COMP 790-124, HW3

Wile E. Coyote

October 30, 2012

Deadline: 11/9/2012 11:59PM EST
Submit `hw3.pdf` by e-mail, `mailto:vjojic+comp790+hw3@cs.unc.edu`.

Instructions

Software In most cases the department and university machines come with software that we need pre-installed. If not you will need to install

1. a LaTeXimplementation, on Windows MikTex `http://miktex.org/`, on Mac MacTex `http://www.tug.org/mactex/2011/`
2. MATLAB following instructions from `http://software.unc.edu`
3. RECOMMENDED: git to keep version of your homework files as you work on them `http://git-scm.com/downloads/`

If you are working on a department/university machine some or all of this may already be installed.

Files for HW3 Download `http://www.cs.unc.edu/~vjojic/comp790/hw3.zip` and unpack it into a directory. In the directory you unpacked the zip file, you will find several .m,.tex,.sty files and hw3.pdf. You will be changing hw3.tex and adding some pdf generated by MATLAB scripts into this directory. You should learn MATLAB and LaTeX, but for this assignment instructions will be pretty straightforward. If you find yourself stuck on how to write out a particular piece of math in LaTeXlook at `http://tobi.oetiker.ch/lshort/lshort.pdf`.

Problems and points Maximum number of points that contribute to your grade from this homework is 10. If you earn more than 10 points, this will be recorded, but it will not affect your grade.

Read all the problems CAREFULLY, some will have an obvious $\boxed{\texttt{answer}}$ box which means that you will find `\answer` in hw3.tex and replace it with your answer. Others will have you change something in a piece of code or may require you to run a piece of code and include a .pdf in the hw3.tex. Hence, not all the problems will have explicit and obvious $\boxed{\texttt{answer}}$ box. As a rule of thumb EVERY problem will have you change the hw3.tex. If you completed a problem, but have not changed hw3.tex, go back and read the problem you are forgetting something. Once you have changed hw3.tex remake hw3.pdf and look at it. Make sure that your answer has been updated and it looks clean and readable. Keep copies of hw3.tex so you can easily revert to an older version (subversion is fine for this).

Turning the homework in You will send me hw3.pdf file by e-mail. If you send me multiple versions of the document I will read the latest version that arrived prior to the deadline.

We will assume existence of an N long backbone sequence s in an N. In this assignment the alphabet will be of size 4, corresponding to nucleotides. We will construct a hidden Markov model that generates a shorter sequence from the backbone sequence. The shorter sequence will consist of two parts of equal length L. First part of sequence corresponds to offsets, 1 through L, and the second part of the sequence corresponds to offsets, $L+1$ to $2L$. With each offset i in the two-part sequence we will associate a hidden variable that points to a position in the backbone sequence. The probability of a letter x_i, given pointer h_i is

$$p(x_i|h_i) = \begin{cases} 0.99, & \text{if } x_i = s_{h_i} \\ \frac{0.01}{a-1}, & \text{if } x_i \neq s_{h_i}. \end{cases}$$

Finally, we define transition probability on the h_i

$$h_{i+1}|h_i \propto \begin{cases} \text{TruncPoiss}(h_{i+1} - h_i), & \text{if } i = L \\ \pi_{\text{ins}}, & \text{if } h_{i+1} = h_i, i \neq L \\ \pi_{\text{del}}, & \text{if } h_{i+1} = h_i + 2, i \neq L, h_i + 2 \leq N \\ \pi_{\text{copy}}, & \text{if } h_{i+1} = h_i + 1, i \neq L, h_i + 1 \leq N. \end{cases}$$

where TruncPoiss denotes a truncated poisson, given by

$$\text{TruncPoiss}(l) \propto \begin{cases} \text{Poiss}(l, \lambda = 100), & \text{if } 90 \leq l \leq 110 \\ 0, & \text{otherwise} \end{cases}$$

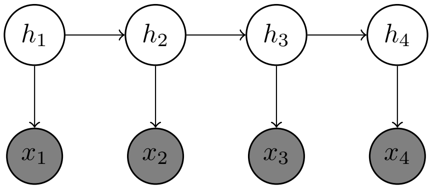
Hence, we move the pointer forward with an occasional skip (deletion) or lag (insert), except when we reach offset L where we make a leap of approximately 100 positions.

We will use a `logsum` function

```
function s = logsum(vec)
m = max(vec);
s = log(sum(exp(vec - m))) + m;
```

and `logProbPoiss`

```
function logProb = logProbTruncPoiss(i,lambda)
logProb = log(lambda)*i + (-lambda) - sum(log(1:i));
```



Problem 1(3pt) Given the HMM model specification above implement a forward pass in the HMM. Storing the transition matrix explicitly would be too costly. But the matrix is very sparse. With the exception of $i = L$, from a given offset h_i we can transition to *at most* three different offsets. Hence computation of the forward pass will require us to explicitly account for these possibilities. Note that if $h_i = N$ then the only possible transition is to $h_{i+1} = N$.

Note that transition from h_L to h_{L+1} is made according to a Poisson.

```
function m_f = fw(s,x,pins,pdel,pcopy)
N = length(s);
L = length(x)/2;
m_f = -realmax*ones(N,2*L);

logpins = log(pins);
logpdel = log(pdel);
logpcopy = log(pcopy);
for a=1:4
    for b=1:4
        logmut(a,b) = log(0.99)*(a==b) + log(0.01)*(a~=b);
    end
end

m_f(1:N,1) = -log(N) + logmut(s,x(1));
for i=2:2*L
    for prev=1:N
        if i~=L+1
            % insert
            vala = m_f(prev,i);
            valb = m_f(prev,i-1)+ logmut(s(prev),x(i)) + logpins;
            m_f(prev,i) = logsum([vala valb]);
            if prev<=N-2
                % delete
                vala = m_f(prev+2,i);
                valb = ...
                m_f(prev+2,i) = logsum([vala valb]);
            end
            if prev<=N-1
                % copy
                vala = m_f(prev+1,i);
                valb = ...
                m_f(prev+1,i) = logsum([vala valb]);
            end
        else
            if prev+90<=N
                for next=prev+90:min(prev+110,N)
                    vala = m_f(next,i);
                    valb = ...;
                    m_f(next,i) = logsum([vala valb]);
                end
            end
        end
    end
end
```

Run forward pass on inputs stored in `hw3.mat` and run this script

```
m_f = fw(seq,x,0.005,0.005,0.99);
logProb = logsum(m_f(:,end))
```

The resulting `logProb` is $\boxed{\texttt{answer}}$

Problem 2(3pt) Implement a backward pass.

```
function m_b = bw(s,x,pins,pdel,pcopy)
N = length(s);
L = length(x)/2;
m_b = -realmax*ones(N,2*L);

logpins = log(pins);
logpdel = log(pdel);
logpcopy = log(pcopy);
for a=1:4
    for b=1:4
        logmut(a,b) = log(0.99)*(a==b) + log(0.01)*(a~=b);
    end
end

m_b(:,2*L) = 0;
for i=2*L-1:-1:1
    i
    for next=1:N
        if i~=L
            % insert
            vala = m_b(next,i);
            valb = m_b(next,i+1)+ logmut(s(next),x(i+1)) + logpins;
            m_b(next,i) = logsum([vala valb]);
            if next-2>=1
                % delete
                vala = m_b(next-2,i);
                valb = ...
                m_b(next-2,i) = logsum([vala valb]);
            end
            if next-1>=1
                % copy
                vala = m_b(next-1,i);
                valb = ...
                m_b(next-1,i) = logsum([vala valb]);
            end
        else
            if next-90>=1
                for prev=max(1,next-110):next-90
                    vala = m_b(prev,i);
                    valb = ...;
                    m_b(prev,i) = logsum([vala valb]);
                end
            end
        end
    end
end
```

To check your implementation run following code

```
m_f = fw(seq,x,0.005,0.005,0.99);
m_b = bw(seq,x,0.005,0.005,0.99);
mm = m_f + m_b;
logsum(mm(:,1))
logsum(mm(:,end))
```

If the two logsum calls output different values, you have a bug.

Problem 3(3pt) Implement Viterbi forward and backward pass. To do this transform your forward pass and backward by replacing the `logsum` with `max`

Problem 4(2pt) Use MATLAB commands `tic` and `toc` to measure the time that it takes to run standard forward pass and Viterbi forward pass to complete. The ratio of the these times is $\boxed{\texttt{answer}}$. Is one of these functions faster? If so, why?

Hint: One way to answer this question is to use MATLAB's profiler. Before calling a function you want to profile do folowing

```
profile clear
profile on
```

Run your code and once done (or once you interrupt it)

```
profile viewer
```

Rest ought to be self-explanatory.

Problem 5(3pt) Modify your Viterbi forward pass to also record which of the states h_{i-1} was the most likely to have given rise to the current state h_i. This is sometimes called trace or backward pointers.

Implement code that starting with the state h_L with highest probability in the forward pass, backtracks according to the trace recording the path. Apply this procedure to long sequence `seq` and short sequence `x` in `hw3.mat`.

Paste most likely sequence of offsets in `seq` used to generate `x`

```
...
```