Joel Kemp
Machine Learning Spring 2011
Homework 5 README

Introduction

The system segments a dataset based on the Gaussian Mixture Models with Expectation Maximization (GMMEM) clustering method. The system accepts a Comma-Separated-Value (CSV) filename of datapoints and a positive number of clusters, *m*. This segmentation process involves performing an initial K-means segmentation on the passed dataset of coordinates to get a starting distribution of points about the clusters. Statistics/paremeters (namely: mean, covariance, and mixture coefficients) are then generated about each cluster to form a model. This is known as the initialization phase.

Along with the aforementioned model, a multivariate normal density function is used to generate an assignment score that represents how responsible a cluster should be for each point; this basically dictates cluster membership about the dataset of points. This is known as the expectation phase.

After the assignment score analysis, the original model is revised based on the generated scores. In other words, the parameters are recomputed to reflect the changing point membership about the clusters. We can then use this model to generate the log likelihood value. The log likelihood represents the probability that the segmented data was accurately described by the refined model. This is known as the maximization phase.

The aforementioned processes (namely: expectation and maximization) will continue until either the parameters or the log likelihood converges/settles.

The output of this implementation consists of several lines: the first line is the resulting log likelihood after convergence, the next set of *m* lines are the centroid locations, and the following lines are covariance matrices for each cluster.

The system was built using Matlab 7.10.0 R2010a on a Windows 7 machine. Matlab sometimes produces runtime errors for the multivariate normal distribution computation when the covariance matrices are “ill-conditioned.” The covariance matrices used were generated from a predefined covariance function in Matlab. Hence, the error seems unlikely to be programmer-induced.

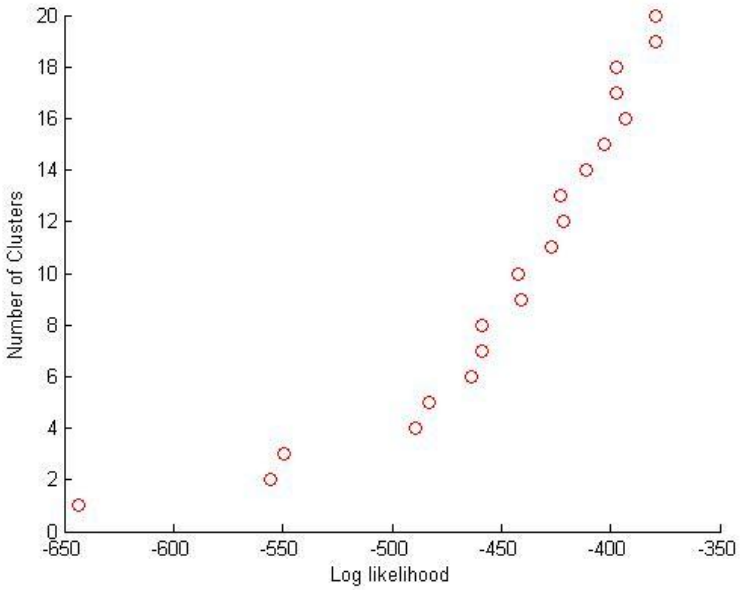
A workaround was implemented where a tiny (one millionth of a unit) was added to the elements of the ill-conditioned covariance matrix which seemed to “fix” the problem. This seemingly uncontrollable error occurs sporadically but the results presented in this readme are gathered from successful runs of the application.

Model Likelihood

The GMMEM segmentation scheme was run on values of m ranging from 1 to 20. The resulting log likelihoods of the models generated for each cluster value is listed in the table below.

m	Log Likelihood
1	-643.450828
2	-555.746736
3	-549.170202
4	-489.138160
5	-483.167176
6	-463.457569
7	-458.547375
8	-458.549024
9	-441.067948
10	-442.171630
11	-427.122970
12	-421.402300
13	-422.645824
14	-411.161234
15	-402.876283
16	-393.030990
17	-397.533703
18	-397.073829
19	-379.594197
20	-379.467493

Table 1- Log Likelihood Values

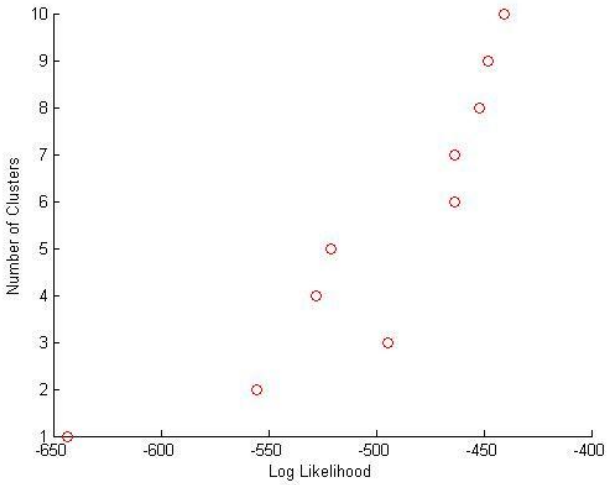


From the table and figure above, we can see that the log likelihood estimations grow larger (more positive) as the number of clusters increases. Since the log likelihood represents the probability that the datapoints correctly belonged to the model, we expect the likelihood to increase as a result of overfitting the model to the training set.

Reusing the first 10 values of *m*, we obtain the resulting table of log likelihoods:

m	Log Likelihood
1	-643.450828
2	-555.746735
3	-494.714438
4	-527.726021
5	-520.869116
6	-463.457569
7	-463.738513
8	-451.853935
9	-447.766072
10	-440.289166

Table 2 - Log Likelihoods of Reused Cluster Numbers



From the table and figure above, we notice a difference in the resulting log likelihoods. This is a result of the randomness about using the multivariate normal distribution in the expectation and maximization phases of the segmentation process. There is an unexpected dip in the results when $m = 3$; this drop in the likelihood was clearly not evident when using 20 different values of m.

Convergence in the system is strictly in regards to the resulting log likelihood of a model. For every iteration of the algorithm, we compare the difference between log likelihoods of the current and previous iterations (except in the first iteration, of course). If the difference is less than a very small (randomly chosen) threshold value, then the algorithm is said to converge.

In regards to how quickly the segmentation algorithm converges for the different values of m, we can observe the results in the table below:

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Steps	2	5	14	35	43	33	53	55	17	55	48	26	48	48	25	56	16	2	5	14

Table 3 - Convergence Steps

From the table above, there doesn't seem to be an obvious pattern about the relationship between the number of steps required for convergence and the number of clusters. However, it can be seen from values of m ranging from 1 to 7 that there is an increase in the number of steps needed for log likelihood convergence. Perhaps no pattern exists due to the strict definition of convergence in the implementation.

When looking at tables 1 and 2, it can be observed that the "ideal" number of clusters is anywhere from 4 to 8 clusters. Ultimately, the "ideal" cluster choice should avoid the area of overfitting (apparent when m increases beyond the value of 8) while still resulting in a relatively high probability that the model generated the datapoints; i.e., while still being a correct model of parameters.

File List

1. **GMMEMSegment.m:** The entry point for the application. This function/file performs the entire segmentation process – calling on the other files for the subprocesses of initialization, expectation, maximization, and computing the log likelihood of the model.
2. **GMMEMInitialize.m:** Performs Kmeans segmentation on the passed data matrix X with an initial number of clusters, *m*. The function then computes an initial set of parameters (mean, covariance, and mixture coefficient) for each cluster.
3. **GMMEMExpectation.m:** Computes the cluster assignment scores for each point in the data matrix.
4. **GMMEMMaximization.m:** Recomputes the model parameters based on the assignment scores passed to this function.
5. **GMMEMLogLikelihood:** Computes the sum of the log likelihoods that each point was generated by the model parameters.

Using the program:

The program can be run from the Matlab console with the command:

```
GMMEMSegment 'clustering.csv' 4
```

This command will produce the following output:

```
Log Likelihood: -489.138160
Cluster 1 Centroid: (1.059064e+001,2.495193e+000)
Cluster 2 Centroid: (3.030133e+000,3.074845e+000)
Cluster 3 Centroid: (1.039045e+001,9.341649e+000)
Cluster 4 Centroid: (-7.742455e-002,2.575636e-002)
Cluster 1 Covariance:
[3.837212 1.372721]
[1.372721 3.414670]
Cluster 2 Covariance:
[0.079829 0.010586]
[0.010586 0.097336]
Cluster 3 Covariance:
[4.826259 -0.100643]
[-0.100643 6.927273]
Cluster 4 Covariance:
[0.776145 -0.022853]
[-0.022853 0.673020]
```

If any problems persist, please contact the author at mrjoelkemp@gmail.com.