Supervision Assignments in MLBI
Lent Term 2018
Set 3

Supervisor : Dionysis Manousakas
dm754@cam.ac.uk

February 24, 2018

1. *(Latent Variable Models.)*

   Describe a real-world data set which you believe could be modelled using a mixture model. Argue why a mixture model is a sensible model for your real world data set. What do you expect the mixture components to represent? How many components (or clusters) do you think there would be? What parametric form would each component have?

   1st example

   An example of the real-world dataset which could be modelled using a mixture model is the collection of data about various rocks collected at one location from different (unknown) parts of Earth's crust. Each rock can be described with parameters such as density, hardness, fraction of specific elements or compounds in the rock etc.

   A mixture model is a sensible model for this dataset because the rocks inherently can be classified into different periods of Earth's history (which correspond to some layer in the Earth's crust) and the rocks from the same period should have same properties (the mean of their attributes should be the same plus some added noise due to various different effects on the rock we are unable to describe). So reasonable thing to do is to try the model the data with mixture of Gaussians. I expect each Gaussian to represent one period in the history of Earth, and the number of Gaussians to be the number of those distinguishable periods.

   2nd example

   Consider noisy observations of the output variable in a nonlinear dynamical system with two equilibria. Input and initial state are unknown to us (we can think that the system as a black box, with a given mathematical description, which allows us only to measure the output). Based on the input and initial state, system variable may be driven to either of the two equilibria. Observed variable is continuous and corrupted with gaussian noise. Measurements are expected to be two gaussians with means the values of the equilibria and variance depending on the quality (noise) of measurements. The mixing coefficients of the model can be initialized to $1/2$ (indicating that based on our prior knowledge of the system we consider equally likely that the system variable will rest at either of the two equilibria). A mixture model makes sense

for this experiment, since measurements after transition time will be located in the neighbourhoods of the two equilibria, thus creating two clusters.

2. *(EM for Binary Data.)*

Consider the data set of binary (black and white) images used in the assignment of Problem Set 1. Each image is arranged into a vector of pixels by concatenating the columns of pixels in the image. The data set has N images $\{\mathbf{x^{(1)}}, ..., \mathbf{x^{(N)}}\}$ and each image has D pixels, where D is (number of rows) $\times$ (number of columns) in the image. For example, image $\mathbf{x}^{(n)}$ is a vector $(x_1^{(n)}, ..., x_D^{(n)})$, where $x_d^{(n)} \in \{0, 1\}$ for all $n \in \{1, ..., N\}$ and $d \in \{1, ..., D\}$.

(a) Write down the likelihood for a model consisting of a mixture of K multivariate Bernoulli distributions. Use the parameters $\pi_1, ..., \pi_K$ to denote the mixing proportions ($0 \leq \pi_k \leq$; $\sum_k \pi_k = 1$) and arrange the K Bernoulli parameter vectors into a matrix P with elements p_{kd} denoting the probability that pixel d takes value 1 under mixture component k.

Just like in a mixture of Gaussians we can think of this model as a latent variable model, with a discrete hidden variable $s^{(n)} \in 1, ..., K$ where $P(s^{(n)} = k|\boldsymbol{\pi}) = \pi_k$.

(b) Write down the expression for the responsibility of mixture component k for data vector $x^{(n)}$, i.e. $r_{nk} = P(s^{(n)} = k|\boldsymbol{x}^{(n)}, \boldsymbol{\pi}, P)$.

(c) Implement the EM algorithm for a mixture of K multivariate Bernoullis. The algorithm should take as input K, a matrix X containing the data set, and a number of iterations. The algorithm should run for that number of iterations or until the log likelihood converges (does not increase by more than a very small amount). Beware of numerical problems as likelihoods can get very small, it is better to deal with log likelihoods. Also be careful with numerical problems when computing responsibilities – it might be necessary to multiply the top and bottom of the equation for responsibilities by some constant to avoid problems. Hand in code and a high level explanation of what you algorithm does.

(d) Run your algorithm on the data set for varying $K = 2, 3, 4$. Verify that the log likelihood increases at each step of EM. Report the log likelihoods obtained (measured in bits) and display the parameters found.

(e) Comment on how well the algorithm works, whether it finds good clusters (look at the responsibilities and try to interpret them), and how you might improve the model.

(a) The probability distribution for a vector $\mathbf{x}^{(n)} = [x_1^{(n)}, x_2^{(n)}, ..., x_D^{(n)}]^T$ governed by a multivariate Bernoulli with parameters $\mathcal{P} = [p_1, p_2, ..., p_D]^T$ is given by the expression:

$$p(\boldsymbol{x}^{(n)}|\mathcal{P}) = \Pi_{d=1}^{D} p_d^{x_d^{(n)}} (1 - p_d)^{1 - x_d^{(n)}}.$$

Now, if we use a model with a mixture of the above distribution and mixing proportions $\pi_1, \pi_2, ..., \pi_k$, with $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$ and arrange the vector parameters of the K distributions in an array $P : K \times D$, we get:

$$p(\mathbf{x}^{(n)}|P, \pi) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}^{(n)}|p_k) = \sum_{k=1}^{K} \pi_k \Pi_{d=1}^{D} p_{kd}^{x_d^{(n)}} (1 - p_{kd})^{1 - x_d^{(n)}}.$$

Thus, the likelihood of N i.i.d. vectors $\mathbf{x}^{(n)}$ described by the above-mentioned model, is as follows:

$$L(\mathcal{D}|P,\boldsymbol{\pi}) = \Pi_{n=1}^{N} p(\mathbf{x}^{(n)}|P,\boldsymbol{\pi}) = \Pi_{n=1}^{N} \sum_{k=1}^{K} \pi_k \Pi_{d=1}^{D} p_{kd}^{x_d^{(n)}} (1-p_{kd})^{1-x_d^{(n)}}$$

(b) Introducing the discrete hidden variable $s^{(n)} \in \{1,..,K\}$, with $P(s^{(n)}=k|\boldsymbol{\pi}) = \pi_k$, we get for the responsibility of component k for data vector $\boldsymbol{x}^{(n)}$:

$$\begin{aligned} r_{nk} :=& P(s^{(n)}=k|\boldsymbol{x}^{(n)},\boldsymbol{\pi},P) \\ =& \frac{P(\boldsymbol{x}^{(n)}|s^{(n)}=k,\boldsymbol{\pi},P)P(s^{(n)}=k|\boldsymbol{\pi})}{\sum_{k=1}^{K} P(\boldsymbol{x}^{(n)}|s^{(n)}=k,\boldsymbol{\pi},P)P(s^{(n)}=k|\boldsymbol{\pi}))} \\ =& \frac{\pi_k \Pi_{d=1}^{D} p_{kd}^{x_d^{(n)}} (1-p_{kd})^{1-x_d^{(n)}}}{\sum_{k=1}^{K} \pi_k \Pi_{d=1}^{D} p_{kd}^{x_d^{(n)}} (1-p_{kd})^{1-x_d^{(n)}}}. \end{aligned}$$

(c.) For numerical reasons, it is better to use the log likelihood in the implementation of EM:

$$\begin{aligned} l(\mathcal{D}|P,\boldsymbol{\pi}) =& \log(L(\mathcal{D}|P,\boldsymbol{\pi})) \\ =& \sum_{n=1}^{N} \log(\sum_{k=1}^{K} \pi_k \Pi_{d=1}^{D} p_{kd}^{x_d^{(n)}} (1-p_{kd})^{1-x_d^{(n)}}). \end{aligned}$$

E-step: In Expectation step we simply evaluate the responsibilities r_{nk} for each point, given the current parameters $\boldsymbol{\pi}$ and P. At our implementation we multiply the numerator and denominator of responsibility by a constant to improve precision of calculation.

M-step: For the Maximization step, we have:

$$\begin{aligned} Q(\boldsymbol{\pi},p) = \langle \log(X,s|\boldsymbol{\pi},P)\rangle_{q(s)} =& \sum q(s) \log[p(s|\boldsymbol{\pi},P)p(x|s,\boldsymbol{\pi},P)] \\ =& \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}[\log(\pi_k) + \sum_{d=1}^{D}[x_d^{(n)} \log(p_{kd}) + (1-x_d^{(n)}) \log(1-p_{kd})]]. \end{aligned}$$

We maximize the above quantity w.r.t. the parameters. Concretely:

- W.r.t. p

$$\begin{aligned} 0 &= \frac{\partial Q}{\partial p_{kd}} \\ &= \sum_{n=1}^{N} r_{nk}(\frac{x_{nd}}{p_{kd}} - \frac{1-x_{nd}}{1-p_{kd}}) \\ &= \sum_{n=1}^{N} r_{nk}\frac{x_{nd}-p_{kd}}{p_{kd}(1-p_{kd})} \end{aligned}$$

or

$$p_{kd} = \frac{\sum_{n=1}^{N} r_{nk}x_{nd}}{\sum_{n=1}^{N} r_{nk}},$$

and thus

$$p_k = \frac{\sum_{n=1}^{N} r_{nk}x_n}{\sum_{n=1}^{N} r_{nk}}.$$

-W.r.t. $\boldsymbol{\pi}$, we should introduce a Laplace multiplier due to the constraint $\sum_k \pi_k = 1$. We have to minimize

$$\Lambda(\boldsymbol{\pi}, \lambda) = Q + \lambda(\sum_k \pi_k - 1)$$

Therefore

$$0 = \frac{\partial \Lambda}{\partial \pi_k} = 1/\pi_k(\sum_{n=1}^{N} r_{nk} + \lambda)$$

and

$$0 = \frac{\partial \Lambda}{\partial \lambda} = \sum_{k=1}^{K} \pi_k - 1$$

Combining the above, we get

$$\pi_k = \sum_{n=1}^{N} r_{nk}/N.$$

We implement below the EM algorithm:

```
function [ Pi, P, l, R, lprev, iter, L] = EM( K, X, maxiter )
%%  EM algorithm for a mixture of K multivariate Bernoulli distributions
% Input: K       -- number of mixture components
%        X       -- matrix containing the data set
%        maxiter -- maximum number of iterations
% Output: Estimates of parameters of the mixture model
%         Pi     -- vector of K mixing proportions
%                   ( 0<=Pi_k<=1, sum_k(Pi_k)=1 )
%         P      -- matrix of parameters of Bernoulli distribution
%                   (rows for each component k)

N = size(X,1);    % number of data points
D = size(X,2);    % dimension of data points

%% Initialization of parameters
Pi =1/K*ones(K,1); % equal prior probalities to each mixture component
P = rand(K,D); % we assign random uniform probabilities to each pixel
%P=P./repmat(sum(P),K,1);

%% Initial Log-likelihood
% l(X|Pi,P)=Sum_n(log(Sum_k(Pi_k( _d ((P_kd)^X_nd*(1-P_kd)^(1-X_nd))))))

outS=0;
for n=1:N
    inS=0;
    for k=1:K
        inS=inS+Pi(k)*prod((P(k,:).^X(n,:)).*((1-P(k,:)).^(1-X(n,:)))); %inner summation over components
    end
    outS=outS+log(inS);     % outer summation over samples
end
l=outS;

conver=1e-10;       % convergence threshold for EM
lprev=l-conver-1;   % artificial first log-likelihood in order to allow entering the loop

iter=0;
R=zeros(N,K); %  responsibilities matrix
L=l;          % vector of log-likelihoods

while iter<maxiter & l-lprev>conver       % loop of EM iterations

    lprev=l;    % store previous log-likelihood
    iter=iter+1;    % current iteration of EM

    %%  E-STEP
    % Evaluate responsibilities
    % R_nk=P_k* Prod_d(P_kd^(X_nd)*(1-P_kd)^(1-X_nd))/[Sum_k (P_k* Prod_d(P_kd^(X_nd)*(1-P_kd)^(1-X_nd))]

    Num=ones(N,K); % numerator of responsibilities
    denom=zeros(N,1); % normalizer of responsibilities
    const=1e05; % constant for division of numerator and denominator of responsibilities ...
                % to avoid numerical problems
    for n=1:N
        for k=1:K
            Num(n,k)=Pi(k)*prod(P(k,:).^X(n,:).*(1-P(k,:)).^(1-X(n,:)))*const;
            denom(n)= denom(n)+Pi(k)*prod(P(k,:).^X(n,:).*(1-P(k,:)).^(1-X(n,:)))*const;
        end
    end
    R=bsxfun (@rdivide, Num, denom);    %responsibilities matrix


    %%  M-STEP
    % Maximize parameters given the responsibilities matrix

    for k=1:K
```

```
        for d=1:D
            P(k,d)=(R(:,k)'*X(:,d))/sum(R(:,k)); %mixing proportions
        end
        Pi(k)=sum(R(:,k))/N; %parameters of Bernoulli distributions
    end

    %%  New Log-likelihood
    % l(X|Pi,P)=Sum_n(log(Sum_k(Pi_k(Prod_d((P_kd)^X_nd*(1-P_kd)^(1-X_nd))))))
    outS=0;
    inS=0;
    for n=1:N
        inS=0;
        for k=1:K
            inS=inS+Pi(k)*prod(P(k,:).^X(n,:).*(1-P(k,:)).^(1-X(n,:)));
        end
        outS=outS+log(inS);
    end
    l= outS;
    L(end+1)=l
end

end
```

In initialization we should avoid the degenerate case of assigning identical initial parameters P_k to each component, since this will result to convergence after only one iteration, for any choice of π_k.

(d) We run the algorithm for $K = 1, 2, 3$ according to the following script.

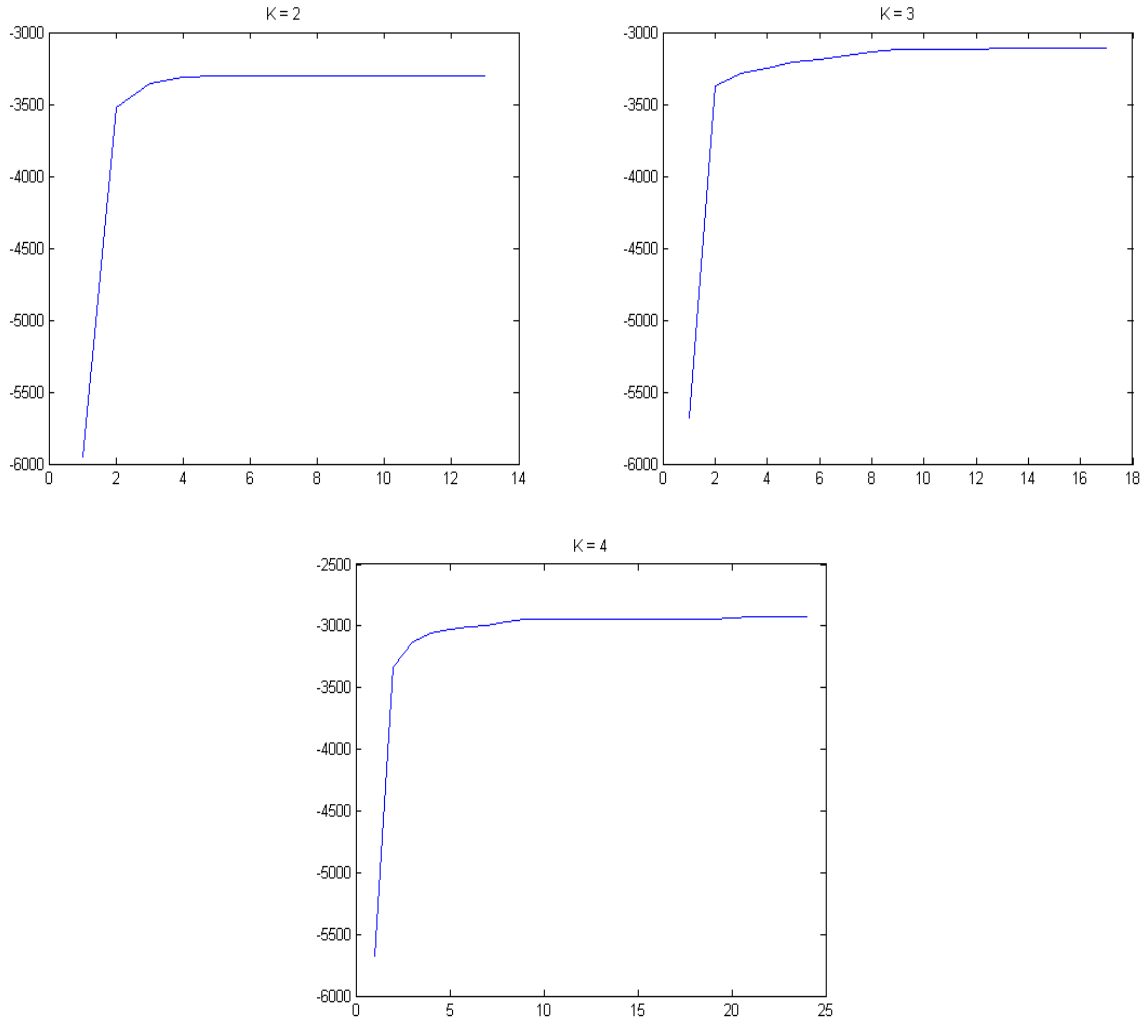
```
close all
clear all

load binarydigits.txt -ascii;
Y=binarydigits;
[N D]=size(Y);
maxiter=100;

Lfinal=zeros(3,1)
Pi=zeros
for k=2:4
    [Pi,P,l, R, lprev, iter, L]=EM(k,Y,maxiter);
    figure; plot(L);
    title(sprintf('K = %i', k))
    Lfinal(k-1)=log(abs(L(end)))
    P_{k-1}=P;
    Pi_{k-1}=Pi;
    R_{k-1}=R
    figure
    colormap gray;
    for j=1:k
        subplot(1,k,j);
        imagesc(reshape(P(j,:)',8,8)'); % plot mixing components
        axis off;
    end
end
```

We confirm that log-likelihood increases in every iteration of the EM algorithm till convergence. Convergence occurs after 20 iterations approximately for $K = 3, 4$ and a few less for $K = 2$.

(e) By observation of the responsibility values for each k we see that our implemen-




```
Lfinal =

    8.1028
    8.0433
    7.9801
```

Figure 1: Log likelihood as a function of iterations of EM algorithm. Displayed are additionally the bit-values of magnitude of log-likelihood at convergence for each K.

```
>> Pi_{1}

ans =

    0.4196
    0.5804

>> Pi_{2}

ans =

    0.3500
    0.3100
    0.3400

>> Pi_{3}

ans =

    0.5800
    0.1300
    0.1600
    0.1300
```

Figure 2: Parameters π for K=2,3,4

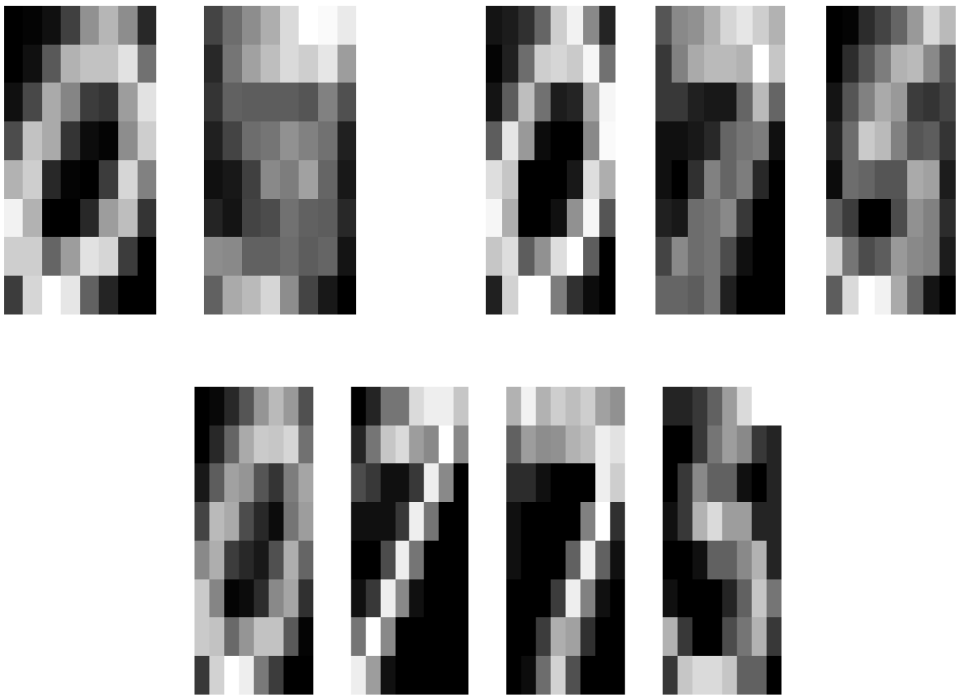


Figure 3: Parameters P, as estimated for $K = \{2, 3, 4\}$ at convergence, displayed as 8×8 image of binary pixels.

tation tends to a hard assignment of the data to one of the components of the mixture model – i.e. for each row n, R_{ni} is approximately equal to 1 for one i and zero for all the rest. (The latter holds even for ambiguous or missclassidied data points.) Thus, a good model, that would assign each image to the cluster it belongs, should have the same number of components with the sources of the data set. We can see that our data correspond to noisy images of the digits 0, 5 and 7. Model with K=3 reveals this fact by learning 3 clusters that correspond to these pictures (see Fig.3). Model with K=2 creates a cluster based on 0 and a superposition of 5 and 7, while model with K=4 creates two copies of 7.

Therefore, model K=3 seems to best fit to our data set, achieving a good classification score (e.g. repeating the execution of EM we count about 12 mistakes over the first 50 data points, which means a score of 75 %). As a result, observing by our eyes the data could help us define the correct number of mixture components and get a good classifier. However, an unsupervised system driven by the minimization of the log-likelihood would prefer K=4, because it converges to a better final value. This discussion leads to the significant generic question how could we determine the components of the model in a fully unsupervised way.

3. (*Zero-temperature EM*)

   In the automatic speech recognition community, HMMs are sometimes trained by using the Viterbi algorithm instead of the forwardbackward algorithm. In other words, in the E step of EM (Baum-Welch), instead of computing the expected sufficient statistics from the posterior distribution over hidden states: $p(s_{1:T}|x_{1:T}, \theta)$, the sufficient statistics are computed using the single most probable hidden state sequence: $s^*_{1:T} = \text{argmax}_{s_{1:T}} p(s_{1:T}|x_{1:T}, \theta)$.

   (a) Is this algorithm guaranteed to converge? To answer this you might want to consider the proof for the EM algorithm and what happens if we constrain $q(s)$ to put all its mass on one setting of the hidden variables. Support your arguments.

   (b) If it converges, will it converge to a maximum of the likelihood? If not, will it oscillate? Support your arguments.

(a) What this version of EM does is essentially to constrain hill climbing over the likelihood function during the Expectation step, by taking into account only $q(s)$ that put all their mass on only one setting of hidden variables s. Thus, we will consider distributions of the general form $q(s) = \delta(s - s_0)$. We can use the following :

$$\mathcal{F}(q, \theta) = \mathcal{L}(\theta) - KL[q(\mathcal{S})||P(\mathcal{S}|\mathcal{X}, \theta)] \leq \mathcal{L}(\theta).$$

From the definition of Kullback-Leibler divergence we have:

$$KL[q(\mathcal{S})||p(\mathcal{S}|\mathcal{X}, \theta)] = \int q(s) \log \frac{q(s)}{p(s|x, \theta)} ds = -\int q(s) \log p(s|x, \theta) = -\log p(s_0|x, \theta),$$

which is minimized for $s^* = \arg\max_s p(s|x, \theta)$.

Thus, at s^*, $-KL$ is maximized and hence $\mathcal{F}(q, \theta)$ is also maximized, in the family of q under consideration.

M-step is performed in the standard way, offering maximization of $\mathcal{F}$ w.r.t. to θ.
Therefore, we get the following scheme that never decreases likelihood:

$$\mathcal{L}(\theta^{(k-1)}) \leq \mathcal{F}(q^{(k)}, \theta^{(k-1)}) \leq \mathcal{F}(q^{(k)}, \theta^{(k)}) \leq \mathcal{L}(\theta^{(k)})$$

After a finite number of iterations free energy of EM is going to reach an asymptote and converge.

(b) At convergence of zero-temperature EM, if we go back to the forward-backward algorithm for the E step and continue the iterations with the original algorithm, we may move to better value of likelihood (because the original algorithm will search over all $q(s)$ and make the free energy equal to likelihood by making the KL divergence equal to zero). Thus, at convergence we get a maximum of the likelihood for the family of distributions of $q(s)$ that put all the mass at a specific s, but not for $q(s)$ in general.