates the base composition of the High-GC and Low-GC regions, calculates the mean length of High-GC and Low-GC regions, and plots a histogram of the lengths of the High-GC and Low-GC regions. (All with respect to the authoritative annotation.)
- Performs Viterbi decoding on the DNA sequence, using a hard-coded HMM designed to detect High-GC and Low-GC regions. (This is the part you will complete.)
- Calculates the base composition of the High-GC and Low-GC regions, calculates the mean length of High-GC and Low-GC regions, and plots a histogram of the lengths of the High-GC and Low-GC regions. (All with respect to the Viterbi annotation.)
- Calculates the accuracy of the Viterbi decoding, defined as the percentage of predicted plus and minus states that match the authoritative annotation.

4. **Gibbs sampling.** Implement a Gibbs sampling motif finder. We provide a template file `gibbs.py`, but you are free to use a programming language of your choice. The Gibbs sampling algorithm can be summarized as follows:

```
GibbsSampler($S_1, S_2, \ldots, S_t, L$):
Inputs:
$S_1, S_2, \ldots, S_t$: strings that contain a common pattern
$L$: the length of the probabilistic pattern to find

Choose a substring of length $L$ from each of $S_1, S_2, \ldots, S_t$

```

```
Repeat until convergence:
Choose a sequences $S_c$

Create a pattern using all substrings except the one from $S_c$

Score all substrings of length $L$ in $S_c$ using the pattern

Stochastically choose a new substring from $S_c$ with
          probability proportional to its score

          Return the estimated pattern
```

Many details of this procedure have intentionally been left ambiguous. Specific implementations are described in lecture and J412, but we encourage you to experiment with different approaches. Make sure to discuss your design decisions in your solutions.

Run your code on the four provided data sets. `data1` is an artificial data set with a planted, identical motif, `data2` is an artificial data set with a degenerate motif, and `data3` and `data4` are real promoter sequences from yeast known to bind the transcription factors ACE2 and MBP1, respectively. The GC content of `data1` and `data2` is approximately 0.5 and the GC content of `data3` and `data4` is approximately 0.37.

Turn in a printout of the code you added in addition to what patterns of length 10 your program found consistently for each the four provided data sets. Repeat the procedure several times for each data set until you believe you have found the strongest patterns. The pattern you find in `data1` should be very strong.

5. **(6.878 only) Gibbs sampling in-depth.** Given only a procedural description of Gibbs sampling (e.g. from problem 4), it may not be obvious that it converges to a meaningful answer. In this problem, we'll walk through a simplified version of the algorithm in order to understand how and why it works.

We will look for a motif of length $(L+1)$ in just two sequences, $X = x_1x_2\cdots x_{M+L}$ and $Y = y_1y_2\cdots y_{N+L}$. Let $P(I=i, J=j)$ be the joint probability that a motif starts at position i in sequence X, $i \in \{1, 2, \ldots, M\}$, and at position j in sequence Y, $j \in \{1, 2, \ldots, N\}$. If we knew this distribution, then we could find the most likely positions for a motif as $\arg\max_{i,j} P(I=i, J=j)$. Unfortunately, we do not know $P(I=i, J=j)$.

(a) Design expressions for the conditional distributions $P(I=i|J=j)$ and $P(J=j|I=i)$. These expressions need not be closed form. Conceptually, $P(I=i|J=j)$ answers the question: given that some motif appears in Y at $y_j\cdots y_{j+L}$, what is the probability that the same motif appears in X at $x_i\cdots x_{i+L}$? (Hint: A simple heuristic would define $P(I=i|J=j)$ to be proportional to the percent identity of $x_i\cdots x_{i+L}$ and $y_j\cdots y_{j+L}$.) It is up to you how, or whether, to incorporate a background model. In any case, for all j, $\sum_{i=1}^{M} P(I=i|J=j) = 1$, and for all i, $\sum_{j=1}^{N} P(J=j|I=i) = 1$.

The Gibbs sampler initializes with an arbitrary position in X, which we'll call I_0. We stochastically choose J_0 (a position in Y) according to $P(J_0=j|I_0=i)$, one of the conditional distributions you defined. Then, we choose I_1 according to $P(I_1=i|J_0=j)$. Then we choose $J_1, I_2, J_2, I_3, J_3, I_4, \cdots$.

(b) We can view the conditional distributions you defined in part (a) as transition matrices in this sequence of positions, giving the probability of the next choice based on the last choice. Assume you have computed the transition matrices A and B where $a_{ji} = P(I = i|J = j)$ and $b_{ij} = P(J = j|I = i)$. Give a formula for the transition matrix C where $c_{ii'} = P(I_{k+1} = i'|I_k = i)$.

(c) Now, consider a Markov chain governed by the transition matrix C, which generates the sequence $I_1, I_2, I_3, \ldots$ The states in this Markov chain correspond to positions in the sequence X. (We could similarly derive a Markov chain for Y.) What is $P(I_{6878} = i|I_0 = i_0)$ in terms of C?

(d) Determine the marginal distribution $P(I = i) = \sum_{j=1}^{N} P(I = i, J = j)$ in terms of C. (Your answer will most likely be specified as a system of $M + 1$ equations.) Given $P(I = i)$, how would you find $\arg\max_{i,j} P(I = i, J = j)$, the most likely positions of the motif?

We now have an exact solution to this motif finding problem for two sequences. In the case of n sequences, we would have n conditional distributions, each dependent on the motif positions in the $n - 1$ other sequences. The transition matrices in part (b) would explode in size, making this strategy impractical.

(e) In the two-sequence case, how could you approximate $P(I = i)$ by simulating the Markov chain? Based on this intuition, outline a strategy for estimating the positions of the motif.

Gibbs sampling is one of a family of approaches, called *Markov chain Monte Carlo* algorithms, for generating samples from a distribution given only limited information about it, such as only its pdf or, in the case of Gibbs sampling, only the conditionals. Gibbs sampling is particularly useful when the conditionals are much easier to formulate than the full joint. Although there is great flexibility in how you can define the conditionals, there are certain restrictions that we didn't mention, which imply the existence of a solution to part (d).