

COMP 790-125, HW2

Calvin Deutschbein

November 5, 2015

Deadline: 11/5/2015 11:59PM EST

Submit `hw2.pdf` by e-mail to `vjojic+comp790+hw2@cs.unc.edu`

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
QGMOMMOMBICKYJ
JDXXGYVMOMMOM
NFYJKMOMMOMOA
KMMOMMGMDCCB
```

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(ps);
S = zeros(T,1);
for t=1:T
    r = rand(1,1);
    temp = cdf - r;
    temp(temp < 0) = 0;
    s(t) = find(temp, 1);
end
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.

It increases the disparity between higher and lower probability sections. Higher powers, higher disparity.

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(theta(:,i),A);
        X(letter,i,t) = 1;
    end
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

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

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(log(sum(times(x,theta))));
```

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 $h \in \{1, \dots, N - L + 1\}$. Why $N - L + 1$? To not insert where the motif won't fit in the string.

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 probability of desired h . B is background probability of stream before motif. C is motif probability. D is background probability of stream after motif.

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

$$p(\mathbf{x}, h|\theta) = \frac{p_f(\mathbf{x}_{h:h+L-1}|\theta) |\mathcal{A}|^N}{(N - L + 1) |\mathcal{A}|^L}$$

Take a log of it

$$\log p(\mathbf{x}, h|\theta) = \log(1/(N - L + 1)) + \log(|\mathcal{A}|^{N-L+1}) + \log p_f(\mathbf{x}_{h:h+L-1}|\theta)$$

Write code that evaluates this log probability:

```
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 = log(1/(N-L+1)) + log((1/A)^(N-L+1)) + 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 = sampledcrete(zeros((1,A)+(1/A)),1);
        X(letter,i,t) = 1;
    end
    % sample motif start offset
    h(t) = sampledcrete(zeros((1,N-L+1)+(1/(N-L+1))),1);
    % sample motif sequence
    sub = samplemotif(theta,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 $\theta = \operatorname{argmax}_{\theta} \sum_t \sum_l q(h^t = l) \log p(\mathbf{x}, h^t = l|\theta)$

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 $p(\mathbf{x}, h|\theta)$

$$\text{LL}(\theta; \mathbf{X}) = \sum_{t=1}^T \log p(\mathbf{x}, h|\theta) - \log(1/N - L + 1) - \log(|\mathcal{A}|^{N-L+1})$$

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(h,t) = logp(xt,h,theta);
    end
    lp = logsum(q(:,t));
    ll = ll + lp;
    q(:,t) = exp(q(:,t) - 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`. We are working in log space with probabilities and this facilitates normalization.

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. The log normalized posterior.

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`? `f` is a measure of fit for `besth`, which is best bet for `h` generated by the algorithm.

Why should `f` be small? Closer fit is more ideal.

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

$$f(\theta) = \sum_t \sum_l q(h^t = l) \left(\sum_{i=1}^L \sum_{a \in \mathcal{A}} \log(\theta_{a,i}^{[x_i=a]}) - \log((N-L+1)) - \log((|\mathcal{A}|)^{N-L-1}) \right)$$

Lagrangian for the optimization problem is

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

Compute, noting x_i depends on t

$$\begin{aligned} \frac{\partial L(\theta)}{\partial \theta_{a,i}} &= \sum_i \lambda_i + \sum_t \sum_l [x_i = a] \frac{q(h^t = l)}{\theta_{a,i}} \\ \frac{\partial L(\theta)}{\lambda_i} &= \sum_a \theta_{a,i} - 1 \end{aligned}$$

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

$$\begin{aligned} \forall a, a' : \sum_t \sum_l [x_i = a] \frac{q(h^t = l)}{\theta_{a,i}} &= \sum_t \sum_l [x_i = a'] \frac{q(h^t = l)}{\theta_{a',i}} = - \sum_i \lambda_i \\ \forall i : \sum_{a \in \mathcal{A}} \theta_{a,i} &= 1 \end{aligned}$$

So we know both ratio between entries in θ and the total sum of rows of θ .

$$\begin{aligned} \theta_{a,i} &= 1 - \sum_{c \in \mathcal{A} \setminus a} \theta_{c,i} = \frac{\sum_t \sum_l [x_i = a] q(h^t = l)}{- \sum_i \lambda_i} \\ \theta_{a,i} &= \frac{\sum_t \sum_l [x_i = a] q(h^t = l)}{\sum_{c \in \mathcal{A}} \sum_t \sum_l [x_i = c] q(h^t = l)} \end{aligned}$$

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 + xt(:,h:h+L-1,t)*q(h,t)
    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? This is an incremental upgrade.

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 = randomtheta(A,L);
theta = theta./repmat(sum(theta),[A 1]);
for it=1:iterations
    [q,ll] = estep(theta,X);
    lls(it) = ll;
    theta = mstep(A,L,q,X);
    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.

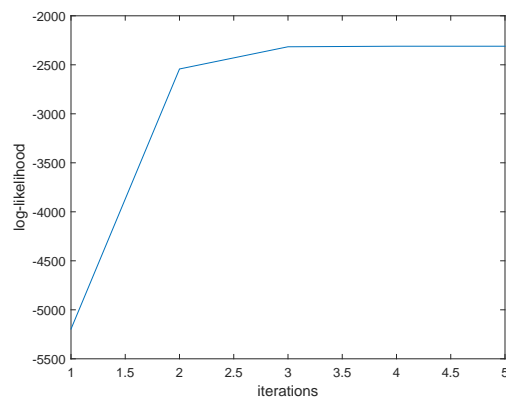


Figure 1: This is not emptiness, but it may still earn no points.

Problem 6(3pt) Use `conv2` between $\log \theta$ 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 .