MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.036—Introduction to Machine Learning
Spring 2015

Project 3: Rating movies with mixtures Issued: Mon. 4/12 Due: Fri. 5/1 at 9am

Project Submission: Please submit two files—a *single* PDF file containing all your answers, code, and graphs, and a *second* .zip file containing all the code you wrote for this project, to the Stellar web site by 9am, May 1st.

Introduction

Your task is to build a mixture model for collaborative filtering. You are given a data matrix containing movie ratings made by users where the matrix is extracted from a much larger Netflix database. Any particular user has rated only a small fraction of the movies so the data matrix is only partially filled. The goal is to predict all the remaining entries of the matrix.

You will use mixtures of Gaussians to solve this problem. The model assumes that each user's rating profile is a sample from a mixture model. In other words, we have K possible types of users and, in the context of each user, we would sample a user type and then the rating profile from the Gaussian distribution associated with the type. We will use the Expectation Maximization (EM) algorithm to estimate such a mixture from partially observed rating matrix. The EM algorithm proceeds by iteratively assigning (softly) users to types (E-step) and subsequently re-estimating the Gaussians associated with each type (M-step). Once we have the mixture, we can use it to predict values for all the missing entries in the data matrix.

1. **Part 1** Warm up.

   For this part of the project you will compare clustering obtained via K-means to the (soft) clustering induced by EM. Our K-means algorithm differs a bit from the one you have learned previously. The means are estimated exactly as before but the algorithm returns a bit more. We use the resulting clusters of points to estimate a Gaussian model for each cluster. Thus our K-means algorithm actually returns a mixture model where the means of the component Gaussians are the K centroids computed by the K-means algorithm. The point here is that we can now directly plot and compare solutions returned by the two algorithms as if they were both estimating mixtures.

   (a) Read a 2D toy dataset using `X = readData('toy_data.txt')`. Your task is to run the K-means algorithm on this data using code `kMeans` we have provided. Initialize K-means by running `(Mu,P,Var) = init(X,K)` where K is the number of clusters. Note that `init(X,K)` returns a K-component mixture model with means, mixing proportions, and variances. The K-means algorithm will only care about the means, however, and returns a mixture that is retrofitted based on the K-means solution. Try $K = [1, 2, 3, 4]$, plotting each solution using our `plot2D` function. Since the initialization is random, make sure you try each K a few times to select the one that minimizes the total distortion cost (as explained in lecture). Submit the associated plots (best solution for each K).

   (b) Recall the Gaussian mixture model presented in class:

$$P(x|\theta) = \sum_{j=1}^{K} p_j N(x; \mu^{(j)}, \sigma_j^2 I),$$

where θ denotes all the parameters in the mixture (means $\mu^{(j)}$, mixing proportions p_j, and variances σ_j^2). The goal of the EM algorithm is to estimate these unknown parameters by maximizing the log-likelihood of the observed data $x^{(1)}, \ldots, x^{(n)}$. Starting with some initial guess of the unknown parameters, the algorithm iterates between E- and M-steps. The E-Step softly assigns each data point $x^{(i)}$ to mixture components. The M-step takes these soft-assignments as given and finds a new setting of the parameters by maximizing the log-likelihood of the weighted dataset (expected complete log-likelihood)

Implement the EM algorithm for a Gaussian mixture model (as recalled above). To this end, expand the skeleton functions `Estep` and `Mstep` provided in `project3_student.py`. In our notation,

- X: an $n \times d$ Numpy array of n data points, each with d features
- K: number of mixture components
- M: $K \times d$ Numpy array where the j^{th} row is the mean vector $\mu^{(j)}$
- P: $K \times 1$ Numpy array of mixing proportions p_j, $j = 1, \ldots, K$
- Var: $K \times 1$ Numpy array of variances σ_j^2, $j = 1, \ldots, K$

Your code will output updated versions of M, P, and Var, as well as an $n \times K$ Numpy array `post`, where post$[i, j]$ is the posterior probability $p(j|x^{(i)})$, and `LL` which is a list of the log-likelihood scores evaluated at the beginning of each EM iteration (returned by `Estep`)

(c) Now that you have a working implementation of EM, let's check that it is indeed correct. First, EM should monotonically increase the log-likelihood of the data. Initialize and run the EM algorithm on the toy dataset as you did earlier with K-means. You should check that the `LL` values that the algorithm returns after each run are indeed always monotonically increasing (non-decreasing). As another validation, let's compare your algorithm to our implementation. Set $K = 3$ and let `(Mu,P,Var) = init(X,K,fixedmeans=True)`. If you run the EM algorithm with this initialization, you should get -1331.67489 as the initial log-likelihood and -1138.89143 as the final one when the algorithm converges.

(d) Generate analogous plots to K-means using your EM implementation. Note that the EM algorithm can also get stuck in a locally optimal solution. For each value of K, you should run the EM algorithm multiple times with different initializations returned by `init` and select the solution that achieves the highest log-likelihood. Compare the K-means and mixture solutions for $K = [1, 2, 3, 4]$. Briefly explain when, how, and why they differ.

(e) So far we have simply set the number of mixture components K but this is also a parameter that we must estimate from data. How does the log-likelihood of the data vary as a function of K assuming we avoid locally optimal solutions? To compensate, we need a selection criterion that penalizes the number of parameters used in the model. Fill in the missing Bayesian Information Criterion (BIC) calculation in `BICmix(X,Kset)`. Find the best K using `Kset=[1,2,3,4]` and the toy dataset. Report the best K and the corresponding BIC score. Does the criterion select the correct number of clusters for the toy data?

2. **Part 2** Mixture models for matrix completion

We can now try to extend our Gaussian mixture model to predict actual movie ratings. Let X again denote the $n \times d$ data matrix. The rows of this matrix correspond to users and columns specify movies so that `X[u,i]` gives the rating value of user u for movie i (if available). Both n and d are typically quite large. The ratings range from one to five stars and are mapped to integers $\{1, 2, 3, 4, 5\}$. We will set `X[u,i]=0` whenever the entry is missing.

In a realistic setting, most of the entries of X are missing. For this reason, we define C_u as the set of movies (column indexes) that user u has rated and H_u as its complement (the set of remaining unwatched/unrated movies we wish to predict ratings for). From the point of view of our mixture model, each user u is an example $x^{(u)} = \texttt{X[u,:]}$. But since most of the coordinates of $x^{(u)}$ are missing, we need to focus the model during training on just the observed portion. To this end, we use $x^{(u)}_{C_u} = \{x^{(u)}_i : i \in C_u\}$ as the vector of only observed ratings. If columns are indexed as $\{1, \ldots, d\}$, then a user u with a rating vector $x^{(u)} = (5, 4, 0, 0, 2)$, where zeros indicate missing values, has $C_u = \{1, 2, 5\}$, $H_u = \{3, 4\}$, and $x^{(u)}_{C_u} = (5, 4, 2)$.

Our first goal is to estimate a mixture model based on partially observed ratings

(a) If $x^{(u)}$ were a complete rating vector, the mixture model from Part 1 would simply say that $P(x^{(u)}|\theta) = \sum_{j=1}^{K} p_j N(x^{(u)}; \mu^{(j)}, \sigma_j^2 I)$. In the presence of missing values, however, we must use the marginal probability $P(x^{(u)}_{C_u}|\theta)$ that is over only the observed values. This marginal corresponds to integrating the mixture density $P(x^{(u)}|\theta)$ over all the unobserved coordinate values. In our case, this marginal can be computed easily. Indeed, provide a brief explanation for why

$$P(x^{(u)}_{C_u}|\theta) = \sum_{j=1}^{K} p_j N(x^{(u)}_{C_u}; \mu^{(j)}_{C_u}, \sigma_j^2 I_{|C_u|\times|C_u|}).$$

Hint: a multivariate spherical Gaussian is simply a product of univariate Gaussians pertaining to each coordinate. In other words, $N(x; \mu, \sigma^2 I) = \prod_i N(x_i; \mu_i, \sigma_i^2)$ where $N(x_i; \mu_i, \sigma_i^2)$ is just a univariate (1-dimensional) Gaussian.

(b) We need to update our EM algorithm a bit to deal with the fact that the observations are no longer complete vectors. To this end, provide an updated expression for the posterior probability $p(j|u) = P(y = j|x^{(u)}_{C_u})$.

(c) Once we have evaluated $p(j|u)$ in the E-step, we can proceed to the M-step. As before, we find the parameters that maximize the expected complete log-likelihood (weighted log-likelihood, weighted by the posterior probabilities):

$$l(X; \theta) = \sum_{u=1}^{n} \left[\sum_{j=1}^{K} p(j|u) \log\big(p_j N(x^{(u)}_{C_u}|\mu^{(j)}_{C_u}, \sigma_j^2 I_{|C_u|\times|C_u|})\big) \right]$$

where the posterior probability $p(j|u)$ can be interpreted as a blended soft assignment of $x^{(u)}_{C_u}$ across the mixture components. To maximize $l(X; \theta)$, we keep $p(j|u)$ (the soft-assignments) fixed, and maximize over the model parameters. Some of the parameters can be updated exactly as before with complete example vectors. For example,

$$\hat{p}_j = \frac{\sum_{u=1}^{n} p(j|u)}{n}.$$

But we must be more careful in updating $\mu^{(j)}$ and σ_j^2. This is because the parameters appear differently in the likelihood depending on how incomplete the observation is. Notice that some coordinates of $\mu^{(j)}$ do not appear or impact observations $x^{(u)}_{C_u}$ at all. But we can proceed to separately update each coordinate of $\mu^{(j)}$.

- Derive the update equation for $\mu^{(j)}_i$ (i^{th} coordinate). For this, it may be helpful to use an indicator function $\delta(i, C_u) = 1$ if $i \in C_u$ and zero otherwise. Note that since the data is incomplete, it is easy to "exhaust" data from some coordinates, i.e., the mixture component may focus around incomplete examples that have no values for specific coordinates. To this end, we should update the value of $\mu_i^{(j)}$ only when enough data is available, e.g., when $\sum_{u=1}^{n} p(j|u)\delta(i, C_u) \geq 1$.

The update equation for the variance is not too different from before

$$\hat{\sigma}_j^2 = \frac{1}{\sum_{u=1}^{n} |C_u| p(j|u)} \sum_{u=1}^{n} p(j|u) \|x_{C_u}^{(u)} - \hat{\mu}_{C_u}^{(j)}\|^2$$

where $|C_u|$ is the number of observed rating values from user u.

(d) We are now ready to implement the EM algorithm. Modify your E- and M-steps from Part 1 so that they work with partially observed vectors where missing values are indicated with zeros. Notice that since our toy data does not contain any zero values, your code should run as before on this data. As another check, you can try different values of K on the incomplete data `X = readData('netflix_incomplete.txt')` and make sure that the log-likelihood values are indeed monotone. If you set $K = 1$, the resulting log-likelihood of data should be -1114416.2363. Estimate a Gaussian mixture with $K = 12$ from the incomplete data matrix. As before, you will need to try several different initializations, and select the best one. Report the log-likelihood value of the best mixture.

(e) Now that we have a mixture model, how do we use it to complete a partially observed rating matrix? Provide an expression for completing a particular row, say x_C where the observed values are $i \in C$.

(f) Expand the function `fillMatrix` that takes as input an incomplete data matrix `X` as well as a mixture model, and outputs a completed version of the matrix `Xpred`.You can test the accuracy of your predictions against actual target values by loading the complete matrix `Xc = readData('netflix_complete.txt')` and measuring the root mean squared error between the two matrices `rmse(Xpred,Xc)`. Please use your best mixture from part (d) generate all the results and provide it in your write up.