

ICS 274 Homework 5

Probabilistic Learning: Theory and Algorithms, ICS 274, Winter 2009

Due Date: Thursday, March 6th, by end of class

Assigned Reading

- Note Set 4 on the EM algorithm for Gaussian mixtures. Optional additional reading is Chapter 9 of Bishop.

MATLAB: K-Means and Gaussian Mixture Learning

This homework consists of quite a bit of MATLAB programming. If you take it step by step it is straightforward, but be sure to start working on this well in advance of the deadline. If you prefer to do this in R rather than MATLAB, that is fine.

Data Sets

There are four data sets for experimentation: all are low-dimensional (2 or 3 dimensions) so that you can visualize the results. The data format is 1 data point per row, each column corresponds to a feature. The data sets are available from the class Web page. Data set 3 also has class labels, called `labelset3`, which are not to be used in clustering but just for evaluating your results afterwards. Some of the data sets were written out by MATLAB in floating point format (if you look at them with a text editor) but once you read them into MATLAB they will be show up in their “natural” form. You should read the data into MATLAB and plot some scatter plots, before you do any clustering, just so that you have an idea of where the data are in various 2-dimensional plots of the data. Data set 1 is probably the best one to use initially when debugging your algorithms (it has the most data and has the simplest structure).

1. Data set 1: this is 2d data simulated from 2 Gaussians, with some degree of overlap. The first 800 points belong to 1 Gaussian, the 2nd to the other Gaussian.
2. Data set 2: this is 2d data simulated from 3 Gaussians, with considerable overlap. There are 500 points from each Gaussian, ordered together in the file.
3. Data set 3: this a real 2d data set that I have worked on with Professor Christine McLaren, Department of Epidemiology, UCI. Each data point represents an individual person: one group is normal healthy people, the other has iron deficient anemia (so the true value of K is 2). The two measurements on each individual are their mean red blood-cell volume and their mean hemoglobin concentration. Both measurements tend to be lower for anemic individuals. The class labels are provided in the file `labelset3`. You are to run the clustering

algorithms in an unsupervised manner, i.e., without class labels. You can then use the labels after clustering if you wish to check to see how the clustering matches with “truth.”

4. Data set 4: this is another real data set taken from Banfield and Raftery, ‘Model-based Gaussian and non-Gaussian clustering,’ *Biometrics*, 49, 803–821, 1993. The data are plasma measurements on medical patients and consist of 3 features called glucose-area, insulin-area, and steady-state-plasma glucose response. 145 subjects were classified into 3 groups by medical experts: normal, chemically diabetic, and overtly diabetic. This “true” classification is not provided in the publicly-available data set so this data set truly involves unsupervised learning (although the true labels are available from the authors of the original medical study).

Note that in some of these data sets the rows may be ordered by class, i.e., all data from one class first, then another, etc., since these data sets came from data where class labels exist. I recommend you randomize the order of the rows for these data sets before you run your clustering algorithms, just to avoid any possible biasing of your algorithm results based on row order.

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 a simple function called `euclid.m` that calculates the Euclidean distance between a vector x of dimension $1 \times d$ and each row of a matrix A of dimension $n \times d$. This is a basic function in the K -means algorithm below. Similarly you can define specific functions for initialization, the E-step, the M-step, etc., for Gaussian mixtures below, where each of these functions may call more basic functions, such as a function that evaluates the Gaussian density for each row of a matrix A given the Gaussian parameters. By making your code modular in this fashion you can test individual functions independently from the rest of the code, your code will be much easier to read and debug, and you will be able to re-use the same functions in different places.

You should also write your equations in MATLAB using vector and matrix notation where possible, and avoid for-loops when you can. This will make your code simpler and faster.

You may also want to use the same general array structure for Gaussian parameters and data matrix formats that we used in Homework 3, unless you have a preference to use your own data structures. Keep in mind that for Homework 3 we knew the class labels: here we are estimating the class memberships (or cluster memberships) in an unsupervised manner from the data.

Algorithm 1: K -means Clustering

Implement the basic version of the algorithm as described in class. Feel free to experiment with different variants of the algorithm, e.g., using different methods other than purely random choice to initialize the K -means. You should probably also 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

As discussed in class, in this approach we assume a Gaussian mixture model of K components for the data and we find the parameters of the model using maximum likelihood. The equations for updating are provided in Note Set 4 on the class Web page. These equations are the heart of Gaussian mixture modeling.

The “M-step” equations provide new parameter estimates at each iteration (the left hand sides of the equations). It is important to compute the equations in the order given. The M-step equations are computed K times at each iteration, once for each mixture component/cluster.

The E-step provides the “membership” probabilities for each data point for each class: so we should get an $n \times K$ matrix of probabilities, where each row sums to 1. On the right-hand side of the E-step equations we use the most recently available parameter estimates (e.g., those from the most recent M-step).

I recommend that you use the following general outline for your Gaussian mixture MATLAB code:

- Initialize the parameters to θ^0 , $i = 0$.
- While(convergence condition not satisfied):
 - E-step: compute membership probabilities using the current $\theta^{current}$ values.
 - M-step: compute new parameters θ^{new} using the membership probabilities from the E-step
 - Convergence condition: Calculate the log-likelihood using θ^{new} and check if the relative increase since the last iteration is below some threshold (see below for more details). If so, halt and return the current parameters. If not, continue to iterate
- end

I also recommend that your algorithm runs this process multiple times from a number r (e.g., 10) of different starting points (different θ^0 values) and picks the one that results in the highest log-likelihood (since EM in general only finds local maxima). One heuristic is to select r different randomly-chosen starting conditions. For example, for each start, select the initial K Gaussian means by randomly selecting K initial data points, and select the initial K covariances as all being some multiple of the overall data covariance—the selection of initial covariances is not as critical as the initial means). Another option for initialization is to randomly assign class labels to the training data points and then calculate θ^0 based on this initial random assignment (or begin the iterations by executing a single M-step, which is also fine).

This is a non-trivial algorithm to get working properly: please try and debug it carefully. Check that the likelihood is non-decreasing at each step (i.e., have your code print it out as it goes through each iteration—if the log-likelihood ever decreases you have a bug in your code). You can also write out the final parameters for the components and for the first 2 data sets (where we know the true classification and the true model and can, thus, compute the true values of the parameters) and check that the estimated parameters are roughly equal to the true parameters. Note that you should also be able to partially reuse some of the code you have already written in earlier homeworks for Gaussian classifiers.

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

1. Convergence: you will have to decide when its 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 2 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 could also impose a maximum number of iterations to halt the algorithm (e.g., 100 or 500) if it gets that far and still has not converged. This is particularly handy when debugging!
2. Singular solutions: the likelihood can go to ∞ 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. 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 ϵ (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 to check each diagonal entry: if any are less than the threshold, then replace them with the threshold. 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 approach (but useful) for doing this is the BIC criterion, i.e, choose K such that

$$l(D|\hat{\theta}) - \frac{p_K}{2} \log n$$

is maximized, where $l(D|\hat{\theta})$ is the maximizing value of the log-likelihood as found by EM using data D with K components (this value of the log-likelihood can be returned by your Gaussian mixture code), p_K is the total number of parameters in mixture model K (you will need to compute this for each 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 $\frac{p_K}{2} \log n$ can be shown to be a good approximation to a true Bayesian penalty term. For mixture models the theory does not hold in general, but nonetheless it can be a useful approximate technique for model selection with mixtures.

To write a MATLAB program to do this is very simple given your EM 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 (e.g., the medical 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).

An additional option if you are interested in this problem of model selection (purely for your own interest, will not be graded) is to download the paper *Model selection for probabilistic clustering using cross-validated likelihood* accessible at www.datalab.uci.edu/papers/tr9809_rev.pdf. It describes a data-driven method for finding the best K in the mixture model clustering framework that provides a useful alternative (in some cases more accurate) to BIC.

What to Submit

Please submit both:

1. Your MATLAB or R code to the Homework 5 dropbox. Please upload all of your functions in a single compressed file (please use .gz or zip format)
2. A written report of your results (details below).

Please try to put multiple plots on the same page using “subplots” in MATLAB (where different pages could correspond to different data sets and different algorithms), to avoid printing lots of pages.

1. For each data set, using the true K value for each one, show the following
 - (a) The K -means solution (scatter plot in two dimensions (any two dimensions for Data Set 4) 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 (Algorithm 2 above) for the highest-likelihood solution. You can use `plotgauss.m` from earlier homeworks to do the plotting.
 - (d) A sample 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.