ISTA 410/510 Homework VI

For contribution to the final grade, due dates, current late policy, and instructions for handing the assignment in, see the assignment web page.

Please create a PDF document with your answers and/or the results of any programs that you write. You should also hand in your programs.

Questions marked by * are required for grad students only. They count as extra problems for undergraduates.

Questions marked by ** are challenge problems. They count as extra problems for both grads and undergraduates.

Any non-challenge problem can be replaced by a challenge problem; please make it clear that this is what you are doing (e.g., for a required problem you could answer "see optional problem #3"). The point here is to enable students to avoid problems that they feel are not instructive.

Extra problems (please indicated in your answer when you are doing an extra problem) are eligible for modest extra credit. The maximum score for an assignment will be capped at 120%. The maximum score for all assignments taken together is capped at 65/60.

For simplicity, problems are generally all worth the same, except ones marked by "+" that are expected to substantively more time consuming, and are worth double (or two no "+" problems). Additional "+"s scale linearly.

For full marks, undergraduates need 5 points, grad students need 8.

1. (++++, i.e., 5 points total) Implement EM for a GMM where the number of clusters is K, the dimensionality of the problem is D=2, and the covariance matrices are diagonal. However, the variances for each dimension, for each cluster, should be allowed to vary ($). You should monitor the log-likelihood while EM iterates. You should also structure your code to hold out every 10th point, and monitor the log-likelihood of the held out data as well. Your program should plot the data it reads in, and once it is done, it should plot the points again, now colored by cluster. For simplicity, your coloring can be based on the maximally likely cluster ($).

   You will probably find it best to generate various kinds of data to test your algorithm. However, some data sets that you should test on are provided on the web side.

   Test you code for D=2, and K=2,3,5,7, and 10, on these data sets using several random initializations.

   Report whether or not ($): a) the log likelihood of the training data always goes up; b) the log likelihood of the held data always goes up; c) whether the clustering is a function of the initialization.

   Note that the scale of the two log likelihoods is proportional to the number of points. Hence, the test log likelihood and the held out log likelihood are only comparable if you consider a factor of 10. More generally, we can think in terms of the per-point log-likelihood. In general (i.e., you may need to do a number of runs and/or analyze or visualize your data), report whether the per-point log-likelihood is better for the training data or the held out data ($). Explain whether this is what you expected or not ($).

   Hand in plots for two examples of clustering that you like, and two examples of clustering that you don't like ($).

2. (*,++, i.e., three points total) Modify your program so that the full covariance matrix can be learned ($). Provide plots for a clustering that you like, and a clustering that you do not like ($). Is the general behavior of the held out log likelihood consistent with what you found in (1)? Now comment on the absolute values of the fits (likelihoods) on the training and held out data ($).

3. (**,++) Consider using a mixture model for classification, or (case (iii)) below, perhaps you want to evaluate a clustering on the basis of its capacity to predict labels. Call the data *X*, each data point has a label, *l*, from a set *L*, with prior *p(l)*. Index clusters by *c*, and points by *n*. In a training set there are N_l points with label *l*. There are *N* training points total. After training we may want to classify a new point, *x*, into one of labeled classes. Three approaches come to mind (feel free to suggest others):

   i. Train a GMM for each label set.

   ii. Develop a multimodal mixture model, where the mixture consists of clusters whose statistical model has both a Gaussian factor and a discrete factor which are conditionally independent, given the cluster.

   iii. Assume that you trained the GMM without knowledge of the labels, and then you were provided with the labels. Develop an expression for *p(l|x)* that uses the labels for the GGM.

   For (i) provide the classification formula ($). For (ii) develop the model and provide the classification formula ($). For (iii) develop the classification formula.

4. (**, 3 to 5 points). If you have already implemented EM on toy problems, and are familiar with the vision lab C libraries, you might prefer to use the existing implementations for EM with GMM for (1) and (2) above. Follow this by improving the implementations which are a bit of a mess. In addition to improving modularity, making the feature sets more consistent over the models, rationalizing parameters, improving efficiency, and so on, I suggest making the code multi-threaded, where a

sensible number of threads can be set using an option, with the default ideally set based on the platform. (A pair of students made a start on this last year, and if you are interested in continuing this work, I will put you in touch with them). You should do timing tests. Another direction to investigate is the problem of doing EM if the data is too large to fit into memory.

If you chose this assignment, let Kobus know. If more than one person is keen, we will coordinate the effort.

5. (**, 3 to 5 points). If you have already implemented EM, and do not like question (3), propose your own EM related mini-project.