EECE 5698

Homework 5: Due Wednesday, March 19, 2014

Computer Assignment: Clustering

Write a Matlab implementation of k-means and EM of a finite multivariate Gaussian mixture.

This homework contains a bit of Matlab programming although it's pretty straightforward. Be sure to work on this homework well before the deadline.

General Advice

In coding your algorithms, you should try to make your code as modular as possible, by defining relatively short general functions that can be used in different places in the code. For example, you can define specific functions for initialization, the E-step, M-step, function that evaluates the Gaussian density for each row of an *nxd* matrix A given the parameters, etc.

Please use the following general array structure for Gaussian parameters and data matrix formats, unless you have a preference to use your own data structures:

```
%          data:  n x d data matrix, n observations (rows), d variables (columns)
%        labels:  n x 1 vector of class labels corresponding to "data",
%                 where label values are assumed to go from 1 to m
%       cparams:  an m x 1 array of structures, where cparams(k) is the kth such
%                 such structure containing:
%      cparams(k).prior = estimated prior probability for class i
%      cparams(k).mu = d x 1 vector containing the maximum likelihood
%                 estimate of the mean for class i
%      cparams(k).covar = d x d matrix containing the maximum likelihood
%                 estimate of the covariance matrix for class i
%
```

Algorithm 1: K-means Clustering

Implement the basic version of the algorithm as described in the text in Section 10.4.3 on page 527. Feel free to experiment with different variants of the algorithm, e.g., using different methods other than purely random restart to initialize K-means. You should have an option to run the algorithm *r* times from *r* different randomly chosen initializations, where the final clustering is selected (from the *r* possible clusterings) that gives the lowest sum of squares error. Note that *r* and K are inputs to the algorithm supplied by the data analyst.

Algorithm 2: Gaussian Mixture Clustering

Use the EM update equations provided in the class notes for a finite multivariate Gaussian mixture.

Use the following general outline for your Gaussian mixture Matlab code:

1. Initialize the parameters $\theta^{(0)}$, t=0.

2. While (convergence condition is not satisfied):

    a. E-Step: compute membership probabilities using $\theta^{(t)}$.

    b. M-Step: compute new parameters $\theta^{(t+1)}$ using $E[z_{ij}]$ from (a).

    c. Convergence condition: Calculate the log-likelihood using $\theta^{(t+1)}$ and check if the relative increase since the last iteration is below some threshold. If so, halt and return the current parameters. If not, continue to iterate.

    d. Increment $t = t+1$.

Note: This is a non-trivial algorithm to get working properly. Please try and debug it carefully, e.g., check that the log-likelihood is non-decreasing at each step (i.e., have your code print it out as it goes through each iteration), write out the final parameters for the components. To see how your algorithm is progressing, you may find it interesting to plot the parameter estimates for each EM iteration. For the synthetic data, you can compare and check whether the estimated parameters are roughly equal to the true parameters.

Some implementation details: There are a number of options in implementing this algorithm, which I will leave up to you to determine, e.g.,

1. Initialization: Your algorithm can end up in a local maximum. I recommend that you apply *r* (e.g., 10) random restarts to avoid getting trapped at this local optimum. That is, select *r* completely randomly chosen starting conditions: e.g., select the initial K mean vectors by randomly selecting K initial data points, and select the initial K covariances to be the identity matrix (the selection of the initial covariances is not as critical as the initial means). Then, run your algorithm until convergence starting from each of these *r* initial conditions. Pick the one that results in the highest log-likelihood.

2. Convergence: you will have to decide when it's no longer worth iterating, e.g., from a practical viewpoint, it may not be worth continuing if the increase in log-likelihood between the last two iterations is less than (say) 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.

3. Singular solutions: the likelihood can go to infinity if the determinant of the covariance matrix goes to zero (e.g., 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 $\in$ (e.g., 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 *r* 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.

Algorithm 3: Choosing K for Gaussian Mixture Clustering

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

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

is maximized, where $L(D|\theta)$ is the maximizing value of the log-likelihood as found by EM using data D with K components, p_k is the total number of parameters in your mixture model K, and n is the number of data points in 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.

To write a Matlab program to do this is very simple given the algorithm 2 that you have already written. Simply have a loop that runs the EM algorithm with values of K going from 1 to some K_{max} where K_{max} is selected by the user (e.g., 10 for all of the data sets we are using here). 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. 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 K_{max} smaller, and/or run a larger number of random restarts).

What to Hand In:

Please try to put multiple plots all on the same page (where different pages may have different data sets and different algorithms) using "subplots" in Matlab, to avoid printing lots of pages.

1. For each data set, using the true K (the K equal to the number of labeled classes) value for each one, show the following:

    a. The K-means solution (scatter plot in any two dimensions illustrating the location of the solution (i.e., the cluster means), and plotting the data from different clusters with different symbols.

    b. A plot of the sum-squared-error (divided by *n*) as a function of iteration number in the K-means algorithm.

    c. The initial parameter values and the final parameter values (2 plots, each showing means and covariances for each cluster) for the EM/Gaussian mixtures code for the highest-likelihood solution. Use plotgauss.m from homework 1 to do the plotting. Plotgauss.m only plots one Gaussian at a time. One way to plot multiple Gaussians is to call plotgauss.m one at a time and use the "hold" command to overlay the plotting of the following figures on the previous ones. Also, show a scatter plot of the data and display the data coming from different classes (using the true labeled classes) with different colors and/or symbols. This will give you an idea on how well the discovered clusters correspond to the "true" classes.

    d. A plot of the log-likelihood as a function of iteration number during EM.

    e. Add some brief comments (1 paragraph) on the difference between K-means and EM for each data set.

2. For each data set, generate a table of log-likelihood and BIC scores for K going from K=1 to some maximum value. Comment briefly on the results.

3. Attach as an Appendix, printouts of your Matlab code.

Datasets:

There are four data sets in this homework. The data format is one data point per row, and each column corresponds to a feature. The last column corresponds to the class labels. Do not use this labeled information (delete the last column) when running your clustering algorithm. You may use this information to check/evaluate your results. You should definitely read the data into Matlab and plot some scatter plots, before you do any clustering, so that you'll have an idea on how the data looks like. Use data set 1 to debug your code.

1. Data set 1:

   This is a simulated two-D data generated from two Gaussian components with some overlap. The true means are at [0,0] and [0,3], with identity covariance matrices.

2. Data set 2:

   This is a simulated two-D data generated from three Gaussian components. The true means are at [0,0], [0,0], and [2,0], with covariance matrices:

$$\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix} \quad \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 1.0 \end{pmatrix} \begin{pmatrix} 0.2 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

3. Data set 3:

   The iris data is the "best known database to be found in the pattern recognition literature." This is a very simple data set with 150 samples and four features. This data set contains three classes: iris setosa, iris versicolour and iris virginica. The features are measures in centimeters of sepal length, sepal width, petal length, and petal width. This data set was obtained from the UCI repository. You can obtain various benchmark data from this site: http://www.ics.uci.edu/~mlearn.

4. Data set 4:

   The geyser data represents intervals between eruptions and durations of eruptions of the Old Faithful Geyser at Yellowstone National Park. For this data set, use 1000 random restarts, and vary k from 1 to 6. This data set was obtained from http://www.cs.toronto.edu/~roweis/csc2515/assignments.html.