CSC 535 – Probabilistic Graphical Models

Assignment Eight

Due: 11:59pm (*) Monday, Nov 26.

(*) There is grace until 8am the next morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted.

Weight is about 7 points

This assignment can be done in groups of 2-3. You should create one PDF for the entire group, but everyone should hand in a copy of the PDF to help me keep track. Make it clear on the first page of your PDF who your group is. Group reports are expected to be higher quality than individual ones. This assignment is unrelated to any previous assignment, so if you want to change groups, or create new ones, that is fine.

The purpose of this assignment is to learn about approximate inference for missing value problems by implementing the EM algorithm. There is one optional problem at the end.

Deliverables

Deliverables are specified below in more detail. For the high level perspective, you are to provide a program to output a few numbers and to create figures. You also need to create a PDF document that tells the story of the assignment including output, plots, and images that are displayed when the program runs. Even if the question does not explicitly remind you to put the resulting image into the PDF, if it is flagged with **($)**, you should do so. The instructor should not need to run the program to verify that you attempted the question. See

http://kobus.ca/teaching/grad-assignment-instructions.pdf

for more details about preparing write-ups. While it takes work, it is well worth getting better (and more efficient) at this. A substantive part of each assignment grade is reserved for exposition.

GMM and EM

1. (++++, i.e., 5 points total) The next paragraph describes a program you should develop. The deliverable for this part is to hand in your code **($)**. Below, are some data sets to play with followed by a specific set of things to put into your writeup

   Implement EM for a GMM where the number of clusters is K, the dimensionality of the problem is D=2 (but best to make this easy to change), and the covariance matrices are diagonal. However, the variances for each dimension, for each cluster, should be allowed to vary. You should structure your code to hold out every 10th point (so 9 points for training, and then one for test). You should monitor the log likelihood for the training and held out data. To make these comparable, you should divide each of these two log likelihoods by the number of points that contribute to them to get a "per-data-point" estimate—otherwise the training and held out log likelihoods are not comparable, roughly be a factor of 9. Your program should provide plots for the log likelihoods of both the training and held out data as function of iteration. 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 should be based on the maximally likely cluster.

   Note that there are many implementations of EM for GMM available on the web, and there is even one in Matlab. ***However, you are to implement it yourself****. This is the sort of thing that it is useful to implement once.*

   It is possible that as you try larger K clusters will seem to disappear in the sense that they do not own any points based on maximal probability. This can be mitigated by a method covered in the "EM in practice" part of the lecture notes. You are encouraged to experiment with such things, but for the purpose of this assignment, you need not worry about it.

   For consistency, we will characterize convergence based on the per-point training log-likelihood $\mathbb{L}^{(s)}$, where the superscript s is the time step, and define $\delta = \left| \frac{\mathbb{L}^{(s)} - \mathbb{L}^{(s-1)}}{\mathbb{L}^{(s-1)}} \right|$. We will say that we have converged if we have used up 200 time steps, or we observe $\delta \leq 10^{-8}$, whichever comes first. You will likely find it a good idea to monitor the δ after each iteration (except the first) as well as the per-point training log-likelihood and held-out data log-likelihood.

   Tip. The training log-likelihood must go up. It is tempting to write off small deviations from this requirement as due to machine precession, but it is ***far more likely*** *it is a bug. If you think it is due to precision issues you can use some of the tricks covered in class to mitigate them. Having done this, the training data log likelihood* ***must*** *go up.*

   You will probably find it useful to be able generate various kinds of data to test your algorithm. However, for your report, you should provide results based on the following data sets as detailed below. These data sets were generated using the distributions and number of clusters suggested by the file name. For the first one, the data co-vary with dimension (i.e., X and Y). For the others ("independent"), the data from each dimension are independent given the cluster.

   http://kobus.ca/teaching/cs535/data/gaussian-3-clusters-A.txt
   http://kobus.ca/teaching/cs535/data/independent-gauss-3-clusters-A.txt
   http://kobus.ca/teaching/cs535/data/independent-gauss-3-clusters-B.txt
   http://kobus.ca/teaching/cs535/data/independent-gauss-5-clusters-A.txt
   http://kobus.ca/teaching/cs535/data/independent-uniform-5-clusters-A.txt
   http://kobus.ca/teaching/cs535/data/independent-uniform-5-clusters-B.txt

Check the effect of the random initialization by providing panels showing the clustering for K=3 for the first and third data sets, with four different initializations for each **($)**.You do not need to provide the log-likelihoods, only two panels with four plots each. Report whether the clustering is a function of the initialization, and provide additional analysis as appropriate **($).**

Next run your program using K=3 and K=5 on all 6 data sets for enough random initializations so that you have some confidence that what you observe is not special due to the initialization. Report whether or not: a) the log likelihood of the training data always goes up (**$**); b) the log likelihood of the held data always goes up (**$**); and c) whether the per-point log-likelihood is better for the training data or the held out data, and explain whether or not this is what you expected.

Provide plots showing the clustering and the log likelihoods for K=3 and K=5 for the 6 data sets **($)**. Provide some comments about your results. If you do not think they are that interesting, provide some alternative results (different K's, different data sets generated by yourself, etc.) and comment on those.

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 in comparison to the diagonal covariance case **($)**.

[Optional problem.]

3. (**). Consider using mixture models for classification, or for case (c)) 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):
   (a) Train a GMM for each label set. The GMMs should have the same number of clusters, which we will denote by C.
   (b) Develop a multimodal mixture model, where the mixture consists of clusters whose statistical model has both a Gaussian factor for the observations, and a discrete factor for the label, which are conditionally independent, given the cluster.

   Example*. Consider image regions with features* **x** *and a label L. For example, we might have a cluster where the Gaussian for features aligns with "orange and black stripy" and the label TIGER. But since we plan to learn such "concept" clusters, we need a distribution over labels, and further, some regions are naturally associated with multiple words, even without uncertainty. Hence, each cluster in the mixture has a Gaussian for the features and a discrete probability distribution over labels.*

   (c) 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 (a) provide the classification formula (**$**). For (b) develop the model and provide the classification formula (**$**). For (c) develop the classification formula (**$**).

What to Hand In

Hand in the PDF file hw8.pdf with the story of your efforts into D2L. If you wrote code for this assignment (you should!), hand in a program hw8.<suffix> (e.g., hw8.m if you are working in Matlab). If you are working in groups, the code and the PDF can be the same for each person in the group, but each member should hand in a copy to help me keep track of things. PDFs from groups are expected to be of higher quality than individual ones.