PROJECT-2

EM Algorithm For

Multivariate Gaussian

By:- Sunny Kant
Roll- 150740
September 27, 2018

Contents

---

1 Introduction

Expectation-maximization (EM) algorithm is an iterative method to find maximum likelihood estimates of parameters in the models, where the model depends on unknown parameters of some distribution function. Algorithm involves initialisation of the unknown parameters and then alternates between an expectation (E) step, which finds the expected value of the missing observations using the current estimate for the parameters, and a maximization (M) step, which computes parameters by maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution in the next E step and this process continues until convergence of parameters.

In this project, EM algorithm is used determine the parameters of mixture of two multivariate Gaussian distributions.We treat this as missing value problem as we don't know whether the particular given data belongs to Gaussian distribution one or two($\delta = 0$ or 1).We estimate the expected value of δ of all data using the calculated or initialised parameters of the Gaussian distributions in the E-step and using the calculated value of δ ,we maximize the log—likelihood function of complete data in the M-step to obtain the parameters of the two Gaussian distributions and the p value.We will iterate these EM steps several times to convergence and obtain all the unknown parameters of the distributions.After that,using the calculated parameters,we classify the training data and cross validation data using the same expectation method used in E-step to check whether δ=0/1 ,which is used to classify our data as original currency or fake currency.
The probability density function of Mixture model is given by-

$$f(x) = pf_1(\vec{x}; \mu_1, \Sigma_1) + (1-p)f_2(\vec{x}; \mu_2, \Sigma_2)$$

where f_1 and f_2 are multivariate Gaussian distributions and $\vec{x}$is 6-variate input.

2 Description of EM Algorithm

Gaussian mixture modelling is an important task in the field of clustering and pattern recognition.Multivariate Gaussian distribution functions are given by-

$$f(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp^{\frac{-1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})}$$

where $\vec{\mu}$ is mean, Σ is covariance matrix, which is symmetric and n is the dimension of x.
The goal is to maximize the likelihood function with respect to the parameters comprising the means and covariances of the components and the mixing coefficients. We maximize the log-likelihood function considering the data using the fact that we know the data comes from which distribution 1 or 2,which are calculated as expectation of δ given the parameters of both the distribution and mixing coefficient p.
Steps involved in EM Algorithm are as-

2.1 Initialisation:

The convergence of the EM algorithm highly depends on the chosen initial parameters.There are many methods to initialise the parameters.Firstly I initialise the parameters randomly but it doesnot converges easily and gives significant error ,then I initialised the parameters by the sample mean and variance for each distribution function.But the most effective that I found during the testing on my data is Using K-means clustering Algorithm,which automatically cluster similar data examples together.The intuition behind K-means is an iterative procedure that starts by guessing the initial centroids, and then refines this guess by repeatedly assigning examples to their closest centroids and then recomputing the centroids based on the assignments.I initialised the two mean with the centroids calculated by K-means algorithm and Co-variance matrices be unit matrices.

```
p =  0.50000
>> mean1
mean1 =

     0     0     0   214   129    10

>> sigma1
sigma1 =

Diagonal Matrix

   1   0   0   0   0   0
   0   1   0   0   0   0
   0   0   1   0   0   0
   0   0   0   1   0   0
   0   0   0   0   1   0
   0   0   0   0   0   1

>> mean2
mean2 =

     0     0     0   130    10   140

>> sigma2
sigma2 =

Diagonal Matrix

   1   0   0   0   0   0
   0   1   0   0   0   0
   0   0   1   0   0   0
   0   0   0   1   0   0
   0   0   0   0   1   0
   0   0   0   0   0   1

>> |
```

2.2 E-step:

Estimate the expected values of δ using the current parameter values.

$$E(\delta|\vec{X},\vec{\theta}) = \frac{P(\delta=1,\vec{x}<\vec{X}<x+\vec{dx})}{P(\vec{x}<\vec{X}<x+\vec{dx})}$$
$$= \frac{p*f_1(\vec{x};\mu_1,\sigma_1)}{p*f_1(\vec{x};\mu_1,\sigma_1)+(1-p)*f_2(\vec{x};\mu_2,\sigma_2)}$$

where p is mixing coefficient. So, all data are given expected values of δ in this step to calculate MLE in the next step.

$$\alpha_m^{(j)} == \frac{p*f_1(\vec{x}_m^{(j)};\mu_1^j,\sigma_1^j)}{p*f_1(\vec{x}_m^{(j)};\mu_1^j,\sigma_1^j)+(1-p)*f_2(\vec{x}_m^{(j)};\mu_2^j,\sigma_2^j)}$$

where $\alpha_m^{(j)}$is the expected value of δ of m^{th} data in the j^{th}iteration and p is the updated mixing coefficient.

2.3 M-step:

Maximizing the log-likelihood function with expected values of δ, we get the following formulas to estimate the parameters.

$$p^{(j)} = \frac{1}{N}\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)$$

$$\mu_1^{(j)} = \frac{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)*\mathbf{x}^i}{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)}$$

$$\Sigma_1^{(j)} = \frac{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)(\mathbf{x}^i-\mu_m^{(j)})(\mathbf{x}^i-\mu_m^{(j)})^T}{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)}$$

$$\mu_2^{(j)} = \frac{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)*\mathbf{x}^i}{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)}$$

$$\Sigma_2^{(j)} = \frac{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)(\mathbf{x}^i-\mu_m^{(j)})(\mathbf{x}^i-\mu_m^{(j)})^T}{\sum_{i=1}^{N}\alpha_m^{(j)}(\mathbf{x}^i)}$$

where j is the iteration number and $\alpha_m^{(j)}$ is the expected value of δ of m^{th} data in the j^{th}iteration,x_i is the 6-variate i^{th} data.

2.4 Convergence checking:

We check the log value of parameters in the jth iteration to j-1 th iteration upon the jth iteration value and if it is sufficiently small ,then we assume that the parameters converges.

$$\frac{\log p(\Theta^{(j)}) - \log p(\Theta^{(j-1)})}{\log p(\Theta(j))} < \epsilon,$$

We don't check the above stated formula value in each iterarion as it might be costly , so we do this on every 20th iteration.

3 Data Analysis

Firstly we load the data in the program and divide it in two parts: training data(90 each) and cross validation data(10 each).We perform the above stated EM algorithm on the training data and got the final calculated the error in classification for both training data and cross validation data by comparing the values that we got from our classification model(stated below) and the real classified given values.

4 Classification Methods and Results

We find the expected value of δ for each data using the final parameters of the distribution function.The values are in the range of [0,1], so we classify δ=0,if the values are in the range [0,0.5],else we classify δ=1 for each 6-variate data. We do these for training set and cross-validation set separately.

We get the following value of parameters:

```
>> p
p =  0.43577
>> meanl
meanl =

   214.9879
   130.0314
   129.7969
     8.3239
    10.3286
   141.1017

>> sigmal
sigmal =

   0.1754502   0.0504487   0.0442048   0.0629674   0.0110553   0.0088002
   0.0504487   0.1597768   0.1104895   0.0713728   0.1173385  -0.2323380
   0.0442048   0.1104895   0.1531358   0.0649473   0.0944258  -0.2071282
   0.0629674   0.0713728   0.0649473   0.4552231  -0.2905022  -0.1242331
   0.0110553   0.1173385   0.0944258  -0.2905022   0.6366109  -0.5200640
   0.0088002  -0.2323380  -0.2071282  -0.1242331  -0.5200640   1.3508875
```

```
>> mean2
mean2 =

   214.824
   130.199
   130.080
    10.273
    10.897
   140.018

>> sigma2
sigma2 =

   0.1561920   0.0177435  -0.0016388  -0.1978131  -0.0633476   0.1438412
   0.0177435   0.1451591   0.1220264   0.3078692   0.1643465  -0.3199877
  -0.0016388   0.1220264   0.1801418   0.4763395   0.1972398  -0.3969620
  -0.1978131   0.3078692   0.4763395   3.2870963   0.6319111  -1.7283589
  -0.0633476   0.1643465   0.1972398   0.6319111   0.7414307  -0.8778148
   0.1438412  -0.3199877  -0.3969620  -1.7283589  -0.8778148   1.9510753
```

5 Conclusion

We found the parameter values of the Mixture Multivariate Gaussian distribution using EM algorithm and use these values to classify our data.We classified the training data with training error=11.67 percent as there are 21 out of 180 data are wrongly classified and cross-validation error=10 percent as there are 2 out of 20 are wrongly classified.
We can use different initialisation technique to achieve more accuracy in the future.

6 Appendix

Code for EM Algorithm in Octave(similar to Matlab):
main.m file is as:

```
pkg load statistics
s=dlmread("swiss-bank.dat");
itable=s(:,2:7);
table1=itable(2:91,:);
table2=itable(102:191,:);
table=[table1;table2];
cv1=itable(92:101,:);
cv2=itable(192:201,:);
cv=[cv1;cv2];
tr1=s(2:91,1:1);
tr2=s(102:191,1:1);
tr=[tr1;tr2];
cvr1=s(92:101,1:1);
cvr2=s(192:201,1:1);
cvr=[cvr1;cvr2];
```

```
nt=50;
mean1=[0 0 0 214 129 10];
sigma1=eye(6,6);
mean2=[0 0 0 130 10 140];
sigma2=eye(6,6);
p=0.5;
n=size(table,1);
a=rand(n,1);
for i=1:nt
   a=estep(table,mean1,sigma1,mean2,sigma2,p,n);
  [p, mean1, sigma1, mean2, sigma2]= mstep(table,mean1,sigma1,mean2,sigma2,a,n);

endfor
mycvr=estep(cv,mean1,sigma1,mean2,sigma2,p,size(cv,1));
c=a<=0.5;
cv=mycvr<=0.5;
```

e-step function code:

```
function [retval] = estep (table,mean1, sigma1,mean2,sigma2,p,n)
  retval=rand(n,1);
  for i=1:n
    val1=p*mvnpdf(table(i,:),mean1',sigma1);
    val2=val1+((1-p)*mvnpdf(table(i,:),mean2',sigma2));
    if (val2==0)
      continue;
    else
      retval(i,1)=val1/val2;
    endif
  end
endfunction
```

m-step function code:

```
function [p,m1,s1,m2,s2] = mstep (table,mean1,sigma1,mean2,sigma2,a,n)
  m1=ones(6,1);
  m2=ones(6,1);
  s1=zeros(6,6);
  s2=zeros(6,6);
  epsilon=exp(-10);
  newn=sum(a);
  p=sum(a)/n;
  m1=sum(a.*table)'/newn;
  m2=sum((1-a).*table)'/(n-newn+epsilon);
  for i=1:n
    s1=s1+(a(i,1).*(table(i,:)'-mean1)*(table(i,:)-mean1'));
    s2=s2+(a(i,1).*(table(i,:)'-mean2)*(table(i,:)-mean2'));
  end
  s1=s1/newn;
  s2=s2/(n-newn+epsilon);
```

```
endfunction
```

code for K-means Clustering:

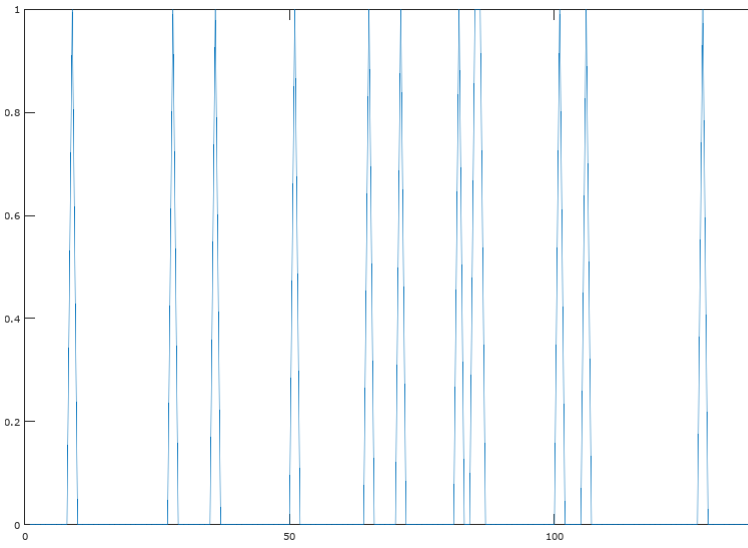
```
X=dlmread('swiss-bank.dat');
X=X(2:201,2:7);
K = 2; % 3 Centroids
initial_centroids = [3 3 3 3 3 3; 6 2 4 1 2 1];
idx = findClosestCentroids(X, initial_centroids);
centroids = computeCentroids(X, idx, K);
max_iters = 70;
[centroids, idx] = runkMeans(X, initial_centroids, max_iters, true);
fprintf('\nK-Means Done.\n\n');
fprintf(' %f %f \n' , centroids');
pause;
```

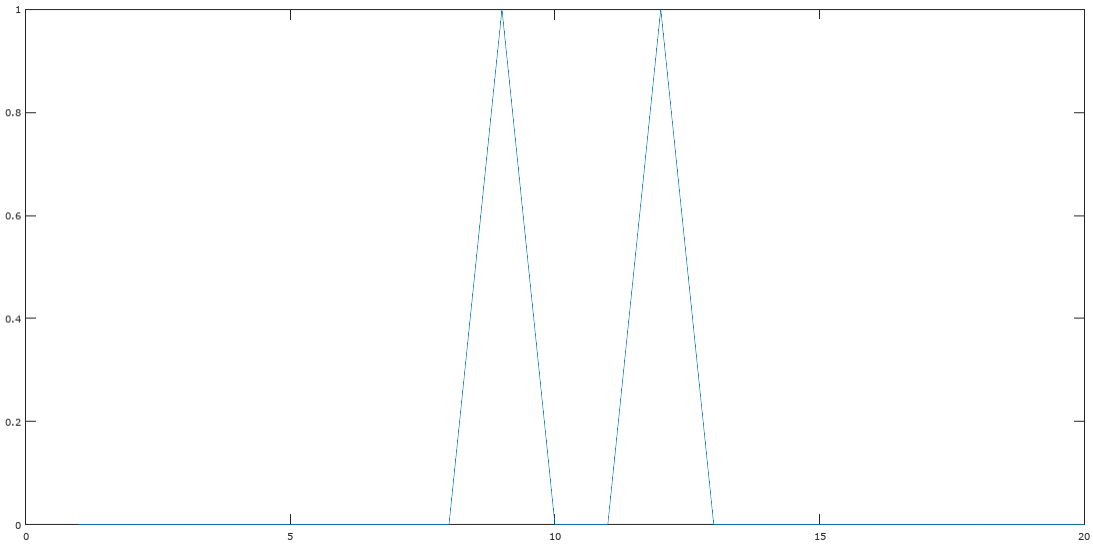
Function:

```
function centroids = computeCentroids(X, idx, K)
[m n] = size(X);
centroids = zeros(K, n);
M=zeros(K,m);
for i=1:K
   pr=(idx==i);
   s=sum(pr);
   if(s~=0)
   pr=(1/s)*pr;
   endif
  M(i,:)=pr';
end
centroids=M*X;
end
```

Graph for compared values:

```
>> sum(abs(c-tr))
ans =  21
>> sum(abs(cv-cvr))
ans =  2
```




1
0.8
0.6
0.4
0.2
0
0
5
10
15
20