COMP 790-125, HW2

Wile E. Coyote

October 25, 2015

Deadline: 11/5/2014 11:59PM EST
Submit `hw2.pdf` by e-mail, `mailto:vjojic+comp790+hw2@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 LaTeX implementation, 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/`

Files for HW2 Download `http://www.cs.unc.edu/~vjojic/comp790/notes/hw2.zip` and unpack it into a directory. In the directory you unpacked the zip file, you will find several .m,.tex,.sty files and hw2.pdf. You will be changing hw2.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 LaTeX look 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 `answer` box which means that you will find `\answer` in hw2.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 hw2.tex. Hence, not all the problems will have explicit and obvious `answer` box. As a rule of thumb EVERY problem will have you change the hw2.tex. If you completed a problem, but have not changed hw2.tex, go back and read the problem you are forgetting something. Once you have changed hw2.tex remake hw2.pdf and look at it. Make sure that your answer has been updated and it looks clean and readable. Keep copies of hw2.tex so you can easily revert to an older version (`git` is fine for this).

Your code may be in Python or R, instead of MATLAB. Just paste it in place of the MATLAB code where appropriate.

Turning the homework in You will send me hw2.pdf file by e-mail to `mailto:vjojic+comp790+hw2@cs.unc.edu`. If you send me multiple versions of the document I will read the latest version that arrived prior to the deadline.

Read carefully and fill in the details. This homework is not meant to be difficult, rather it is meant to establish flow of putting together a generative model, and developing an EM algorithm.

We will derive and an algorithm for motif discovery.

We are given

- an alphabet $\mathcal{A}$
- a dataset of sequences over the alphabet, each containing a subsequence conforming to a pattern.

The task of motif discovery is to uncover the pattern.

Here is a simple example of a motif discovery dataset

```
VCKMMOMMOMUDG
QGMOMMOMBIKYJ
JDXXGYVMOMMOM
NFYJKMOMMOMOA
KMOMMOMGMDCBB
```

The motif here is `MOMMOM`.

Our approach is going to be to specify a probabilistic model that might have been used to generate the dataset. Then we will train this model on the dataset to obtain parameters describing the pattern.

Problem 1(2pt) We will propose a probabilistic model for motif sequences of length L. Under this model, any sequence of length L will have a non-zero probability. However, a trained model ought to assign low probability to most sequences.

Our probabilistic model for a sequence generated from a motif will be given by

$$p(\mathbf{x}|\theta) = \prod_{i=1}^{L} \prod_{a \in \mathcal{A}} \theta_{a,i}^{[x_i=a]}.$$

where $\mathbf{x}$ is a sequence, and θ are parameters describing a motif. Note that θ is of size $A \times L$ where A is size of alphabet, and L is length of sequence. We will assume that $\theta_{i,a} > 0$ and $\sum_a \theta_{i,a} = 1$. Hence, every column of matrix θ is a categorical distribution.

Sampling categorical distribution We will need to sample discrete distribution. So, we will write code for this.

Implement inverse cdf sampling for a discrete distribution

```
function s = samplediscrete(ps,T)
cdf = cumsum( ... )
S = zeros(T,1);
```

```
for t=1:T
     r = rand(1,1);
     s(t) = find( ... , 1);
end
```

`find(vec,1)` finds first occurrence of a non-zero value in vector `vec` and returns its index.

Check your code by

```
ps = [0.1 0.3 0.6];
T = 1000;
data = samplediscrete(ps,1000);
f = [sum(data==1)/T sum(data==2)/T sum(data==3)/T]
```

If you did everything right, vector of frequencies `f` should be close to the vector of probabilities used to draw the sample `ps`. Something like

```
f =
    0.0940    0.3110    0.5950
```

Generating a θ matrix In order to play with synthetic data, we will need to generate a θ matrix. You could do this by hand, but we can also generate it randomly. We want a matrix that puts much of its probability mass on one or two of possible states.

```
function theta = randomtheta(A,L)
theta = rand(A,L)
theta = theta.^10;
theta = theta./repmat(sum(theta),[A 1]);
```

The code above raises entries to the 10^{th} power. Play with different exponents and compare outputs of `randomtheta`. Explain what happens. `answer`

Sampling sequences from the model We will write code to generate synthetic sequences from a motif model (specified by a θ matrix).

Recall that we represent sequences as binary matrices rather than strings.

Implement code that samples sequences given a motif.

```
function X = samplemotif(theta,T)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
% does each column sum to 1
assert(all(abs(sum(theta,1) - 1.0)<1e-6))
% are all probabilities greater than 0
assert(all(theta(:)>0))
X = zeros(A,L,T);
for t=1:T
    for i=1:L
```

```
        % drawn a single letter according to theta(:,i)
        letter = samplediscrete(...);
        X(letter,i,t) = 1;
    end
end
```

Generate some motifs

```
A = 4; L = 5; T = 300;
groundtruththeta = randomtheta(A,L);
mymotifs = samplemotif(mytheta,T);
```

Take a look at this data.

Computing probability of a sequence under motif model $\log p_f(\mathbf{x}|\theta)$ We will denote probability of a motif sequence as p_f where f stands for foreground.

Our model tells us that probability of a sequence of length L under motif model is

$$p_f(\mathbf{x}|\theta) = \prod_{i=1}^{L} \prod_{a \in \mathcal{A}} \theta_{a,i}^{[x_i=a]}.$$

Hence it's log is

$$\log p_f(\mathbf{x}|\theta) = ...$$

Implement computation of the log probability, remember sequence input is a matrix of indicators not a string:

```
function lp = logpf(x,theta)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
assert(size(x,1) == A && size(x,2) == L);
lp = sum( sum(... ) )
```

Problem 2(2pt) In motif discovery task, we are provided with a set of sequences that contain examples of the motif. We have not yet specified how to generate parts of the sequences that are outside of the motif.

We will assume that parts of sequences that are not coming from a motif are generated from the alphabet with uniform probability for each letter.

$$p_b(x_i = a) = \frac{1}{|\mathcal{A}|}, \forall a \in \mathcal{A}$$

Sometimes this is called a background model. The motif can be thought of as foreground. Note that we are using p_b to be able to distinguish foreground $p(\mathbf{x}|\theta)$ and background model $p_b(\mathbf{x})$

Putting together motif and background models We wish to generate full sequences that contain the motif and the rest of the letters are just noise. To do so, we need to decide where in the sequence the motif will occur. We will introduce a latent variable, h, that stores offset at which the motif starts. If a sequence is of length N, then $h \in \{1, \ldots, N-L+1\}$. Why $N-L+1$? `answer`

Since h is a random variable we need to specify a prior distribution for it. We do not have any special information about where the motifs are likely to occur so we choose uniform

$$p(h=i) = \frac{1}{N-L+1}.$$

Let's write out joint probability of a sequence $\mathbf{y}$ and an offset h:

$$p(\mathbf{x}, h|\theta) = \underbrace{p(h)}_{A} \underbrace{\left[\prod_{i=1}^{h-1} p_b(x_i)\right]}_{B} \underbrace{p_f(\mathbf{x}_{h:h+L-1}|\theta)}_{C} \underbrace{\left[\prod_{i=h+L}^{N} p_b(x_i)\right]}_{D}$$

Explain each part of the above expression A is `answer` . B is `answer` . C is `answer` . D is `answer` .

Substitute definitions of $p(h)$ and p_b in the above expression and simplify

$$p(\mathbf{x}, h|\theta) = ...p_f(\mathbf{x}_{h:h+L-1}|\theta)$$

Take a log of it

$$\log p(\mathbf{x}, h|\theta) = \log ... + \log p_f(\mathbf{x}_{h:h+L-1}|\theta)$$

Write code that evalutes this log probabilty:

```
function lp = logp(x,h,theta)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
N = size(x,2);     % length of full sequence
lp = ... + logpf(x(:,h:h+L-1,theta)
```

Remember, you already implemented `logpf`.

We will now write code to generate synthetic examples of full sequences with motifs.

```
function [X,groundtruthh] = samplesequences(N,T,theta)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
X = zeros(A,N,T); % these are full sequences
pbackground = 1/A*ones(A,1);
ph = 1/(N-L+1)*ones(N-L+1,1);
h = zeros(1,T);
for t=1:T
```

```
    % populate sequence according to background model
    for i=1:N
        letter = samplediscrete(...,1);
        X(letter,i,t) = 1;
    end
    % sample motif start offset
    h(t) = samplediscrete( ...,1);
    % sample motif sequence
    sub = samplemotif(...,1);
    % put the motif at the chosen offset
    X(:,h(t):h(t)+L-1,t) = sub;
end
groundtruthh = h;
```

Problem 3(2pt) We are going start implementing an EM algorithm.
EM iterates two steps

E step Compute table for all instances t and all offsets l $q(h^t = l) = \frac{p(h^t=l|\mathbf{x}^t,\theta)}{\sum_o p(h^t=o|\mathbf{x}^t,\theta)}$

M step Given table q update parameters $\theta = \text{argmax}_\theta \sum_t \sum_l q(h^t = l) \log p(\mathbf{x}, h^t = l|\theta)$

Let's compute log-likelihood

$$\text{LL}(\theta; \mathbf{X}) = \sum_{t=1}^{T} \log p(\mathbf{x}|\theta)$$

Express the log-likelihood in terms of joint probability $p(\mathbf{x}, h|\theta)$

$$\text{LL}(\theta; \mathbf{X}) = \sum_{t=1}^{T} \log ...$$

Here is an implementation of log likelihood computation and E-step rolled into one. Correct the code below by placing correct indexes in expression using `q`

```
function [q,ll] = estep(theta,X)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
N = size(X,2);     % length of full sequence
T = size(X,3);     % number of examples
q = zeros(N-L+1,T);
ll = 0;
for t=1:T
    xt = X(:,:,t);
    for h=1:N-L+1
        q(...) = logp(xt,h,theta);
```

```
    end
    lp = logsum(q(...));
    ll = ll + lp;
    q(...) = exp(q(...) - lp);
end

function l = logsum(v)
m = max(v); v = v - m;
l = log(sum(exp(v))) + m;
```

Explain why we used `logsum` and not `sum`. `answer`

What is stored in q at the end of this operation? Explain the code, this can involve math, but you can also use phrases "log", "unnormalized posterior", "logsum", etc. `answer`

We will now check your E step code

```
A = 4;L = 5;N = 10;T = 300;
mytheta = randomtheta(A,L);
[X,groundtruthh] = samplesequences(N,T,mytheta);
q = estep(mytheta,X);
[~,besth] = max(q);
f = sum(besth ~= groundtruthh)/T
```

If you did everything right you should see something close to

```
f =
    0.0500
```

What is stored in `besth` and `f`? `answer`

Why should `f` be small? `answer`

Problem 4(2pt) Given table q we can implement M-step:

$$\theta = \underset{\theta, \sum_a \theta_{a,i}=1, \forall i}{\operatorname{argmax}} \underbrace{\sum_t \sum_l q(h^t = l) \log p(\mathbf{x}, h^t = l|\theta)}_{f(\theta)}$$

We define

$$f(\theta) = \sum_t \sum_l q(h^t = l) \log p(\mathbf{x}, h^t = l|\theta)$$

Expand $p(\mathbf{x}, h^t = l|\theta)$ all the way until you only have product of constants and θ in

$$f(\theta) = \sum_t \sum_l q(h^t = l) \log ...$$

Lagrangian for the optimization problem is

$$L(\theta) = f(\theta) + \sum_i \lambda_i \sum_a (\theta_{a,i} - 1)$$

Compute

$$\frac{\partial L(\theta)}{\theta_{a,i}} = ...$$
$$\frac{\partial L(\theta)}{\lambda_i} = ...$$

Equate all of the partial derivatives to zero and solve for $\theta_{a,i}$

$$\theta_{a,i} = ...$$

Implement update for θ

```
function theta = mstep(q,X)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
N = size(X,2);     % length of full sequence
T = size(X,3);     % number of examples
assert(size(q,1) == N-L+1 && size(q,2) == T);
for t=1:T
    for h=1:N-L+1
        theta = theta + ... * ...
    end
end

for i=1:L
    theta(:,i) = theta(:,i) / sum(theta(:,i))
end
```

Let's check the code

```
A = 4;L = 5;N = 10;T = 300;
mytheta = randomtheta(A,L);
[X,groundtruthh] = samplesequences(N,T,mytheta);
q = estep(mytheta,X);
newtheta = mstep(q,X);
newtheta
mytheta
```

Compare `newtheta` and `mytheta`. If you did everything right, they should be very similar.

Explain why `newtheta` and `mytheta` they should be similar? `answer`

Problem 5(2pt) Put pieces together to obtain EM algorithm.

Initialize theta so that is is near uniform

```
function [theta,q] = em(X,L,iterations)
A = size(X,1); % size of alphabet
N = size(X,2); % length of full sequence
T = size(X,3); % number of examples
theta = ...
theta = theta./repmat(sum(theta),[A 1]);
for it=1:iterations
    [q,ll] = estep(...);
    lls(it) = ll;
    ... = mstep(...);
    plot(lls)
    xlabel('iterations');ylabel('log-likelihood');
    drawnow
end
```

Run this code on data in `hw2.mat`, with `L=10,iterations=50` and place log-likelihood plot in the figure.

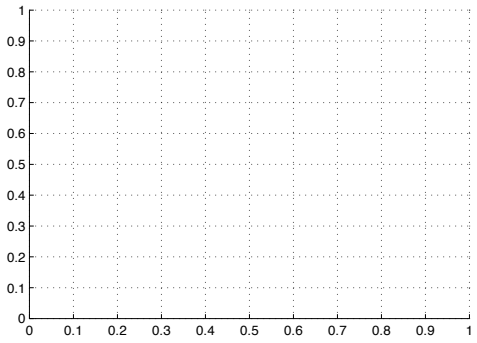

Figure 1: This is emptiness, it earns no points.

Problem 6(3pt) Use `conv2` between log of θ and each sequence $\mathbf{x}$ to efficiently compute posterior. Note that `conv2` flips one of its inputs.

```
function [q,ll] = estep2(theta,X)
A = size(theta,1); % size of alphabet
L = size(theta,2); % length of motif
N = size(X,2);     % length of full sequence
T = size(X,3);     % number of examples
q = zeros(N-L+1,T);
ll = 0;
for t=1:T
    xt = X(:,:,t);
```

```
    ... use conv2 ...
    lp = logsum(q(...));
    ll = ll + lp;
    q(...) = exp(q(...) - lp);
end

function l = logsum(v)
m = max(v); v = v - m;
l = log(sum(exp(v))) + m;
```

Compare outputs of `estep` and `estep2` and make sure that they are, up to numerical error, the same.

Demonstrate speed-up for different N and L.