Programming Assignment 3

Manthan Thakar

Note: More information about the code can be found in README.md file

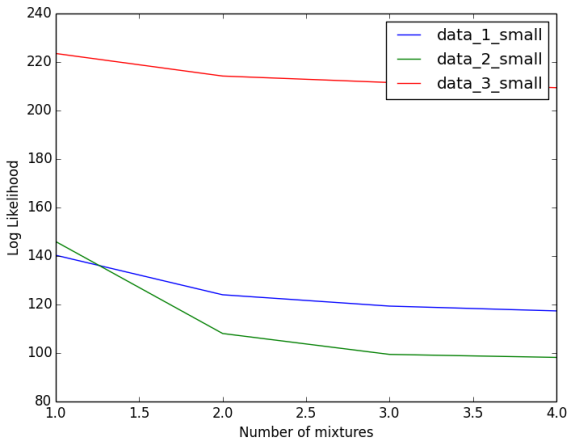
1. EM Algorithm

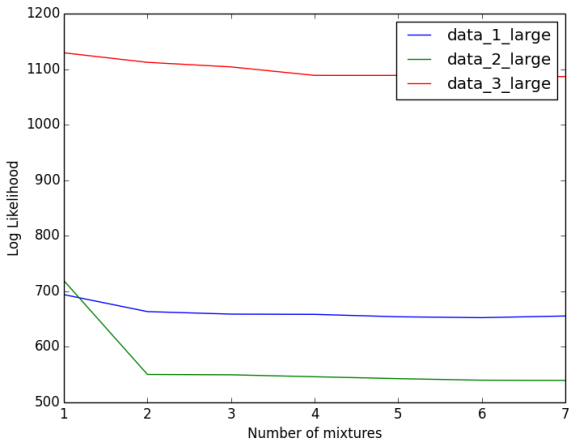
The implementation of EM algorithm for Gaussian Mixture Models can be found in *models.py* file.

Varying number of mixtures:

Following figures show the effect on log likelihood as the number of mixtures are varied on both small and big datasets. Note that the likelihood numbers are positive because it makes for better graph, but in calculations the likelihood values are negative (i.e. positive log likelihood function used). Moreover, for both plots the likelihood of the training dataset is plotted.

As we can see from the following plots, the likelihood drops steeply as the number of mixtures is increased to 2 from 1, but after that there's not a significant change in the likelihood. This is expected, since the using only one mixture to fit the data wouldn't be able to model the kind of data we have at hand.





Initial mixture parameters:

Initially the parameters for each mixture are set as following:

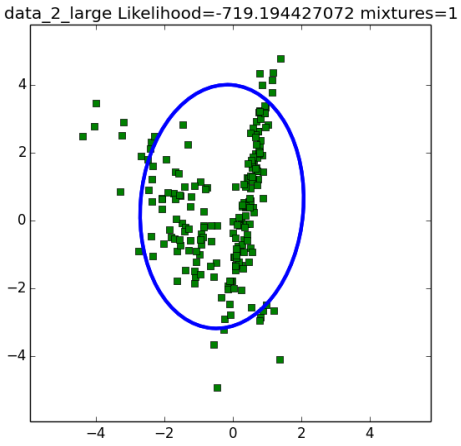
- **mu**: The value for mixture means are selected randomly
- **sigma**: The covariance matrix is chosen to be an identity matrix in the beginning
- **pi:** The weights for each mixture is assigned a uniform value = 1/m, where m is the number of mixtures

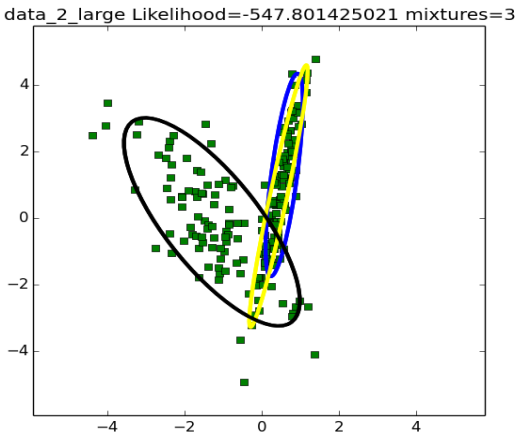
After multiple runs it was observed that when the value of pi is chosen as random the algorithm is more likely to run into the problem of having infinity values in mean and covariance vectors. After assigning uniform values for pi, the algorithm is less likely to run into that error.

Convergence parameter:

Two different convergence parameters are used in this implementation of EM. They are as follows:

- **Likelihood change threshold**: Likelihood change threshold is used to check the difference between previous and current likelihood and if it is less than the given threshold, the algorithm terminates. Using this measure, it was observed that most of the training sessions converged between 30 to 80 iterations.
- **Maximum Iterations**: If the number of iterations exceeds maximum iterations specified, the algorithm terminates. By default, this value is kept 100, since most of the datasets converge with respect to the likelihood values under 100 iterations.





Plots above show the Gaussian mixtures fit by the EM algorithm implementation for data_2_large dataset with number of mixtures 1 and 3. It can be seen that the log likelihood for mixtures = 1 (-719.19) is smaller than mixtures = 3 log likelihood (-547.80).

2. Variations

2.1 Diagonal covariance matrix

For diagonal variance GMM we use the following equation to update covariances:

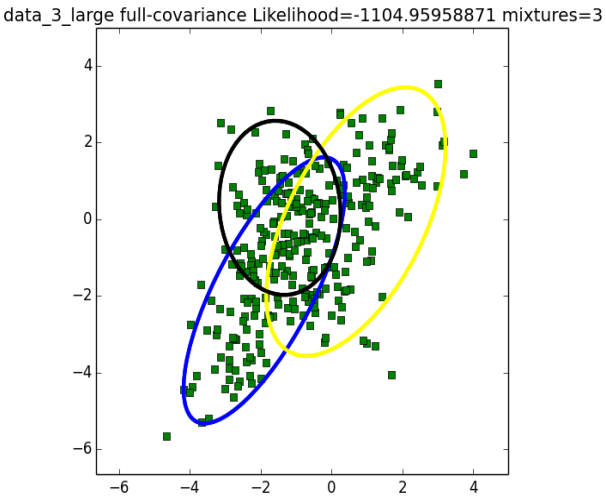
$$\hat{\sigma}_{ik}^2 = \frac{\sum_n p_{kn}(x_{in} - \hat{\mu}_{ik})^2}{\sum_n p_{kn}}, \quad i = 1, \ldots, D$$

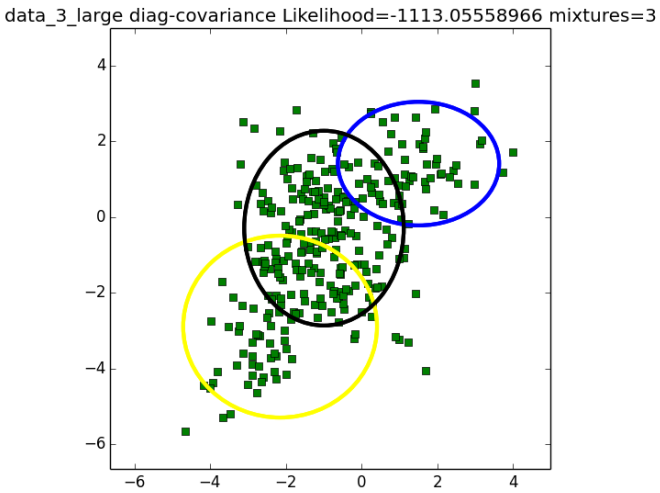
Whereas for full covariance GMM we used following:

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_n p_{kn}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^T}{\sum_n p_{kn}}$$

In this variant, only the diagonal values of the covariance matrices for Gaussians in the mixture are updated and the off-diagonal values are zero. Since, the diagonal of the covariance matrix corresponds to the variance of the features, the Gaussians fit by this variant only scale in the direction of the features i.e. horizontal and vertical in the case of 2D features. Whereas, the Gaussians with full covariance matrix fit Gaussian that are tilted. The

difference can be seen visually in the plots below for the data_3_large and number of mixtures = 3

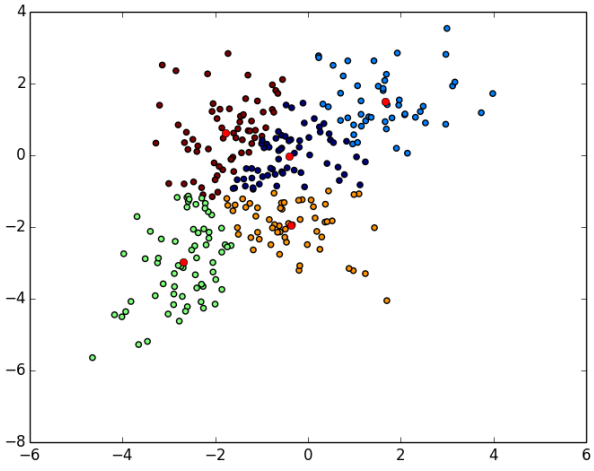




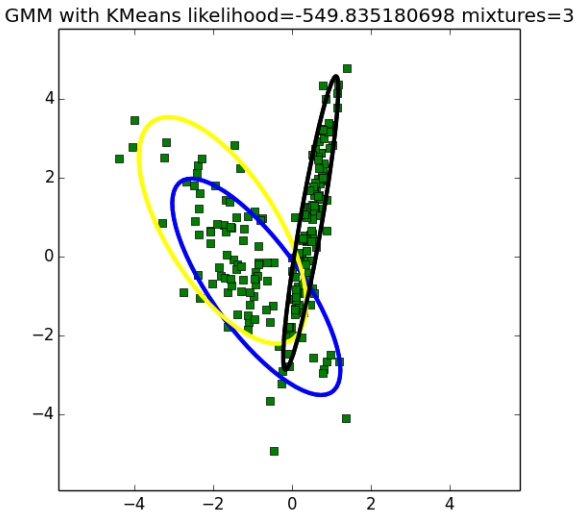
The plots above show the difference between full-covariance GMM (left) and diagonal-covariance GMM (right). The diagonal-covariance Gaussians are only stretched along x and y axis but they don't scale in both x and y direction together. It can also be noted that in this example, the log likelihood for full-covariance GMM is -1104 and for diagonal-covariance GMM it is -1113. This means that full-covariance GMM is slightly better in terms of maximizing log likelihood.

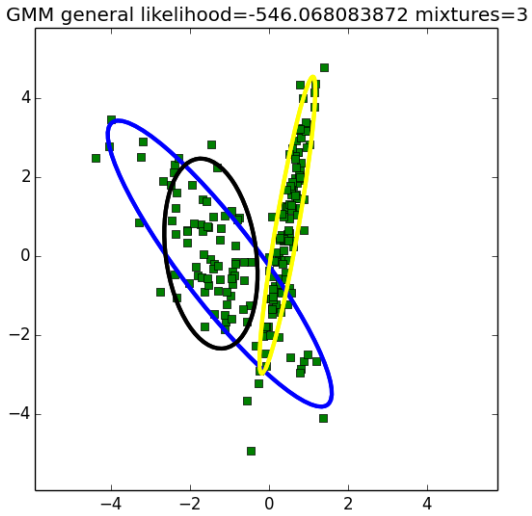
2.2 K-Means

The k-means algorithm is implemented and the final centroids obtained using k-means are used as initial means for mixtures in the Gaussian mixture model. The following plot shows the means obtained by running k-means algorithm on data_3_large.



Difference in Gaussians fitted by general GMM and GMM with kmeans initialization

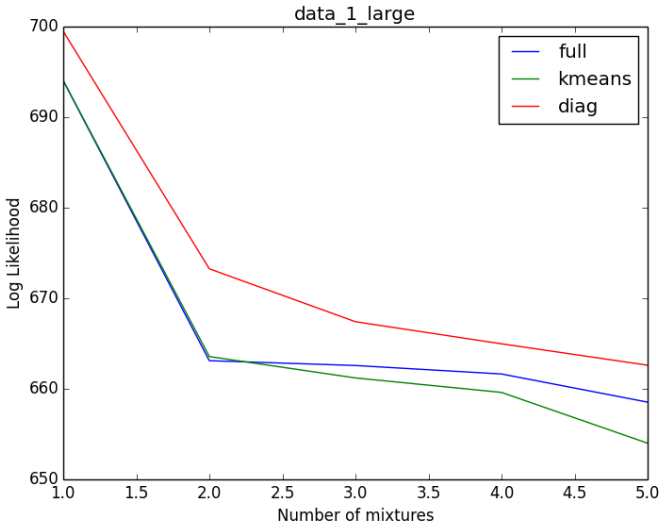


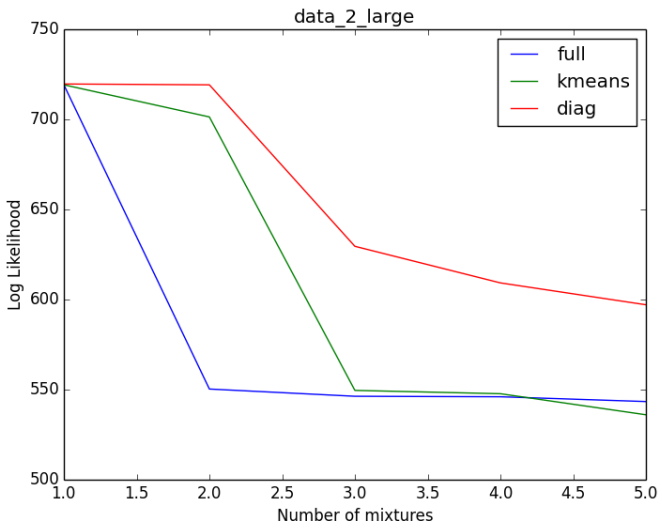


In the plot above we can observe that the Gaussians fit by general GMM and GMM with K-means give slightly different results.

2.3 Analysis

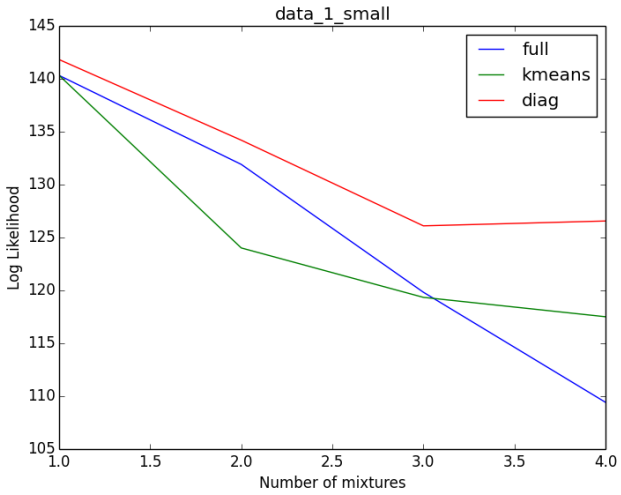
Changing number of components:

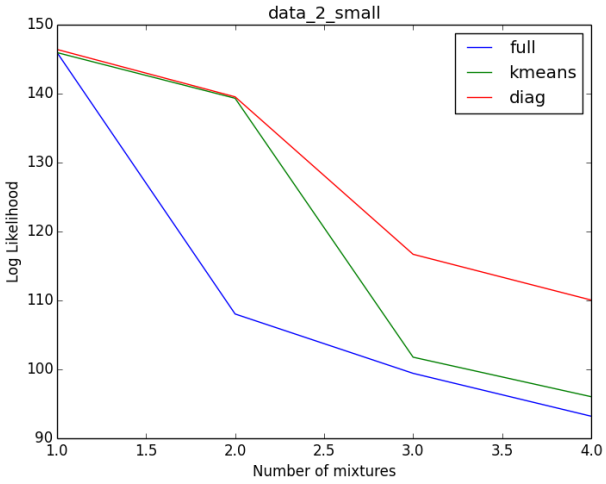




The plots above show the difference in negative log likelihood as the number of mixtures changes for *data_1_large* and *data_2_large*. We can observe that for all the variants negative log likelihood invariably decreases (increases for positive likelihood). Here, full denotes EM with full-covariance, kmeans denotes EM with full-covariance where initial means are k-means centroids and diag denotes diagonal matrix EM. The decrease in likelihood is similar for kmeans and full, although EM with initial means as kmeans centroids reaches the smallest value while the other variants don't.

Moreover, the diag version of EM has similar patterns in change of negative log likelihood as number of mixtures increases, but the likelihood values are bigger than other two variants. This may be due to diag variant having fewer parameters than other variants to build the model.





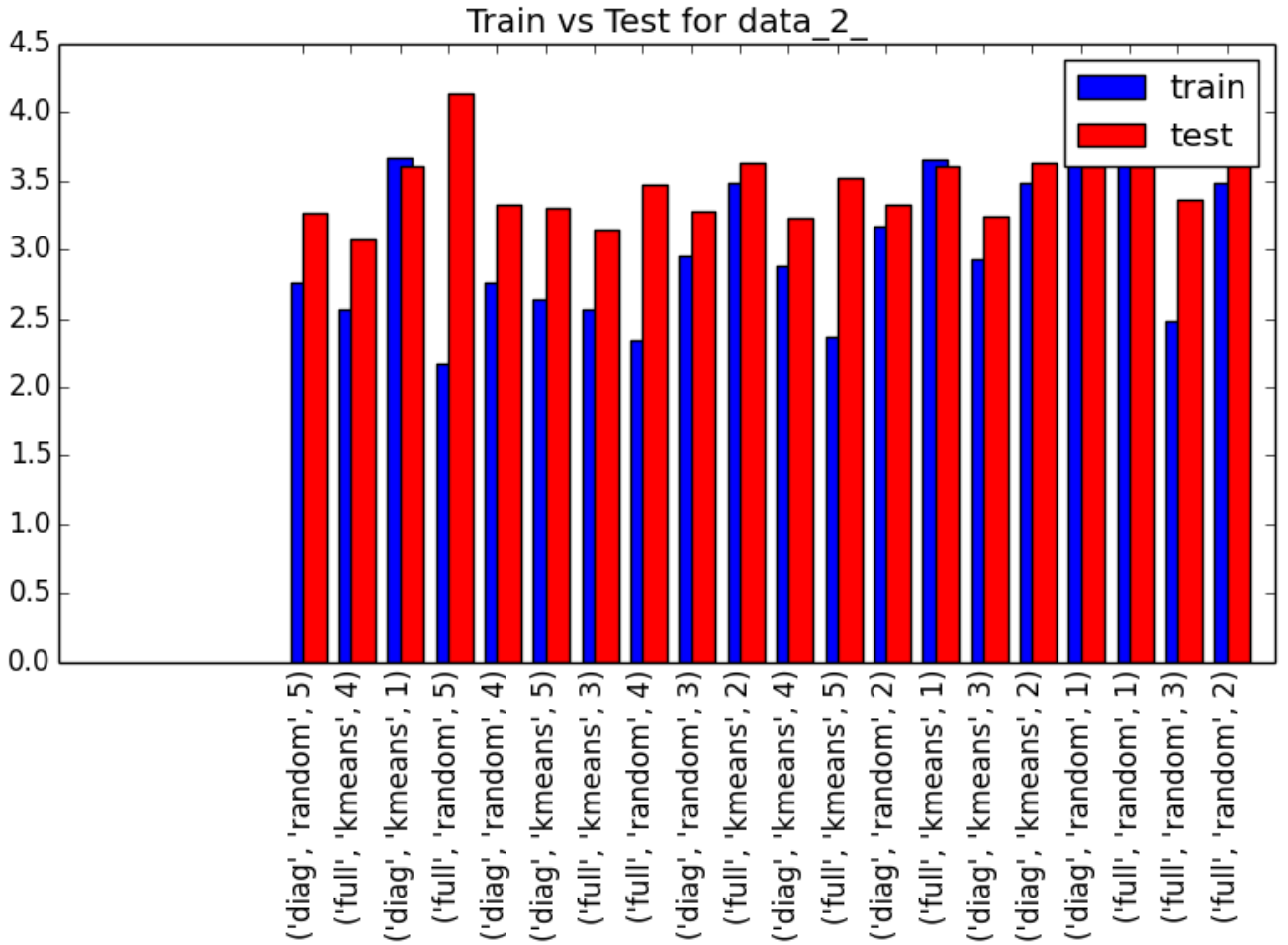
We can observe similar patterns in the plots of negative log likelihood vs number of mixtures for small datasets as well.

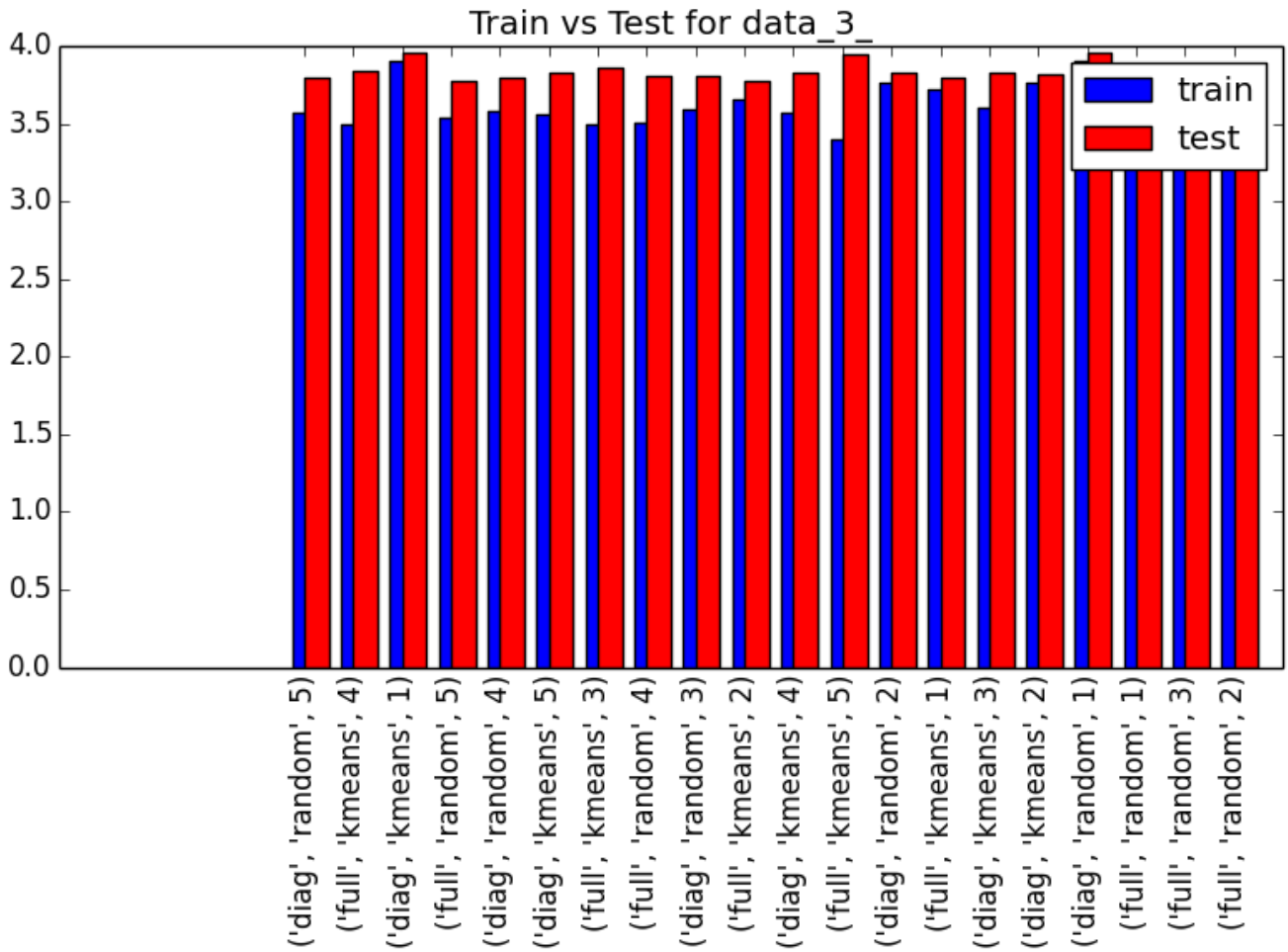
Initial parameters:

As seen in the plots above, GMM with randomly initialized mu values has bigger values for negative log likelihood than GMM with kmeans centroids as initial means. Moreover, as shown in section 2.2, sometimes these two variants may yield slightly different Gaussians for the same datasets and same number of mixtures.

3. Model Selection

3.1 Candidate models

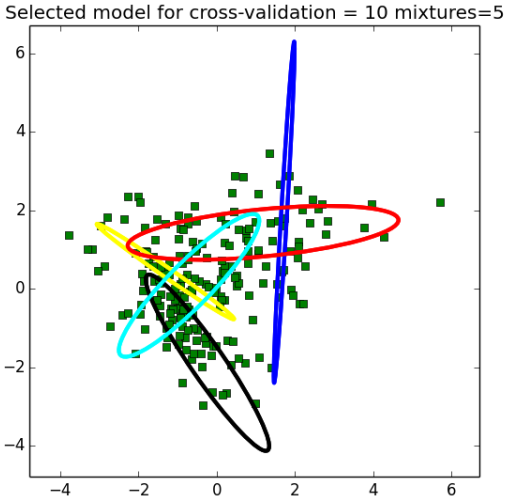

Train vs Test for data_2_
train
test
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
('diag', 'random', 5)
('full', 'kmeans', 4)
('diag', 'kmeans', 1)
('full', 'random', 5)
('diag', 'random', 4)
('diag', 'kmeans', 5)
('full', 'kmeans', 3)
('full', 'random', 4)
('diag', 'random', 3)
('full', 'kmeans', 2)
('diag', 'kmeans', 4)
('full', 'kmeans', 5)
('diag', 'random', 2)
('full', 'kmeans', 1)
('diag', 'kmeans', 3)
('diag', 'kmeans', 2)
('diag', 'random', 1)
('full', 'random', 1)
('full', 'random', 3)
('full', 'random', 2)

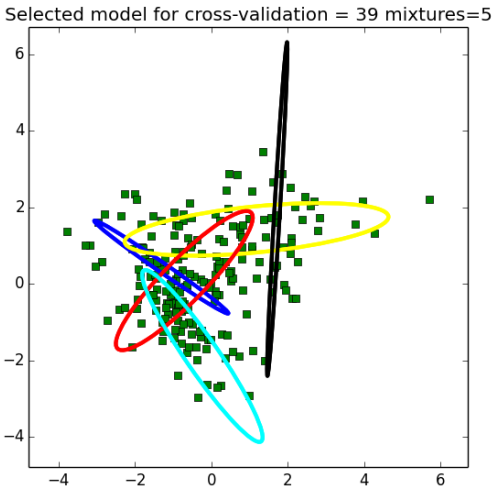

Train vs Test for data_3_
train
test
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
('diag', 'random', 5)
('full', 'kmeans', 4)
('diag', 'kmeans', 1)
('full', 'random', 5)
('diag', 'random', 4)
('diag', 'kmeans', 5)
('full', 'kmeans', 3)
('full', 'random', 4)
('diag', 'random', 3)
('full', 'kmeans', 2)
('diag', 'kmeans', 4)
('full', 'kmeans', 5)
('diag', 'random', 2)
('full', 'kmeans', 1)
('diag', 'kmeans', 3)
('diag', 'kmeans', 2)
('diag', 'random', 1)
('full', 'random', 1)
('full', 'random', 3)
('full', 'random', 2)

The above plots show the negative log likelihood of data_1, data_2 and data_3 with on training set as well as test set. The x-axis denotes the parameters used to train the model. We see that full-covariance matrix with number of mixtures = 5 is the best model for all datasets based on average log likelihood of training examples, but it performs poorly on test datasets. Therefore, average log likelihood might not be a good measure to select models.

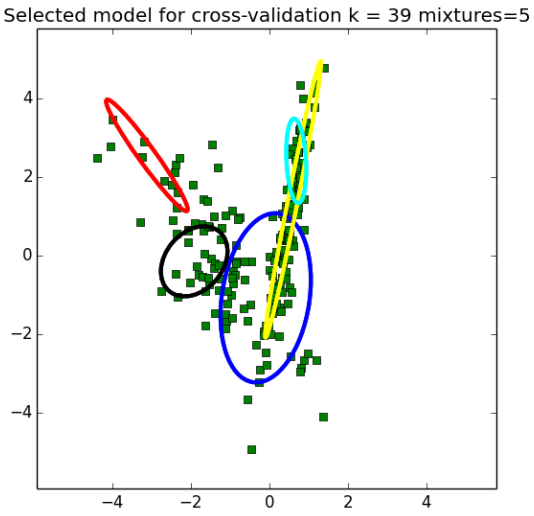
We also observe that, the difference between test likelihood and train likelihood is smaller when number of mixtures is 2.

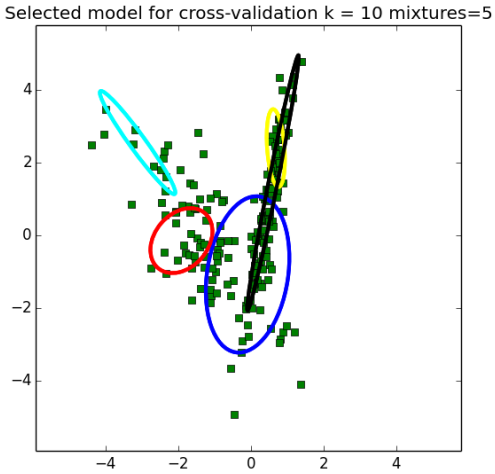
3.2 And 3.3



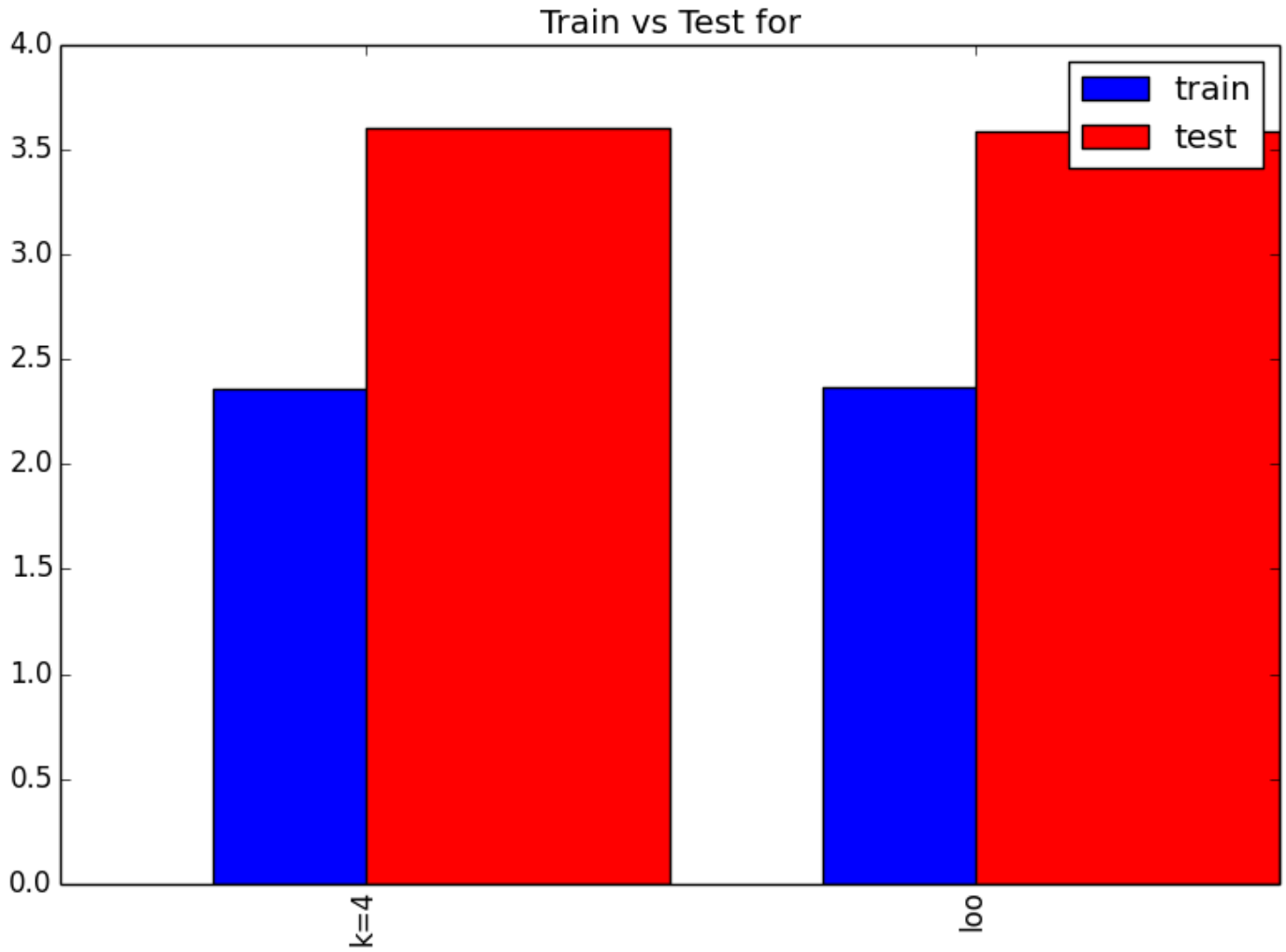


The above plots show the best model selected by 10-fold cross validation and leave one out cross validation for data 1.





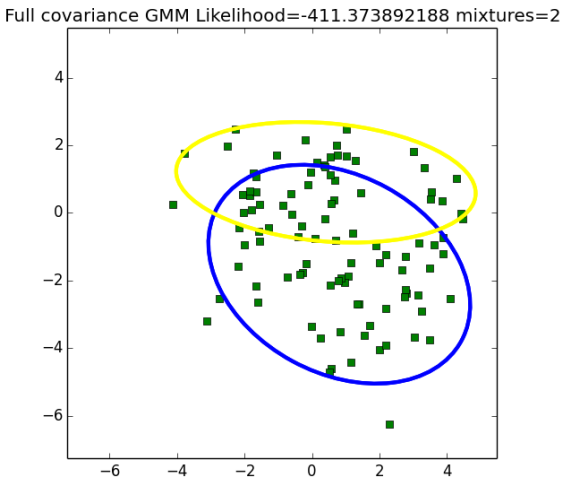
The above plots show the best model selected by 10-fold cross validation and leave one out cross validation for data 2.

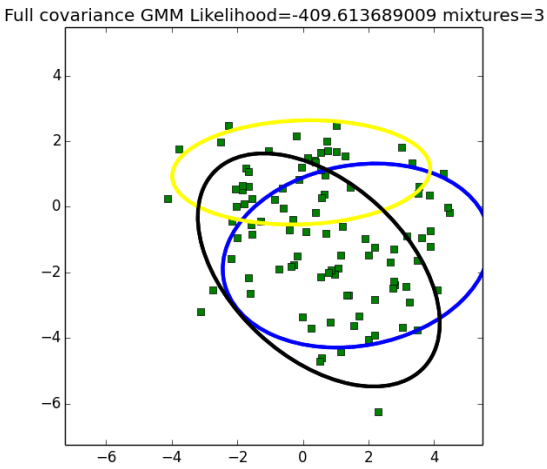


The above plot shows the comparison between best model chosen by K-fold cross validation and leave one out cross validation. There's no significant difference in both models. This can be compared with the performances reported in 3.1 to compare them with other models.

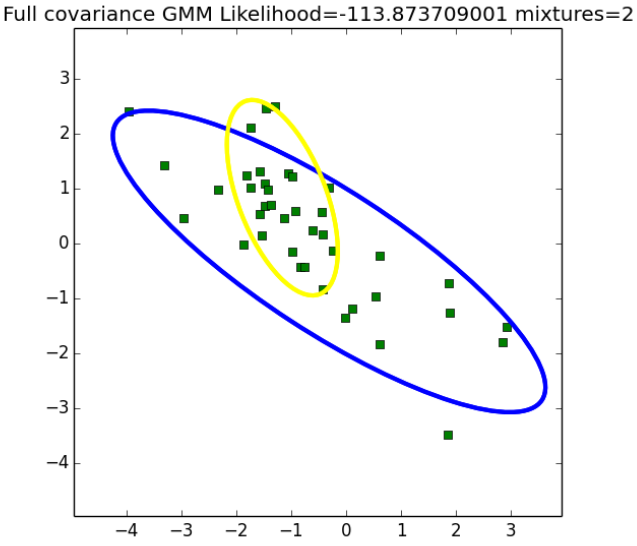
4. Predictions

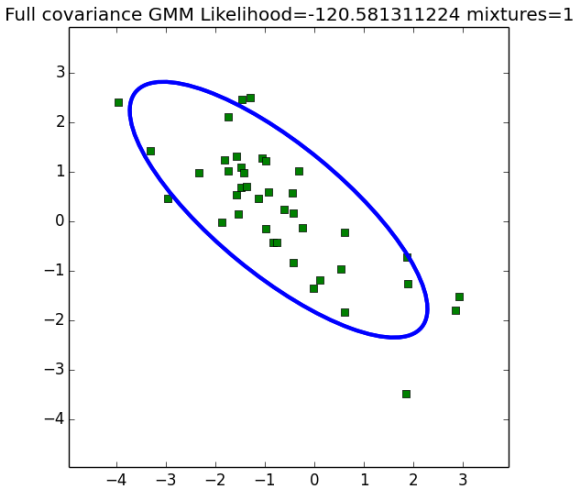
For mystery_1 dataset we choose mixtures = 2 as the parameter.





We can see the difference between number of mixtures = 2 and 3 in the plots above. Visually, it can be seen that with two mixtures, it's easy to separate the data. Although mixtures=3 has bigger likelihood, it over fits the data and lots of points end up in multiple gaussians.





Similarly, for mystery2 dataset we can keep mixtures = 1.