CSE 5523: Homework 5
Gaussian Mixture Models

Soham Mukherjee
mukherjee.126@osu.edu

December 6, 2018

Abstract

Implementation of Expectation Maximization (EM) algorithm on Gaussian Mixture Models (GMM).

1 How to run the program

I wrote this program in python 3.6.3. Set your python interpreter so that it uses Python 3. Open terminal and type in:

```
> python EM.py
```

2 Brief Description

2.1 EStep

The responsibility that a cluster k takes for a data point i is computed in EStep. The Calculation is performed according to Equation 1

$$r_{ik} = \frac{\pi_k p\big(\boldsymbol{x_i}|\boldsymbol{\theta_k^{(t-1)}}\big)}{\sum_{k'} \pi_{k'} p\big(\boldsymbol{x_i}|\boldsymbol{\theta_{k'}^{(t-1)}}\big)} \tag{1}$$

```
def Estep(self):
        for r in range(self.data.nRows):
            for k in range(self.nClusters):
                self.resp[r][k] = np.log(self.priors[k]) + self.LogProb(r, k, self.data)
            denm = logExpSum(self.resp[r, :])
            self.resp[r, :] = np.exp(self.resp[r, :] - denm)
        pass
```

Computes the responsibility each cluster is taking for the datapoint. It is then stored in a numpy array **self.resp**[ ] [ ]

2.2 MStep

In MStep we update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}$ and $\boldsymbol{\theta}$. For $\boldsymbol{\pi}$ we have Equation 2.

$$\pi_k = \frac{1}{N}\sum_i r_{ik} = \frac{r_k}{N} \tag{2}$$

Update of $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_k}$ is according to Equation 3 and Equation 4 respectively.

$$\boldsymbol{\mu_k} = \frac{\sum_i r_{ik}\boldsymbol{x_i}}{r_k} \tag{3}$$

$$\boldsymbol{\Sigma_k} = \frac{\sum_i r_{ik}\big(\boldsymbol{x_i} - \boldsymbol{\mu_k}\big)^{\boldsymbol{2}}}{r_k} \tag{4}$$

Also note for simplicity we are not taking full covariance matrix. Only $tr(\Sigma_k)$ is taken for Homewwork 5 purpose.

```
    def Mstep(self):
        for c in range(self.nClusters):
            rik = self.resp[:, c]
            rk = np.sum(rik)
            self.priors[c] = rk / self.data.nRows
            for col in range(self.data.nCols):
                x = [self.data[i][col] for i in range(self.data.nRows)]
                mean = self.parameters[c][col][0]
                newMean = np.sum(rik[r] * x[r] for r in range(self.data.nRows)) / rk
                newVar = np.sum(rik[i] * (x[i] - mean) ** 2 for i in range(self.data.nRo
                self.parameters[c][col] = (newMean, newVar)
        pass
```

2.3 LogLikelihood

```
def LogLikelihood(self, data):
    logLikelihood = 0.0
    p = list()
    for r in range(data.nRows):
        p.clear()
        for c in range(self.nClusters):
        prob = math.log(self.priors[c]) + self.LogProb(r, c, data)
        p.append(prob)
        logLikelihood = logLikelihood + logExpSum(p)
    return logLikelihood
```

The Loglikelihood of the data is computed by Equation 5.

$$l(\theta) = \sum_{i=1}^{N} log\Big[\sum_{z_i} p\big(\boldsymbol{x_i}, \boldsymbol{z_i}|\boldsymbol{\theta}\big)\Big] \tag{5}$$

3 Output

3.1 Iteration vs. Loglikelihood

The algorithm converged after 26 iterations. The Loglikelihood vs. Iteration plot can be seen in Figure 1. As expected Loglikelihood increases in every iteration for training data until it converges. Loglikelihood for training data was found to be -2573.9405047 and for test data it was -741.0553264.

```
Training Details Iteration:      0 LogLikelihood   -3506.1627787
Test Details Iteration:          0 LogLikelihood   -920.4330070
Training Details Iteration:      1 LogLikelihood   -2806.1847444
Test Details Iteration:          1 LogLikelihood   -795.5638912
Training Details Iteration:      2 LogLikelihood   -2782.3171140
Test Details Iteration:          2 LogLikelihood   -798.1216386
Training Details Iteration:      3 LogLikelihood   -2769.7120448
Test Details Iteration:          3 LogLikelihood   -798.5413769
Training Details Iteration:      4 LogLikelihood   -2764.8449011
Test Details Iteration:          4 LogLikelihood   -797.9621768
Training Details Iteration:      5 LogLikelihood   -2758.8508516
Test Details Iteration:          5 LogLikelihood   -796.0511053
Training Details Iteration:      6 LogLikelihood   -2740.4872095
Test Details Iteration:          6 LogLikelihood   -793.1261276
Training Details Iteration:      7 LogLikelihood   -2700.7380125
Test Details Iteration:          7 LogLikelihood   -793.8158977
Training Details Iteration:      8 LogLikelihood   -2661.6800988
Test Details Iteration:          8 LogLikelihood   -785.0479108
Training Details Iteration:      9 LogLikelihood   -2637.5738499
Test Details Iteration:          9 LogLikelihood   -776.6283098
Training Details Iteration:     10 LogLikelihood   -2622.0138871
Test Details Iteration:         10 LogLikelihood   -771.7761631
Training Details Iteration:     11 LogLikelihood   -2610.1034770
Test Details Iteration:         11 LogLikelihood   -767.0683728
Training Details Iteration:     12 LogLikelihood   -2599.8534743
Test Details Iteration:         12 LogLikelihood   -762.6163787
Training Details Iteration:     13 LogLikelihood   -2585.9941129
Test Details Iteration:         13 LogLikelihood   -752.7453377
Training Details Iteration:     14 LogLikelihood   -2579.8835495
Test Details Iteration:         14 LogLikelihood   -746.2571730
Training Details Iteration:     15 LogLikelihood   -2577.2464374
Test Details Iteration:         15 LogLikelihood   -742.6401839
Training Details Iteration:     16 LogLikelihood   -2576.8856057
Test Details Iteration:         16 LogLikelihood   -741.9465162
Training Details Iteration:     17 LogLikelihood   -2576.4867971
Test Details Iteration:         17 LogLikelihood   -741.5466755
Training Details Iteration:     18 LogLikelihood   -2575.6802575
```

```
Test Details Iteration:         18 LogLikelihood   -741.2089358
Training Details Iteration:     19 LogLikelihood   -2574.4981937
Test Details Iteration:         19 LogLikelihood   -740.9916558
Training Details Iteration:     20 LogLikelihood   -2574.0553139
Test Details Iteration:         20 LogLikelihood   -741.0031824
Training Details Iteration:     21 LogLikelihood   -2573.9733118
Test Details Iteration:         21 LogLikelihood   -741.0297347
Training Details Iteration:     22 LogLikelihood   -2573.9505080
Test Details Iteration:         22 LogLikelihood   -741.0404707
Training Details Iteration:     23 LogLikelihood   -2573.9434707
Test Details Iteration:         23 LogLikelihood   -741.0475016
Training Details Iteration:     24 LogLikelihood   -2573.9412379
Test Details Iteration:         24 LogLikelihood   -741.0522722
Training Details Iteration:     25 LogLikelihood   -2573.9405047
Test Details Iteration:         25 LogLikelihood   -741.0553264
Converged after 26 iterations
```

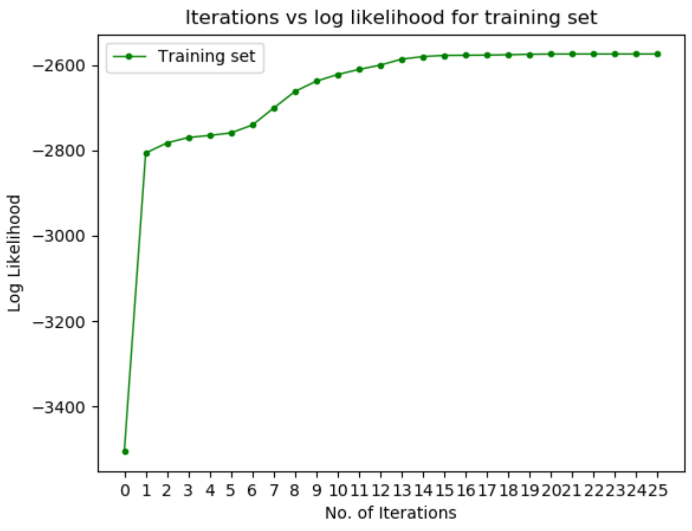
3.2 Running with Different Random Seeds

After running with different random seeds change in Loglikelihood can be seen in Figure 2. Loglikelihood is more or less constant apart from seed with 4 and 5.

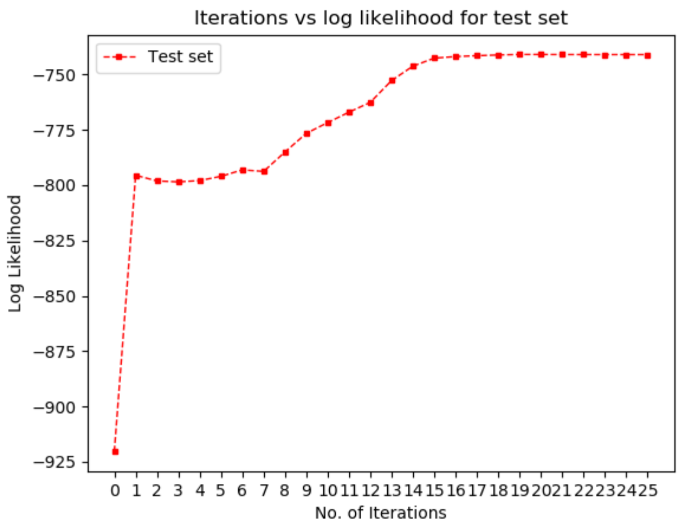
3.3 Cluster Assignment

I got $Accuracy$: 0.9436619718309859. The accuracy is computed in following manner. $IncorrectAssignments = \big(\sum_i |True_i - Assigned_i|\big)/2$.
Then, $Accuracy = \big(total - Incorr.\big)/total$

```
def accuracy(truelabel, label):
    total = 0.0
    maxlabel = np.max(label)
    info = np.zeros(maxlabel)
    trueinfo = np.zeros(maxlabel)
    for ind in range(len(label)):
        info[label[ind] - 1] = info[label[ind] - 1] + 1.0
        trueinfo[truelabel[ind] - 1] = trueinfo[truelabel[ind] - 1] + 1.0
        total += 1.0
    diff = np.sum(np.abs(trueinfo - info)) / 2
    print('True Data: ')
    for ind in range(len(info)):
        print('Cluster:', ind + 1, trueinfo[ind])
    print('Train Data: ')
    for ind in range(len(info)):
        print('Cluster:', ind + 1, info[ind])
    Accuracy = (total - diff) / total
```



(a) lIteration vs. Train Data



(b) Iteration vs. Test Data

Figure 1: Iteration vs. Train and Test Data

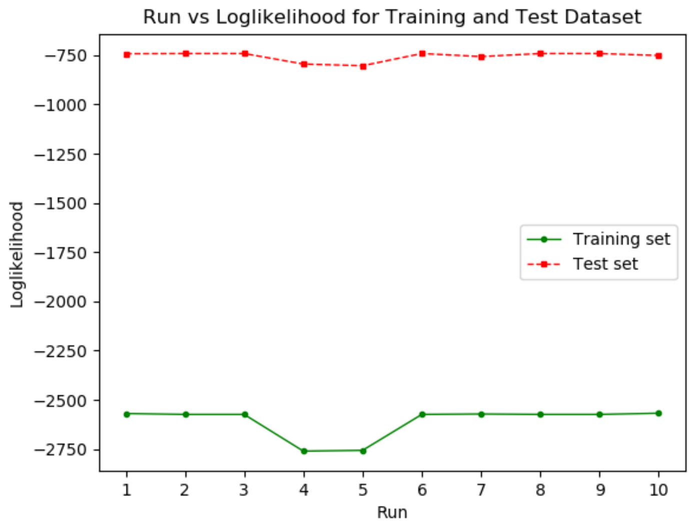


Figure 2: Loglikelihood vs. Run with Different Seeds

```
    return Accuracy
```

The output is shown below. Note that the number is not absolute. That means Cluster 1 in training data may not correspond to Cluster 1 in true data. The relative count in clusters are taken to obtain accuracy.

```
True Data:
Cluster: 1 51.0
Cluster: 2 51.0
Cluster: 3 40.0
Train Data:
Cluster: 1 43.0
Cluster: 2 56.0
Cluster: 3 43.0
Accuracy: 0.9436619718309859
```

3.4 Loglikelihood vs. Number of Clusters

As seen in Figure 3 Loglikelihood increases with increase in number of clusters. The reason being we are assigning the outliers to separate clusters. As the number of clusters increase the probability of outliers finding a cluster increases and the Loglikelihood increases consequently. But the accuracy suffers considerably. Note with Number of Clusters being 4 the Loglikelihood dips a bit. Probable reason would be bad initialization.

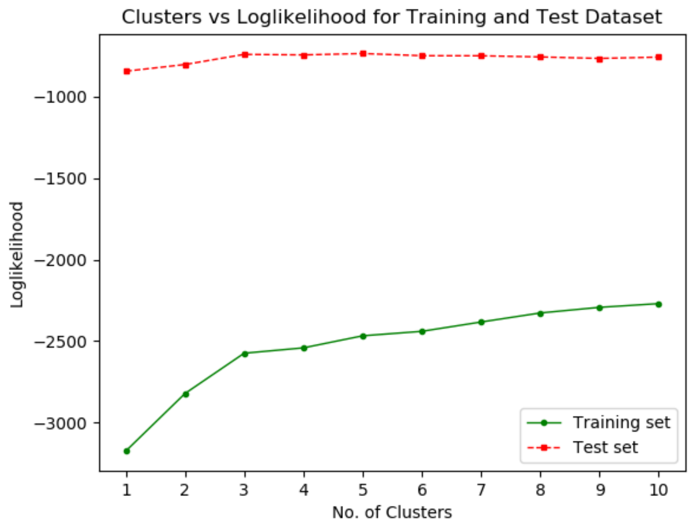


Figure 3: Loglikelihood vs. Number of Clusters

List of Figures