Boston University
Department of Electrical and Computer Engineering

***ENG EC 414 Introduction to Machine Learning***

# HW 8 Solution



**Issued:** Fri 30 Oct 2020 **Due:** 10:00am Fri 6 Nov 2020 in Gradescope (non-code) + Blackboard

**Important:** Before you proceed, please read the documents pertaining to *Homework formatting and submission guidelines* in the Homeworks section of Blackboard. **In particular, for computer assignments you are prohibited from using any online code or built-in MATLAB functions except as indicated in the problem or skeleton code (when provided).**

**Important:** To obtain full grade, please clearly motivate all your answers.

**Note:** Problem difficulty = number of coffee cups ☕

**Problem 8.1** [36pts] *(k-means implementation)* In this problem we will implement $k$-means clustering and explore the impact of initialization and number of clusters on one synthetic and one real-world dataset. We will also explore a dataset where $k$-means will fail to produce meaningful clusters.

(a) [8 points] *(Implement k-means)* Implement the $k$-means algorithm we saw in class in Matlab with prototype

```
[c,obj,y] = k_means_clustering(X, c0, T)
```

where `c` are the returned centers $\in \mathbb{R}^{k\times d}$, `obj` is the final value of the objective function, `y` are the inferred "labels", `X` is the matrix of the training samples $\in \mathbb{R}^{m\times d}$, `c0` are the initial centers $\in \mathbb{R}^{k\times d}$, `T` is the maximum number of iterations. Note that there is no need to pass $k$ because the algorithm can infer it from the size of `c0`.

(b) [4pts] *(Synthetic training set generation)* Generate 3 two-dimensional Gaussian clusters of data points having the following mean vectors and covariance matrices: $\boldsymbol{\mu}_1 = [2,2]^\top$, $\boldsymbol{\mu}_2 = [-2,2]^\top$, $\boldsymbol{\mu}_3 = [0,-3.25]^\top$, and $\Sigma_1 = 0.02\cdot I_2$, $\Sigma_2 = 0.05\cdot I_2$, $\Sigma_3 = 0.07\cdot I_2$, where $I_2$ is the $2\times 2$ identity matrix. You can use `mvnrnd` to generate multivariate Gaussian noise. Let each data cluster have 50 points. Plot the generated Gaussian data. Color the data points in the 1st, 2nd, and 3rd clusters with red, green, and blue colors, respectively. You can use `gscatter` to easily plot the points in different colors. Use your implementation of $k$-means on this dataset with $k = 3$ and the following **initialization**: $\boldsymbol{c}_1^{\text{initial}} = [3,3]^\top$, $\boldsymbol{c}_2^{\text{initial}} = [-4,-1]^\top$, $\boldsymbol{c}_3^{\text{initial}} = [2,-4]^\top$. and the maximum number of iterations equal to 10. Plot the clusters produced by your $k$-means algorithm, plotting the points of each cluster with a different color.

(c) [4pts] *(Effect of different initialization)* Using the same synthetic training dataset from part (b), re-run your $k$-means algorithm implementation for $k = 3$ using the following (different) **initialization:** $\boldsymbol{c}_1^{\text{initial}} = [-.14, 2.61]^\top$, $\boldsymbol{c}_2^{\text{initial}} = [3.15, -0.84]^\top$, $\boldsymbol{c}_3^{\text{initial}} = [-3.28, -1.58]^\top$. Create a new plot of the resulting clusters. Discuss what you observe.

(d) [10pts] *(Best of multiple random initializations)* To reduce the possibility selecting an initialization which results in a "bad" clustering, the $k$-means algorithm is typically run multiple times using different random initializations. The best clustering result, i.e., the one having the smallest objective function is saved and used as the final output. Run your implementation of the $k$-means algorithm on the same synthetic training dataset from part (b) for 10 different random initializations. Report the objective function for each of the 10 trials. Identify the trial which yields the smallest objective function value. Report its objective function value and create a plot of the clustering produced by it.

(e) [6pts] *(Clustering a real-world dataset)* Here we examine a real-world digits, MNIST. The inputs are 28 x 28 images of digits flattened to vectors of size 784. Ignore the labels and apply your implementation of the $k$-means algorithm over the training data with $k = 10$ selecting the best of 10 different random initializations as the final output. Reshape the centers as images and plot the images corresponding to the 10 centers. Hint: you can reshape and display the first training sample with the following command `imagesc(reshape(Xtr(1,:),28,28)')`. Also, if your implementation of $k$-means is too slow, take a subset of the training set.

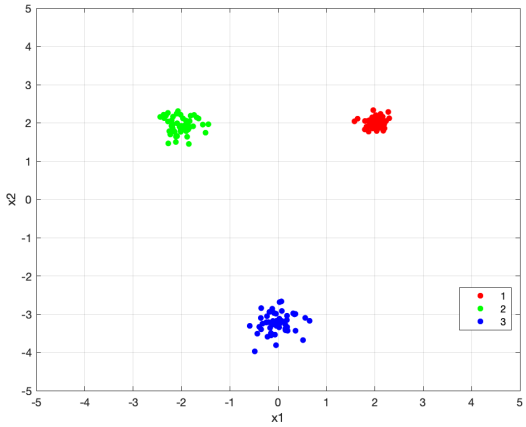
(f) [4pts] *(Failure of k-means)* Here we examine the performance of the $k$-means algorithm on a dataset composed of 3 concentric rings. Use `sample_circle.m` to generate a dataset with 3 concentric ring clusters and 500 points for each cluster. Plot the dataset. Apply your implementation of the $k$-means algorithm on this dataset using $k = 3$ and choosing the best of 10 different random initializations. Create a plot of the best clustered results. Discuss what you observe.

**Solution:**

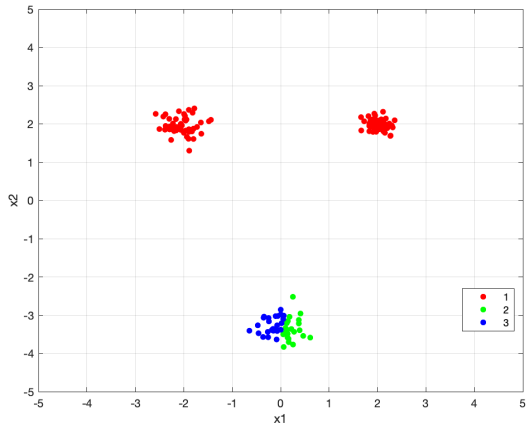
(a) [8 points] See code.

(b) [4pts] Figure 1(a) shows the scatter plot of the generated Gaussian data together with the clusters produced by running 3-means with the initialization provided in the problem. Observe that the three clusters are tightly clustered around the mean vectors of their corresponding Gaussian distributions. All clusters are roughly spherical in shape (rather than an elongated ellipsoid). This is consistent with the fact that the covariance matrices of the 3 Gaussians are multiples of the identity matrix. The dispersion (spread) of data points around each mean vector is highest for cluster 3 followed by cluster 2 and then cluster 1. This is consistent with the ordering of the scalar multiplicative factor associated with the covariance matrices of the 3 Gaussians that were specified in the problem. The initial mean vector for each cluster is very close that cluster. The clusters are also very well separated. So it should not come as a surprise that the 3-means algorithm succeeds in discovering the correct clusters with these choices for initial mean vectors.

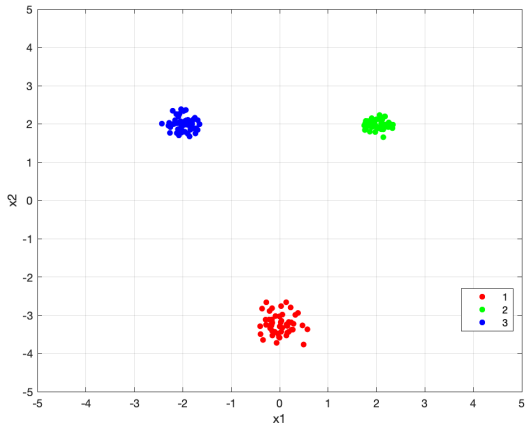
(c) [4pts] The clustering produced by the alternative choice of initial mean vectors is depicted in Figure 1(b). Observe that the two clusters at the top have now been (incorrectly) merged into a single cluster and the bottom monolithic cluster has been (incorrectly) split into two clusters. This is consistent with the locations of the initial mean vectors. The initial mean vector $[-.14, 2.61]^\top$ is right in between the two clusters on the top. That explains why those two clusters got merged into a single cluster. Similarly, the initial mean vector $[3.15, -0.84]^\top$ is closest to the right-half of the bottom cluster and the initial mean vector $[-3.28, -1.58]^\top$ is closest to the left-half of the bottom cluster resulting in the bottom cluster being split into two clusters. Overall, this part shows that poor choices for initial mean vectors could result in poor clustering results produced by $k$-means.




(a)




(b)




(c)

**Figure 1:** *(a) Synthetic dataset of 3 Gaussians in two-dimensional space. (b) Result of* 3*-means clustering using initialization of part 4.2(b). (c) Result of* 3*-means clustering using the best of* 10 *different random initializations.*

(d) [10pts]

| Objective function |
|---|
| 417.1567 |
| 14.4908 |
| 417.1567 |
| 417.1567 |
| 14.4908 |
| 417.1567 |
| 14.4908 |
| 417.1567 |
| 417.1567 |
| 14.4908 |

The best clustering result (corresponding to the smallest objective function = 14.4908) is shown in Figure 1(c). Notice that trying multiple random initializations improves the odds of discovering a good clustering. Also notice that multiple choices of the initial mean vectors could result in the same final clustering (and therefore the same objective value). In the problem we suggested choosing the coordinates of the initial centers uniformly at random between their minimum and maximum values observed in the dataset. An alternative strategy is to choose initial centers from among the data points themselves uniformly at random. A better strategy would be to use $k$-means++.

(e) [6pts] Figure 2 shows the centers produced by (the best of 10 runs of) the $k$-means algorithm on

the real-world MNIST dataset together. Observe that, unlike in the Gaussian dataset, here we don't recover a cluster for each digit. In particular, it seems that some digits were split in two clusters (for example "1" and "9") and possibly contaminated by other digits. This problem shows how difficult is in the real-world to recover meaningful clusters from pure unsupervised learning algorithm.

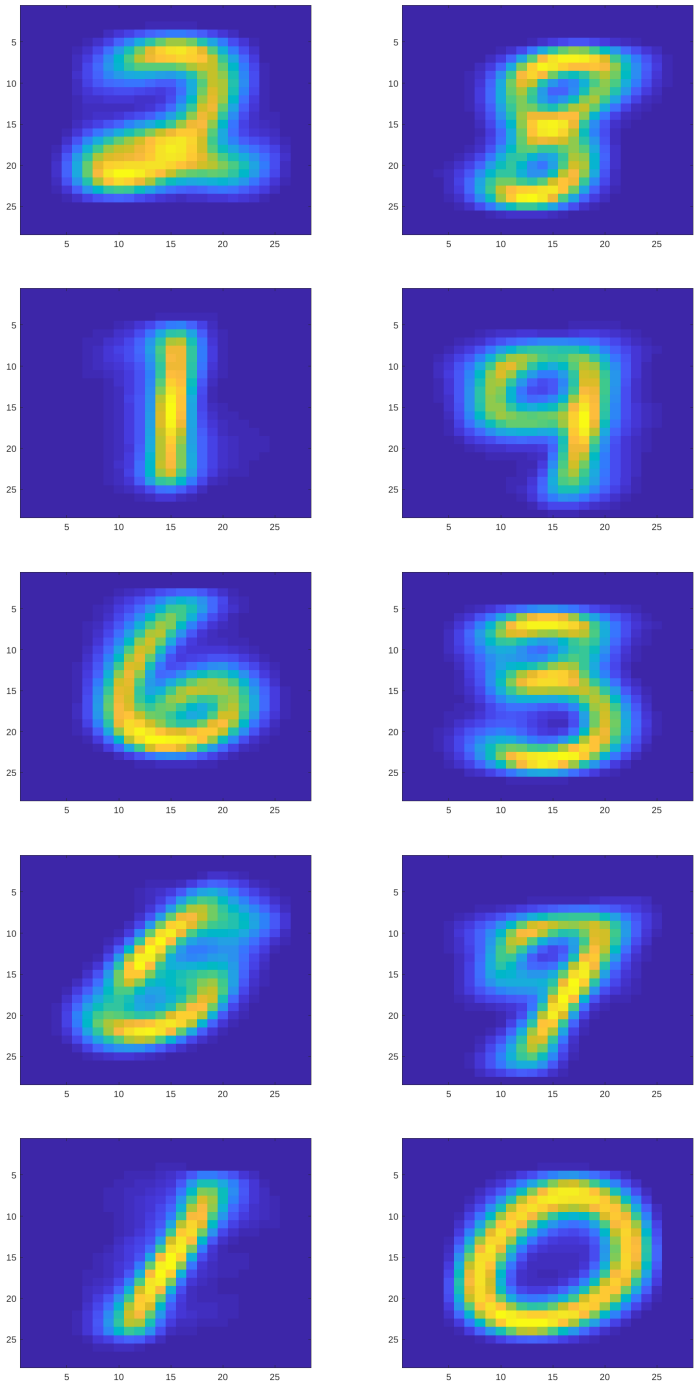


**Figure 2:** *Centers of the k-means algorithm on the MNIST dataset.*

(f) [4pts] Figure 3 shows the clusters produced by running the 3-means algorithm on the dataset of 3

concentric rings. Observe that $k$-means fails to correctly cluster this dataset as it is unable to capture the underlying ring geometry. More concretely, we know that for any placement of 3 centers, their Voronoi cells will be polytopes (intersection of a finite number of half-spaces) which have linear facets. Such linear facets are simply incapable of approximating a ring-shaped region. Other methods, e.g., kernel $k$-means, which we will study later this semester, must be used to correctly cluster this type of data.

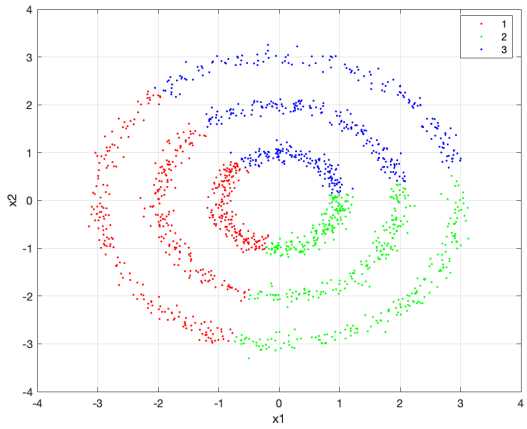



**Figure 3:** *Three concentric rings dataset and clusters produced by running* 3*-means algorithm.*

**Code-submission via Blackboard:** You must prepare 1 files: `k_means_clustering.m` for Problem 8.1(a). Place them in a **single** directory which should be zipped and uploaded into Blackboard. Your directory must be named as follows: `<yourBUemailID>_hwX` where `X` is the homework number. For example, if your BU email address is `charles500@bu.edu` then for homework number 8 you would submit a single directory named: `charles500_hw8.zip` which contains all the MATLAB code (and only the code).

Reach-out to the TAs via Piazza and their office/discussion hours for questions related to coding.