STAT 339: HOMEWORK 7 (CLUSTERING)

OPTIONAL "BONUS" ASSIGNMENT

Instructions. The final grade will be calculated based on the highest of the following three calculations:

(1) 50% of the average of the top five out of seven homework grades + 20% of the project grade + 20% of the midterm grade + 10% of the participation grade (closest to the syllabus: two lowest homeworks dropped, and relative weight of homework preserved)

(2) 60% of the average of the top five out of seven homework grades + 15% of the project grade + 15% of the midterm grade + 10% of the participation grade (increased weight to homeworks if they are higher than the average of the midterm and project grades)

(3) 50% of the average of the top four out of six homework grades + 20% of the project grade + 20% of the midterm grade + 10% of the participation grade (equivalent to if this homework didn't exist)

Create a directory called `hw7` in your `stat339` GitHub repo. Your main writeup should be called `hw7.pdf`.

You may also use any typesetting software to prepare your writeup, but the final document should be a PDF. LaTeXis highly encouraged.

I will access your work by cloning your repository; make sure that any file path information is written relative to your repo – don't use absolute paths on your machine, or the code won't run for me!

All data files referred to in the problems below can be found at

`http://colindawson.net/data/<filename>.csv`.

Date: Due alongside the final project.

1. Clustering with a Mixture of Normals Model

Consider a mixture of D-dimensional multivariate Normals model with K mixture components, where K is specified in advance.

The data consists of N vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_N$, each with D real-valued entries, representing observable features.

1. **Implementing K-Means** Write a function called `kmeans()`, which takes the following arguments:

`X`	the $N \times D$ data matrix $\mathbf{X}$ to be clustered
`K`	an integer (2 or larger) giving the number of clusters
`initialization`	the name of an initialization function (which in turn can be expected to take a data matrix `X` and a number of clusters `K`, and perhaps other optional inputs)

   The function should do the following:

   (i) Standardize the columns of $\mathbf{X}$ as z-scores, so that Euclidean distances between points and centers are meaningful (store the means and standard deviations so that the cluster centers can be transformed back at the end)

   (ii) Call the initialization function passed to `intialization=` to get an initial assignment of points to clusters

   (iii) Run one pass of K-means, terminating when $\mathbf{z}$ stops changing

   Your function should return a dictionary with the following entries:

`"z"`	an array of cluster indicators
`"mu"`	a $K \times D$ array whose rows are cluster centers

2. **Plotting Clusters** Write a plotting function, `plotClusters()`, with the following arguments:

`X`	an $N \times D$ dataset $\mathbf{X}$
`z`	the vector $\mathbf{z}$ indicating the cluster assignment of each data points in `X`
`mu`	a $K \times D$ matrix whose kth row is the mean vector for cluster k
`dims`	a tuple of two column indices, d_1 and d_2, each between 1 and D, which define the plot axes

   The function should produce a scatterplot of columns d_1 and d_2 of the data, color coded by the clusters assigned in $\mathbf{z}$, overlaying the d_1 and d_2 coordinates of the cluster centers in a distinct symbol.

3. **Testing K-Means with the Iris Data** Test your `kmeans()` and plotting function using the iris data (complete data with labels in `iris.csv`, or if you prefer, features only in `iris_features.csv` and labels only in `iris_labels.csv`), including the plot corresponding to the best of 10 runs of the function

4. **Mixture of Normals with EM for Maximum Likelihood** Write a function, `clusterNormalMix()` that does clustering with a mixture of Normals model, with parameters estimated using local Maximum Likelihood as per the EM algorithm. The unknown parameters are $\boldsymbol{\theta}$, which consists of

   - the D-dimensional mean vectors for each cluster: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
   - the $D \times D$ covariance matrices for each cluster, $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$

   and $\boldsymbol{\pi}$, the vector of mixture weights: $\boldsymbol{\pi} = (\pi_1, \ldots, \boldsymbol{\pi}_K)$ where each entry π_k is scalar, and the entries collectively must sum to 1

   The inputs to your function should be the same as the inputs for the K-means function, plus an additional input to specify the covariance structure of the Normals. This `covariance` argument should allow the following possibilities:

`"fixed"`	No learning of covariance matrices (e.g., just set the covariances to the identity matrix in the "standardized" feature space)
`"diagonal"`	Each $\mathbf{\Sigma}_k$ is assumed diagonal, but the diagonal entries can vary by cluster (k) and by feature (d). This is equivalent to making the "naive Bayes" approximation, in which each feature is conditionally independent of the others given the cluster assignment — that is, each coordinate of $\mathbf{x}_n$ is modeled as an independent univariate Normal distribution whose parameters are estimated separately within each M step
`"unconstrained"`	The covariance matrices can have arbitrary structures, with the entries estimated as per the equations shown in class

Implementation Tip: For numerical stability, you will likely want to make use of the following identities for computing the log sums of small probabilities (which will likely produce numerical underflow if calculated naively):

$$\begin{aligned}\log(\sum_{m=1}^{M}\prod_{n=1}^{N} p_{mn}) &= \log(\sum_{m=1}^{M}\exp\{\sum_{n=1}^{N}\log(p_{mn})\}) \\ &= \log(\sum_{m=1}^{M}\exp\{C+\sum_{n=1}^{N}\log(p_{mn})\}\exp\{-C\}\}) \\ &= -C+\log(\sum_{m=1}^{M}\exp(C+\sum_{n=1}^{N}\log(p_{mn})))\end{aligned}$$

where C can be any number but is chosen to make the terms being exponentiated not too far from zero (for example, since log probabilities are negative, we might choose $C = \min_m \sum_{n=1}^{N}\log(p_{mn})$).

The return value should also be a dictionary with the following entries:

`"params"`	a dictionary containing the final parameter estimates, with entries, `"pi"`, `"mu"`, and `"Sigma"`, which are a one-dimensional array of length K, a $K \times D$ array whose rows are the cluster means, and a $K \times D \times D$ array of cluster covariances, respectively
`"Q"`	The $N \times K$ array of "soft" clustering assignments based on the final parameter estimates (essentially, do one extra "E"-step after convergence and return the $\mathbf{Q}$ matrix)
`"logLik"`	An array of "per point" log likelihoods (i.e., overall log likelihood divided by sample size); one entry per iteration (for diagnostic purposes — the values should be strictly decreasing if there is not a bug in the code)
`"lwr"`	An array whose entries consist of the Jensen's inequality lower bound on the log likelihood computed at each iteration (also for diagnostic purposes)

5. **Testing the using the Iris dataset** Test the Mixture of Normals model with EM estimated Maximum Likelihood on the iris data for a few small values of K (say between 2 and 5), plotting the log likelihood and lower bound over iterations, as well as plotting the data (two features at a time), color-coded by the most likely cluster (if you're feeling fancy you could try to use a blended color scheme that takes soft clustering into account, but you don't have to do this). Comment on what effect the choice of K has on the final log likelihood.

6. **Cross-Validation Using Log Likelihood** Implement J-fold cross-validation to select the best value of K for the EM case, using the per-point log likelihood of the validation set as the performance metric. You should be able to call your code from HW1, passing `clusterNormalMix` as the training function, and a function that calculates the per point log likelihood as the "error" metric.

   Your code should plot the training set and validation set per-point log likelihoods.

   Does your cross-validation procedure select a value of K which is close to the true number of iris species in the data (i.e., $K = 3$)?

7. **Applying clustering to cancer microarray data**

   This problem should not require any new implementation; just applying what you did in the previous problem to a more interesting dataset.

   The file `nci60_reduced.csv` comes from a dataset in which cell lines from 64 cancerous tumors were analyzed using a "microarray", which measures the degree to which particular genes are expressed in the sample. In the original data, 6830

gene expression measurements were taken from every cell line. However, we do not want to cluster data with this many dimensions using the simple techniques we have learned; so I have preprocessed the data using Principle Components Analysis (PCA) to reduce the number of dimensions to 4.

(a) Using a mixture of Normals, estimating parameters using EM, and using 10-fold cross-validation, find a suitable number of clusters for this data, and plot the first two dimensions of the data, labeling by highest probabilty cluster for your final choice of K.

(b) The diagnosed cancer types are listed in the file `nci60_labels.csv`. Investigate the extent to which your clusters line up with the human-provided catgories. (There is no one correct way to measure this; be creative)