Spring 2017 EECE 5644 Homework #4

Prof: Jaume Coll-Font TA: Aziz Kocanaogullari, Zhiqiang Tao

Make sure you read the problem statements and answer the questions. Submit the code online. Since this homework requires a lot of plotting, write your code modularly so that making plots with different datasets is easy. This homework will be challenging, so I suggest you start early.

4.1 (60 pts) Expectation Maximization

(a) (20 pts) Implement the Expectation Maximization (EM) algorithm for Gaussian Mixture clustering. The implementation MUST be a function that takes AT LEAST the following inputs:

i. inputData: nSamples x dDimensions array with the data to be clustered
ii. numberOfClusters: number of cluster for algorithm
iii. stopTolerance: parameter for convergence criteria
iv. numberOfRuns: number of times the algorithm will run with random initializations

Notice that I say AT LEAST. You may want to add some optional parameters if you want to give the user more control over the algorithm. The function should output the results for best EM clustering. The output should be AT LEAST:

i. clusterParameters: numberOfClusters x 1 struct array with the Gaussian mixture parameters:
   - .mu - mean of the Gaussian component
   - .covariance - covariance of the Gaussian component
   - .prior - prior of the Gaussian component

Notice that I say AT LEAST. I suggest you also have the following outputs since they will make your plotting easier I think:

i. estimatedLabels: nSamples x 1 vector with labels based on maximum probability. Since EM is a soft clustering algorithm, its output are just densities for each cluster in the mixture model
ii. logLikelihood: 1 x numberOfIterations vector with the log-likelihood as a function of iteration number
iii. costVsComplexity: 1 x maxNumberOfClusters vector with BIC criteria as a function of number of clusters (more on this below)

This algorithm is tricky to get right. I strongly suggest you run a simple test with 4 2D Gaussian clusters very well separated as a way to test your implementation.

There are a couple of details you need to be mindful:

i. Initialization: your algorithm can end up in a local maximum. This is why one of the inputs is the number of times you will run EM with random initializations. By random initial conditions, I mean, select the initial K mean vectors by randomly selecting K initial data points. You can select the initial K covariances to be the identity matrix (the selection of the initial covariances is not as critical as the initial means) and the initial priors to be uniform. Then, run your algorithm until convergence starting from each of these initial conditions. Finally, pick the one that results in the highest log-likelihood.

ii. Convergence: from a practical viewpoint, you need to decide when it is no longer worth iterating. You could stop if the increase in log-likelihood between the last two iterations is less than 0.001% of the change in likelihood between the current value and the log-likelihood value after the very first iteration of the algorithm. For these data sets, you can also impose a maximum number of iterations to halt the algorithm (e.g., 50) if it gets that far and still has not converged.

iii. Singular solutions: the likelihood can go to infinity if the determinant of the covariance matrix goes to zero (for instance, if any individual $\sigma_{ii} \rightarrow 0$). You need to implement some scheme to prevent such singular (and useless) solutions. This is typically only a problem in practice on small data sets, or when the number of components K is large enough that one or more of the mixture components ends up with very few data points. One somewhat ad hoc workaround (that works well in practice) is to constrain all covariance diagonal terms during the M-step to be greater than some small threshold δ (10^{-4} times the variance for that dimension, as calculated on the whole data). A simple way to do this is to calculate the Σ's in the standard M-step manner and then check each diagonal entry: if any are less than the threshold, then replace them with the threshold. This fixes the problem with dealing singular matrices. However, it still gives a spurious cluster solution. Alternatively, if you are running many trials and this happens on one of them, just return no solution for that trial and move on to the next run of EM with (hopefully) a better initial condition.

(b) (10 pts) For dataset1.mat, using the true number of clusters (K=2), run the EM algorithm. Make a 1x3 figure (with subplot) to generate the following plots: (a) true clustering, (b) results of clustering (choose the labels according to maximum probability) with contours showing the gaussian mixture and (c) log-likelihood as a function of iteration number. How does EM perform when you compare it to K-means?

(c) (10 pts) For dataset2.mat, using the true number of clusters (K=3), run the EM algorithm. Make a 1x3 figure (with subplot) to generate the following plots: (a) true clustering, (b) results of clustering (choose the labels according to maximum probability) with contours showing the gaussian mixture and (c) log-likelihood as a function of iteration number. How does EM perform when you compare it to K-means?

(d) (10 pts) For dataset3.mat, using the true number of clusters (K=2), run the EM algorithm. Make a 1x3 figure (with subplot) to generate the following plots: (a) true clustering, (b) results of clustering (choose the labels according to maximum probability) with contours showing the gaussian mixture and (c) log-likelihood as a function of iteration number. How does EM perform when you compare it to K-means? Does EM find the clusters?

(e) (10 pts) For dataset4.mat, using the true number of clusters (K=3), run the EM algorithm. Make a 1x3 figure (with subplot) to generate the following plots: (a) true clustering, (b) results of clustering (choose the labels according to maximum probability) with contours showing the gaussian mixture and (c) log-likelihood as a function of iteration number. How does EM perform when you compare it to K-means? The scatter plot must be done with dimension 1 on the horizontal axis and dimension 2 on the vertical axis.

4.2 (40 pts) BIC criteria to penalize model complexity There are a number of different methods for trying to find K automatically from the data. Essentially, we are trying to find the K that gives the best predictions on new data. A very simple and useful approach for doing this is something called the BIC criterion, i.e., choose K such that

$$L(D|\boldsymbol{\theta}) - (p_k/2)\log n$$

is maximized, with $L(D|\boldsymbol{\theta})$ is the value of the log-likelihood as found by EM using data D with K components in the mixture, p_k is the total number of parameters in your mixture model, and n is the number of data points in the dataset D. There is a rather general theoretical justification for this BIC criterion which loosely speaking says that as K increases we should penalize the log-likelihood of the model with K components according to its increased complexity, and $p_k/2\log n$ can be shown to be a good approximation to a true Bayesian penalty term. This is just like you have seen in class.

To write a Matlab program to do this is very simple given the EM algorithm that you have already written. Simply have a loop that runs the EM algorithm with values of K going from 1 to some maximum K chosen by you. Note that K=1 is important: it might be the case that the data are best explained by a single Gaussian (for the case of K=1 you obviously don't need to run EM). Your code should return a list, for each value of K, of both the log-likelihood (which should increase monotonically as K increases), the BIC criterion as defined above, and the K value that maximizes the BIC criterion. This is why I told you to add more outputs in your EM function. You may want to make it very general so you don't have to write so much code. For some of the smaller data sets, you may find that as K increases you have more problems both with local maxima of the likelihood function and with singular solutions (so you may need to either make the maximum K smaller, and/or run a larger number of random restarts).

(a) (5 pts) For dataset1.mat, plot BIC as a function of cluster size (from K=1 to K=10). Does the maximum match the true number of clusters?

(b) (5 pts) For dataset2.mat, plot BIC as a function of cluster size (from K=1 to K=10). Does the maximum match the true number of clusters?

(c) (10 pts) For dataset3.mat, plot BIC as a function of cluster size (from K=1 to K=10). Does the maximum match the true number of clusters?

(d) (10 pts) For dataset4.mat, plot BIC as a function of cluster size (from K=1 to K=10). Does the maximum match the true number of clusters?

(e) (10 pts) For dataset5.mat, there is no true number of clusters. Run the EM algorithm from K=1 to K=10. Get the BIC for each K and plot it (just like the previous sub-problems). Use this to find K for which the BIC is maximum. Run EM for this K and make a 1x3 figure (with subplot) to generate the following plots: (a) raw data with no labels, (b) results of clustering (choose the labels according to maximum probability) with contours showing the gaussian mixture and (c) log-likelihood as a function of iteration number. Did the K match the one you found with K-means? How do you think EM performed overall?