6140: Programming Assignment 3

In this assignment we will look at the EM approach for fitting Gaussian Mixtures to data.

Please view this assignment as a chance to code, analyze the methods and subsequently document the observations. This assignment is to be completed individually (i.e., no teams). You have to create all deliverables yourself from scratch. In particular, you are not allowed to look at or copy someone else's code/text and modify it. If you use publicly available code/text, you need to cite the source in your report! Cheating and other acts of academic dishonesty will be referred to OSCCR (office of student conduct and conflict resolution) and the College of Computer Science.

All programming assignments must be completed in **Python**. You will find a zip file with some useful code and data in the Resources section of the Piazza course page. You can use the code in zip file or external libraries to read files and plot results. However, you shouldn't use external libraries to implement your main algorithm. You need to make sure your code is **runnable** before you submit it. We will grade mainly based on your document submission, so make sure you have a clear explanation in the .pdf file including plots, running results, and explanations.

To submit your solutions,

1. Use your CCIS Github account and access the repository named ml-<your_name>
2. Create a new folder for this assignment: (e.g. PA3)
3. Push your solutions for this assignment to that folder.
4. Your submission should include one .py file and one .pdf file. Please name these files with your name (e.g. Kechen_Qin_PA3.pdf).

Please submit your solution by the due date shown online (usually two weeks from release). Late assignments will be penalized by 10% for each day late. For example, if you turned in a perfect programming assignment two days late, you would receive an 80% instead of 100%.

1 EM Algorithm

1. Implement EM for Gaussian Mixtures as described in Murphy (section 11.4.2[1]). Your program will need an input data set, initial mixture parameters and a convergence threshold (pick your favorite way of deciding convergence). The file `em_test` has a simple skeleton for reading data and plotting results. Note that the plots show mixture components as ellipses at two standard deviations.

2. To avoid numerical underflow problems, you will want to compute log likelihood. Note, however, that for a mixture distribution you will need to compute the log of a sum[2]. Look up the "logsumexp trick" to see how to deal with this. You can find `logsumexp` in Numpy. In any case, the idea is simple (once you see it).

3. Describe the behavior of your algorithm on all (non-mystery) training data sets provided as you vary (a) the number of components in the mixture, (b) the initial mixture parameters, (c) the convergence parameter and (d) the choice of small/large. Report the log-likelihoods and show selected plots of good and bad performance.

2 Variations

1. Modify your implementation of the EM algorithm so that it can build two types of models: ones with general covariance matrices and ones with diagonal matrices. Note that in each case different components have different means. Explain (in math, not code) the difference in the EM algorithm for these two variants.

2. Implement the K-means algorithm as a way to get an initial estimate of the mixture components.

3. Explore the performance of these algorithm variants using similar tests to what you did on the original algorithm.

3 Model Selection

In general, we can pick among candidate models on the basis of our estimate of how likely they make unseen data. We can use average (per data point, instead of sum) log likelihood to rank the models so that we can compare likelihood estimates based on different number of data points. Estimating average log-likelihood of a model on the training data is a very biased estimate of the

[1] You can find useful formulas in this section. You should also read other sections in Chapter 11 to learn the ideas of EM algorithm and GMMs

[2] $\ln P(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$

likelihood on unseen data. One common technique for getting a better estimate is cross-validation, in which multiple training and validation sets are constructed from the original training data.

1. Construct a candidate set of models for each of the data sets that differ on (a) the choices of covariance matrices (diagonal vs. general) and (b) the number of mixture components (1 – 5). Run the models for each **small** (non-mystery) training set based on average log likelihood from applying EM to the training set. You will need to decide exactly how to use EM (how to initialize, whether to run multiple times, etc); document your choices. Compare your results to the ranking of the models on the **large** "test" sets. Explain your findings.

2. Implement a cross-validation procedure that can be used to rank models. Your procedure should have a parameter K that determines how many folds of cross-validation are used; when K equals $N - 1$, where N is the number of training data points, this corresponds to "leave one out cross-validation".

3. Test your cross-validation procedure (for different values of K). (a) Compare the results on all the small and large data sets. (b) Compare the models you got by using cross-validation with the models you got by ranking on training set log likelihood. Test these models on test sets and explain your findings.

4 Predictions

1. We have given you some `mystery` training data sets. Make your best prediction for mixture parameters for these data sets. Explain how you made the predictions.