Omics Big Data

Spring 2019

Homework 4, week 4

1. Roll your own PROSITE pattern scanner in Perl or any programming language of your choice.
   Write a Perl script or a program with your favorite programming language that searches a protein sequence database with a PROSITE pattern.

   Input: Your script should take two arguments: a PROSITE pattern string, and a filename for a FASTA format sequence database. For example:
   % ./protsite.pl "[RK]-G-{EDRKHPCG}-[AGSCI]-[FY]-[LIVA]-x-[FYLM]." /share/home/ccwei/courses/2019/omics/hw4/ws_215.protein.fa
   Be sure that your program can accept any PROSITE pattern gracefully. A full definition of PROSITE syntax is listed in the a file located in our course server at /share/home/ccwei/courses/2019/omics/hw4/pattern_syntax_rules. You will want to deal with the following pattern elements:
   []: sets of allowed amino acids
   {}: sets of disallowed amino acids
   -: separator between pattern elements
   (x) and (x,y): length ranges
   <,>: anchor to N- or C-terminus
   .: end of pattern.

   (Hint: By far the easiest strategy will be to convert the PROSITE pattern to a Perl regular expression.)

   Output: for every match, your script should print out the name and description of the target sequence and the start/end point of the pattern match. Your script should be able to deal with multiple hits per target sequence. For example:

```
52   59   RU1A_HUMAN    U1   SMALL    NUCLEAR   RIBONUCLEOPROTEIN A (U1 SNRNP A PROTEIN).
138  145  PAB1_HUMAN    POLYADENYLATE-BINDING PROTEIN 1 (POLY(A) BINDING PROTEIN 1
231  238  PAB1_HUMAN    POLYADENYLATE-BINDING PROTEIN 1 (POLY(A) BINDING PROTEIN 1
233  340  PAB1_HUMAN    POLYADENYLATE-BINDING PROTEIN 1 (POLY(A) BINDING PROTEIN 1
```

2. The Rosencrantz and Guildenstern inference problem
   For some people, this may be a nontrivial problem. Leave time to do it properly. I think it'll be worth your time… you should learn how HMM algorithms work.

   In the Tom Stoppard play Rosencrantz and Guildenstern Are Dead, the play opens with Guildenstern flipping coins that always come up heads, against all odds, leading to a discussion of probabilities that sets the tone for the play. In the play, Rosencrantz has no doubt about what the next flip will produce, and he complains that the flipping game is boring. Consider the following more complicated problem, in which Guildenstern's flips are a

hidden Markov process, and Rosencrantz must use HMM theory to figure out what's going on. Stoppard, I'm sure, would approve.

Here are the rules to a game that our more interesting Guildenstern plays every day. Guildenstern has two coins. The first coin is fair, and the second coin is biased. He starts with the first coin. After each flip, there is a small probability that he will secretly switch to the second coin. Then after each flip of the second coin, there is a small probability that he will stop flipping coin for the day. (He never switches back to the fair coin.) He asks Rosencrantz to guess when he switched to the second coin, based solely on the observed sequence of heads and tails.

Clearly Rosencrantz can model this as a hidden Markov process. The two coins behave as two HMM states, emitting H and T with some probability; the small switching probability behaves as a transition probability from the first to the second state, and the small ending probability behaves as a transition probability to the usual HMM special end state.

An example of a string that Rosencrantz might see:
THHHHTTHTTHTTHHTTTTHHTHTHTHTTTTTTHHHHHHTTHTHTHTHHHH
Rosencrantz's problem: he can't tell for sure just from looking at a string like this when Guildenstern switched to the biased coin. But, if he learns HMM theory, he can make a really good guess (or, as we say to make it sounds less like guesswork, a statistical inference).

Rosencrantz sees a different sequence of heads and tails every day that they play the game. A file of the sequences for 100 days of the game is the example file /share/home/ccwei/courses/2019/omics/hw4/example; the format is one sequence per line, and some of the lines can be fairly long. Have a look at the file… do you think you could guess where the switch happens in each sequence? It's pretty hard to do by eye (I can't do it).

Part a. The HMM architecture.
Draw the HMM architecture (states, emissions and state transitions) that corresponds to Guildenstern's game. You don't need to turn this picture in; but you need it for your own use for the next parts of the question, and we'll see its structure anyway in the two Perl scripts you write. (You might include a description of the architecture in the comments of your Perl script, though – it'll help us understand what you're doing, if you do something we don't expect!).

Part b. Viterbi HMM alignment
Let's make it a bit easy on Rosencrantz: let's assume he's played the game so much that he knows Guildenstern's parameters. The first (fair) coin has emission probabilities p(H)=p(T)=0.5. The second (biased) coin has emission probabilities p(H) = 0.8, p(T) = 0.2. The probability of switching from the first to the second coin is 0.01; the probability of ending the sequence after flipping the second coin is 0.05.
Implement a Viterbi dynamic programming alignment procedure in Perl for the HMM you drew above, using these parameters. Have your script read a file such as the example data

file /share/home/ccwei/courses/2019/omics/hw4/example and, for each sequence, find the maximum likelihood state path. For each sequence, have your Perl script print out the maximum likelihood guess for which position was the first flip of the biased coin (e.g. a number from 2..N for a sequence of length N).

Part c. Forward-Backward HMM alignment and posterior decoding

Of course, the maximum likelihood position is still just a guess; its count can easily be wrong. We can use Forward-Backward and posterior decoding to get an even more detailed inference for Rosencrantz.

Implement Forward and Backward dynamic programming procedures in Perl for the HMM, using the same parameters as above. Have your Perl script read a file such as the example data file /share/home/ccwei/courses/2019/omics/hw4/example), and then, for each sequence in the file, use the Forward and Backward variables to calculate the posterior probability distribution over positions of the first flip of the biased coin… e.g. for a given position, what is the probability that this position is the correct guess? For each sequence, have your Perl script print out the best (most probable) five positions, followed by the summed posterior probability of all five (e.g. is the probability that one of the five is right).

Hints:

In this case, the position with maximum a posterior probability will agree with the position indicated by the Viterbi trace. This is not always the case for more complicated HMMs.

For debugging, it might be helpful to write an auxiliary Perl script that generates strings from Guildenstern's "HMM". The script I used, for example, is /share/home/ccwei/courses/2019/omics/hw4/generate.pl. This way you can generate data and keep track of the correct answer (when the model switched to the biased coin), and see how well your inference engines guess.

Another debugging tip: in this case, there's only N-1 possible state paths for a sequence of length N. You can actually enumerate them all, and calculate likelihoods explicitly! This isn't terribly efficient but it is a good way to double check the answers from your DP algorithms. Normally, the number of possible state paths for an HMM is exponentially large, and you can't do this.

Turning in your work

Submit your homework to http://cgm.sjtu.edu.cn/test/obd/index.html. Please submit it before 10:00AM on March 27th, 2019. You are also required to submit a hard copy before the class start on March 27th, 2019.

Ask **TA Huimin Lu: linuslu6@outlook.com** in case you have any question about the homework submission webpage.

----------cut-----here----------

独立作业承诺：（请选择一个， 并签名）

1. 本人，_____，保证本次作业由自己独立完成。

   签名

   时间　　年　月　日

或者

2. 本人，_____，保证本次作为和_____同学讨论后，由自己独立完成。
   讨论内容包括_____

   签名
   时间　　年　月　日