CSE446 Machine Learning, Spring 2013: Homework 4

Due: Friday, May 31^{st}, beginning of class

Instructions There are 2 written questions on this assignment, plus a fourth coding question. Submit both your written answers (as a .txt or a .pdf) and your implementation to the Dropbox at `https://catalyst.uw.edu/collectit/assignment/darylh/26829/108644`

1 Manual calculation of one round of EM for a GMM [30 points]

(Extended version of: Murphy Exercise 11.7) In this question we consider clustering 1D data with a mixture of 2 Guassians using the EM algorithm. You are given the 1-D data points $x = [1\ \ 10\ \ 20]$.

M step

Suppose the output of the E step is the following matrix:

$$R = \begin{pmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{pmatrix}$$

where entry $r_{i,c}$ is the probability of observation x_i belonging to cluster c (the responsibility of cluster c for data point i). You just have to compute the M step. You may state the equations for maximum likelihood estimates of these quantities (which you should know) without proof; you just have to apply the equations to this data set. You may leave your answer in fractional form. Show your work.

1. *[5 points]* Write down the likelihood function you are trying to optimize.
2. *[5 points]* After performing the M step for the mixing weights π_1, π_2, what are the new values?
3. *[5 points]* After performing the M step for the means μ_1 and μ_2, what are the new values?
4. *[5 points]* After performing the M step for the standard deviations σ_1 and σ_2, what are the new values?

E step

Now suppose the output of the M step is the answer to the previous section. You will compute the subsequent E step.

1. *[5 points]* Write down the formula for the probability of observation x_i belonging to cluster c.
2. *[5 points]* After performing the E step, what is the new value of R?

2 PCA via Successive Deflation [30 points]

(Adapted from Murphy Exercise 12.7)

Suppose we have a set of n datapoints $x_1, \ldots, x_n$, where each x_i is represented as a d-dimensional column vector.

Let $\mathbf{X} = [x_1; \ldots; x_n]$ be the $(d \times n)$ matrix where column i is equal to x_i. Define $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ to be the covariance matrix of $\mathbf{X}$, where $\mathbf{C}_{ij} = \sum_n \mathbf{X}_{in}\mathbf{X}_{jn} = covar(i, j)$.

Next, let $v_1, v_2, \ldots v_k$ be the first k eigenvectors with larges eigenvalues of $\mathbf{C}$, i.e., the principal basis vectors. These satisfy

$$v_j^T v_k = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}$$

v_1 is the first principal eigenvector of $\mathbf{C}$ (the eigenvector with the largest eigenvalue), and as such satisfies $\mathbf{C}v_1 = \lambda_1 v_1$. Now define $\tilde{x}_i$ as the orthogonal projection of x_i onto the space orthogonal to v_1:

$$\tilde{x}_i = (\mathbf{I} - v_1 v_1^T) x_i$$

Finally, define $\tilde{\mathbf{X}} = [\tilde{x}_1; \ldots; \tilde{x}_n]$ as the $(d \times n)$ **deflated matrix** of rank one less than $\mathbf{X}$, which is obtained by removing from the d-dimensional data the component that lies in the direction of the first principal eigenvector:

$$\tilde{\mathbf{X}} = (\mathbf{I} - v_1 v_1^T)\mathbf{X}$$

1. *[10 points]* Show that the covariance of the deflated matrix, $\tilde{\mathbf{C}} = \frac{1}{n}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ is given by

$$\tilde{\mathbf{C}} = \frac{1}{n}\mathbf{X}\mathbf{X}^T - \lambda_1 v_1 v_1^T$$

   (Hint: Some useful facts: $(\mathbf{I} - v_1 v_1^T)$ *is symmetric,* $\mathbf{X}\mathbf{X}^T v_1 = n\lambda_1 v_1$*, and* $v_1^T v_1 = 1$*.*
   Also, for any matrices A and B, $(AB)^T = B^T A^T$*.*
   Furthermore, $v_1^T \mathbf{X}\mathbf{X}^T = n\lambda_1 v_1^T$*)*
2. *[10 points]* Show that for $j \neq 1$, if v_j is a principal eigenvector of $\mathbf{C}$ with corresponding eigenvalue λ_j *(that is,* $\mathbf{C}v_j = \lambda_j v_j$*)*, then v_j is also a principal eigenvector of $\tilde{\mathbf{C}}$ with the same eigenvalue λ_j.
3. *[5 points]* Let u be the first principal eigenvector of $\tilde{\mathbf{C}}$. Explain why $u = v_2$. (You may assume u is unit norm.)
4. *[5 points]* Suppose we have a simple method for finding the leading eigenvector and eigenvalue of a positive-definite matrix, denoted by $[\lambda, u] = f(\mathbf{C})$. Write some pseudocode for finding the first K principal basis vectors of $\mathbf{X}$ that only uses the special f function and simple vector arithmetic.
   (Hint: This should be a simple iterative routine that takes 2-3 lines to write. The input is $\mathbf{C}, K$*, and the function* f*, the output should be* v_j *and* λ_j *for* $j \in [1:K]$*.)*

3 Programming Question (clustering with K-means) [40 points]

In class we discussed the K-means clustering algorithm. Your programming assignment this week is to implement the K-means algorithm on a different subset of the digit data you worked with last week.

3.1 The Data

There are two files with the data. The first

`digitdata.txt`

contains the 1000 observations of 157 pixels (a subset of the original 785) concerning handwritten digits. The second file

`digitlabels.txt`

contains the true digit label (either 1, 3, 5, or 7). You can read both data files in with

```
data <- read.table("digitdata.txt")
truelabs <- read.table("digitlabels.txt")
```

3.2 The algorithm

Your algorithm should be implemented as follows:

1. Randomly select k starting centers from your data set.
2. Assign each data point to the cluster associated with the nearest of the k center points.
3. Re-calculate the centers as the mean vector of each cluster from (2).
4. Repeat steps (2) and (3) until convergence or iteration limit.

Define convergence as no change in label assignment from one step to another **or** you have iterated 20 times (whichever comes first). Since there are 158 features this algorithm may take a couple of minutes to run. As such I will provide some useful R functions below that may help.

3.3 Within group sum of squares

The goal of clustering can be thought of as minimizing the variation within groups and consequently maximizing the variation between groups. A good model has low sum of squares within each group. We define sum of squares in the traditional way. Let C_k be the kth cluster and let μ_k be the empirical mean of the observations x_i in cluster C_k. Then the within group sum of squares for cluster C_k is defined as:

$$SS(k) = \sum_{i \in C_k} (x_i - \mu_{C_k})^2$$

Then if there are K clusters total then the "sum of within group sum of squares" is just the sum of all K of these individual $SS(k)$ terms.

3.4 Mistake Rate

Given that we know the actual assignment labels for each data point we can attempt to analyze how well the clustering recovered this. For cluster C_k let its assignment be whatever the majority vote is for that cluster. For example if for one cluster we had 270 observations labeled **one**, 50 labeled **three**, 9 labeled **five**, and 0 labeled **seven** then that cluster will be assigned value **one** and had $50 + 9 + 0 = 59$ mistakes for a total mistake rate of $59/(270 + 59) = 17.93\%$. If we add up the total number of "mistakes" for each cluster and divide by the total number of observations (1000) we will get our total mistake rate.

3.5 Questions

When you have implemented the algorithm please submit the following:

1. *[20 points]* Your solution code.
2. *[10 points]* A plot of the sum of within group sum of squares versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.
3. *[10 points]* A plot of total mistake rate versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

3.6 Some useful functions

First you may find it useful that functions in R can return lists. If, for example, your function creates several variables that you wish to return (say within group sum of squares and the labels) you can return:

```
results <- list(labels = labels, withinss = withinss)
return(results)
```

Then to access each item we use the $ sign. For example if the function was called "kMeans(...)" we could do the following

```
myresults <- kMeans(...)
labels <- myresults$labels
```

For finding distance you should use the built in

`dist()`

function which returns the euclidean distance between two rows of a matrix. For the minimum distance you should remember

`which.min()`

If you have your data stored in a data frame called "data" and you have a function "nearest(x, centers)" which takes a point x and a matrix "centers" where each row is a center of one cluster and the function returns which row (center) the point x is closest to, then the following function will return the nearest cluster for each data point:

```
assignCenters <- function(data, centers){
  labels <- apply(data,1, FUN = nearest, centers)
  return(labels)
}
```

Then if you have a vector called "labels" which says says which cluster $(1, \ldots, K)$ each element of data is assigned to, then we can define the new centers as follows

```
newCenters <- function(data, labels){
  centers <- aggregate(data, list(label = labels),mean)
  centers <- as.matrix(centers[,-1])
  return(centers)
}
```

You may want to use a similar version of "aggregate(...)" with a different function (that you can write yourself) to calculate the within group sum of squares.