CSC412 – Assignment #3

Due: March7, before 4pm in the AI office, LP283
Worth: 10%
Late assignments not accepted.

1 Learning Mixtures of Gaussians

In this question you will implement the EM algorithm for training a mixture of full covariance Gaussians:

$$p(\mathbf{x}) = \sum_k p(z=k)p(\mathbf{x}|z=k) = \sum_k \alpha_k |2\pi\Sigma_k|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{x}-\mu_k)\right\}$$

1.1 What to do

- Using the data provided in `a3geyser.mat`, train mixtures with 1,2,3 and 6 components, starting at 1000 random initializations and training until convergence.

  In case you are interested, the data represents intervals between erruptions and durations of erruptions of the Old Faithful Geyser at Yellowstone National Park, USA.

1.2 What to hand in

- The equation you are using to update the covariance matrices in the M-step (see below).
- A plot of the best fit (highest likelihood) that you found for each number of components, showing the training data as points and the one-standard deviation ellipses of the Gaussians. (4 plots total, on a single page.)
- A histogram, for each number of components, of the final log likelihoods of the training data and of the number of iterations of EM it took for training to converge. Don't use too few or too many bins in your histograms. (8 histograms total, arranged in two columns of 4, on one page.)

1.3 Some hints

- You might find the the function `plotGauss.m` helpful in making the one-sigma Gaussian ellipse plots.
- You can assess convergence of EM by monitoring the log likelihood and stopping when the fractional change `(newlik-oldlik)/abs(newlik)` is less than one part in a million. Your log likelihood should never decrease!
- Try to get the EM algorithm working *without* the covariance updates at first (ie just update the means). Then when that is working, and your likelihood is always going up, try adding in the covariance updates.
- Initialize the model parameters by setting the means randomly in some sensible range, for example the covariance of the entire data set. Set the covariances to some small but not tiny values. Make sure there is enough randomness in your initialization that you fall into different local maxima.
- You might find it interesting to plot the parameters after every iteration of EM, so you can see how things are going. If your code issues a `drawnow` command in MATLAB after plotting the data and all the Gaussian ellipses this will update the plots on the screen. (If these ellipses don't fit the data at all, you may have an error somewhere in your code or in your covariance update equation!)

- Every once in a while, if you are unlucky, a Gaussian cluster will get trapped on a single datapoint and its variance will get very small. This can cause numerical problems, so watch out for it. If this happens, you can just restart at a different random initialization.

1.4 Reminders

Amongst other things, your code will need to:

- Compute the log probability of every datapoint under each mixture component:

$$\ell_{nk} = \log p(\mathbf{x}^n, z = k|\mu_k, \Sigma_k)$$

- Compute the log probability of every datapoint:

$$\ell_n = \log p(\mathbf{x}^n) = \log \sum_k p(\mathbf{x}^n, z = k|\mu_k, \Sigma_k)$$

  NB: Make sure to do this in a numerically stable way. In particular, *do not* use $\ell_n = \log \sum_k \exp(\ell_{nk})$.
  Hint: You may find the function `logsum.m` helpful.

- Compute the E-step "responsibilities":

$$r_{nk} = p(z = k|\mathbf{x}^n) = \frac{p(\mathbf{x}^n, z = k)}{p(\mathbf{x}^n)} = \exp\left(\ell_{nk} - \ell_n\right)$$

  (You probably don't want to use $\exp(\ell_{nk})/\exp(\ell_n)$. Think about why.)

- Update the mixing proportions in the M-step:

$$\alpha_k^{new} = \frac{\sum_n r_{nk}}{N}$$

- Update the means in the M-step:

$$\mu_k^{new} = \frac{\sum_n r_{nk}\mathbf{x}^n}{\sum_n r_{nk}}$$

- Update the covariance matrices Σ_k in the M-step.
  (I've left this equation for you to write down from the class notes and the book.)
  Use full (not diagonal or sphereical) covariances.

$$\Sigma_k^{new} = ???$$

- At each iteration, you should also compute the total log probability of the training set:

$$\ell = \sum_n \log p(\mathbf{x}^n) = \sum_n \ell_n$$

  and verify that this never goes down.

2 Fully Observed Trees

In this assignment you will train a fully observed tree model on word-document vectors taken from USENET articles.

2.1 What to do

- Using the data provided in `a3newsgroups.mat`, train a fully observed tree model with the optimal structure. The data has 100 (binary) features and 16242 training cases.
- The matrix `documents` contains one column per USENET article.
  Each of the 100 rows is a binary variable indicating whether a certain keyword appears in the posting or not. The cell array `wordlist` has the 100 words corresponding to these rows.

2.2 What to hand in

- Any *compact, clear and unambiguous* representation of the optimal (undirected) tree structure found by your algorithm. For example, you might print out all pairs of words `(word_i--word_j)` corresponding to edges chosen by the tree learning algorithm. Or you might want to draw a picture of the graph (see below).
  DO NOT hand in lists of numbers corresponding to pairs of features chosen – convert everything back to words using the `wordlist` cell array.
- A histogram (with at least 100 bins) of the log likelihoods of the training cases under your optimal model.
  Compute and print out the min,max,mean and median log likelihood.
  Which training case (give its number) has the worst (lowest) log likelihood?
  Print out the list of keywords appearing in this posting.
  What percentage of postings mention both "god" and "ftp"?
- After finding the optimal tree structure for the 100 variables, you may choose any node you wish as the root and form a directed graph. Different choices of root node will result in different output when you visualize a directed tree, but of couse the undirected model structure always remains the same. Chose a root word. Hand in a *compact, clear and unambiguous* representation of a directed tree structure created by directing all arcs in the optimal structure away from a your chosen root. For example, you might print out a breadth-first list of directed links `(word_i->word_j)` in the tree. Alternately, you might want to draw a picture (see below).

2.3 Some hints

- Be careful when computing the mutual information weights for pairs of variables that have zero counts for some of their joint settings. In this case the mutual information should be zero, but a careless calculation will lead to division by zero or log of zero errors in the code.
- To save you some trouble, I've written the function `mwst.m` for you, which finds minimum or maximum weight spanning trees given a weight matrix.
- Here is a trick for computing the 2 by 2 joint count table of two binary column vectors `v1` and `v2`:
  ```
  counts12 = reshape(hist(v1+2*v2,0:3),2,2);
  ```
  but make sure you understand it before using it!
- The data is almost a megabyte in MATLAB , stored as a sparse matrix, so be careful about making copies of it in memory. You should be able to do everything you need without making another (sparse or nonsparse) copy of the whole dataset.
- In case you are interested, the array `newsgroup` tells you the toplevel newsgroup category of each posting, `1=comp.*, 2=rec.*, 3=sci.*, 4=talk.*`, although we won't use it in this assignment.

2.4 Reminders

Amongst other things, your code will need to:

- Compute the marginal count for every feature being on or off summed across all documents.
- Compute the joint counts for every pair of features being on together, off together, on-off or off-on, summed across all documents.
- Convert these counts into mutual information weights.
  Be careful to do this properly for joint counts which are zero!
- Find the best spanning tree over the features given these weights. Depending on how you compute the weights, you will either need a minimum or maximum weight spanning tree.
  (This will in turn affect how you call `mwst.m` if you choose to use it.)
- Select a root for the tree and direct all edges away from it to get a directed graph.

2.5 Automatic Graph Layout

- If you want to be really fancy, you can try using the function `dottree.m` and the automatic graph layout programs `dot` and `neato` to draw pictures of your final structures. It is very cool looking, but you will not get any more or any fewer marks for using this method or a simpler method to display your results.
  Caution: do not waste too much time trying to get this to work. Get the rest of the assignment finished first.
- Call `dottree(tree,'fname',wordlist)` and it will write out two files, `fname.u` and `fname.d`.
  Now call the `neato` and `dot` programs to convert these files into pictures.
- Information and downloads for `dot` and `neato` are available at `http://www.graphviz.org/`.