**Problem Set 5 Solution**

**By**

***Ananya Banerjee***

*Problem 1: K-means and Gaussian Mixture Model*

1.1 K-Means

|  |  |  |
| --- | --- | --- |
| Value of K | Mean for 20 runs of objective function | Variance for 20 runs of objective function |
| 12 | 576837.8979402108 | 4658387447015.135 |
| 18 | 670759.5787147961 | 6298857774141.11 |
| 24 | 294735.2874111416 | 1216164455048.5957 |
| 36 | 257671.03947259593 | 929521104174.4335 |
| 42 | 814415.8367550839 | 9285824172218.768 |

1.2 GMMs

|  |  |  |
| --- | --- | --- |
| Value of K | Mean of Log Likelihood | Variance of Log Likelihood |
| 12 | 6578.38780870 | 94500.435563637 |
| 18 | 6455.76867670 | 77555.489212104 |
| 24 | 6588.41958867 | 138992.540225459 |
| 36 | 6677.38289937 | 105288.689830106 |
| 42 | 6675.82818901 | 93445.541132065 |

1.3 For k=36, I would prefer Gaussian Mixture Model.

1.4

K-Means++

|  |  |  |
| --- | --- | --- |
| Value of K | Mean of Log Likelihood | Variance of Log Likelihood |
| 12 | 138956249.739 | 231671475721.8249 |
| 18 | 139267702.392 | 227210290205.0353 |
| 24 | 139599033.511 | 396224192952.43164 |
| 36 | 140027690.098 | 324104046574.8119 |
| 42 | 140430132.674 | 422592345433.8253 |

K-Means ++ initialization with GMM

|  |  |  |
| --- | --- | --- |
| Value of K | Mean of Log Likelihood | Variance of Log Likelihood |
| 12 | -460.32228867201104 | 6474618.740709777 |
| 18 | -715.1208988554038 | 10733042.450439887 |
| 24 | -504.7603544900611 | 4069450.999962587 |
| 36 | -452.28061566237665 | 2209728.8754221676 |
| 42 | -695.5883287130567 | 4498314.797822158 |

1.5 We can change The EM algorithm update like on slide 26 to see how it changes in this case. The (x^(i) – mu^(t+1)) multiplied by its transpose will give a m\*m matrix (m=dim), which is positive definite. But, if we will just multiply it with identity matrix , we will get a diagonal matrix ,as all non-diagonal elements will be 0 and we can find derivative of let’s say the jth component to proceed ahead.

*Problem 2: Logistic Regression*

2.1 The accuracy on the test set is 84.7%. Without any form of regularization in logistic regression, the weights may not converge (even when though the predicted label for each data point effectively does) because when the data is linearly separable, there is no maxima and thus the weights go to infinity, thereby never converging and causing overfitting. Thus penalizing high weights can help prevent overfitting.

2.2 For logistic regression with l2 penalty, the chosen value of C is C=0.000001 and the corresponding weight and bias is:

W=[array([[ 1.37244242e-03, 1.15575127e-03, 8.34953760e-04,

9.96897233e-08, 7.96200017e-10, 5.59499729e-08,

5.70971191e-08, 1.67842359e-07, 5.99520964e-07,

5.62714807e-06, 3.19021919e-07, 3.65598810e-07,

4.73349284e-07, 9.57081963e-07, 4.04386134e-07,

2.09388776e-04, 6.77213371e-06, 8.79858405e-06,

-4.73182855e-05, 3.76535088e-06, 3.13986850e-05,

3.88056539e-06]])]

B= array([[0.00119636]])

Accuracy for this C=0.000001 on test set is 72.8%. Best accuracy on validation set using l2 penalty is 74.13%.

2.3 For logistic regression with l1 penalty, the chosen value of C is C=1 and the corresponding weight and bias is:

W= [[ 0.00354773, -0.00757317, -0.00908208, 0. , 0. ,

0. , 0. , 0. , 0. , 0. ,

0. , 0. , 0. , 0. , 0. ,

0.02676081, 0. , 0. , 0.84476241, 0. ,

3.43346369, 0. ])]

B=array([[0.])

The accuracy for C=1 on test set is 83.05%. Best accuracy on validation set using l1 penalty is 86.2%.

2.4 l1-norm tends to produce sparse weight as shown above.