Due date: Sep. 30, 2020, 11:59 PM (Arlington time). ou have **two** late days — use it at as you wish. Once you run out of this quota, the penalty for late submission will be applied. You can either use your late days quota (or let the penalty be applied). **Clearly indicate** in your submission if you seek to use the quota.

What to turn in:

1. Your submission should include your complete code base in an archive file (`zip, tar.gz`) and `q1/, q2/,` and so on), and a very clear README describing how to run it.
2. A report (**typed up with LaTeX**, submit as a PDF file, NO handwritten scanned copies) describing what you solved and implemented and perhaps known failure cases. The report is **important** since we will be evaluating the grades mostly based on the report.
3. Submit your entire code and report to Canvas.

Notes from instructor:

- Start early!
- You may ask the TA or instructor for suggestions, and discuss the problem with others (minimally). But **all parts of the submitted code must be your own**.
- Use Matlab or Python for your implementation.

Problem 1

(k-means, **30pts**) Implement your own k-means clustering algorithm.

Use the following parameters to generate 2D Gaussian random samples for different clusters.

$$\mu_1 = [-3, 0],\ \mu_2 = [3, 0],\ \mu_3 = [0, 3],\ \Sigma = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix} \tag{1}$$

1. **(10pts)** Write a function `cluster = mykmeans(X, k)` that clusters data X into k clusters. Use ℓ_2-norm of vectors to compute distance between different samples.
2. **(10pts)** Generate X using the parameters above, and you should know the ground truth for cluster centers and cluster assignments. (You need to show data from 3 different clusters simultaneously.) Generate $N = 300$ for each cluster and test your implementation on the synthetic data with different $k = 2, 3, 4, 5$. Visualize the changes of clustering result and cluster centers, compute clustering accuracy when $k = 3$. How accurate are the estimated cluster centers when $k = 3$? Summarize observations of all the results in your own words.
3. **(10pts)** Now, change the μ to $\mu_1 = [-2, 0]$, $\mu_2 = [2, 0]$, $\mu_3 = [0, 2]$, generate new X and test your k-means algorithm. Is this X easier or harder than the previous case? Generate $N = 300$ for each cluster and test your implementation on the synthetic data with different $k = 2, 3, 4, 5$. Visualize the changes of clustering result and cluster centers, compute clustering accuracy when $k = 3$. How accurate are the estimated cluster centers when $k = 3$? Summarize observations of all the results in your own words.

Problem 2

(k-nearest neighbors, **70pts**) Implement your own k-NN algorithms for classification.

1. **(20pts)** Write a function `class = myknnclassify(train, test, k)` that classifies the class of input `test` given a training set `train` using k-NN classifier where k is the number of neighbors. Use the following parameters to generate 2D Gaussian random samples for different clusters.

$$\mu_0 = [1, 0],\ \mu_1 = [0, 1], \Sigma_0 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix},\ \Sigma_1 = \begin{bmatrix} 1 & -0.5 \\ 0.5 & 1 \end{bmatrix} \tag{2}$$

   Generate $N = 200$ training samples for each class and assign class labels 0 and 1 for each class. You can do the same for testing samples; generate $N = 50$ for each class for testing set to evaluate your model.

   The first 2 columns in `train` should be the 2D Gaussian random data and the last column should be the class label. Use ℓ_2-norm of vectors to compute distance between different samples. Try out different $k = 1, 2, 3, 4, 5, 10, 20$ to test your k-NN classifier. Show the changes of accuracy w.r.t. the k. Summarize your observation.

2. **(20pts)** Write a function `value = myknnregress(X, test, k)` that that regresses the target value of input `test` given a training set `train` using k-NN regressor where k is the number of neighbors. This time, use the following parameters to generate 2D Gaussian random data **x**

$$\mu_0 = [1, 0], \Sigma_0 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix} \tag{3}$$

   and assign target values as $y = 2x_1 + x_2 + \epsilon$ where ϵ is a Gaussian random noise with $\sigma = 0.5$. Generate $N = 300$ samples for training and 100 for testing sets. The first 2 columns in `train` should be the 2D Gaussian random data and the last column should be the target value. Use ℓ_2-norm of vectors to compute distance between different samples. Try out different $k = 1, 2, 3, 5, 10, 20, 50, 100$ to test your k-NN classifier. Show the changes of accuracy in average ℓ_2-norm w.r.t. the k. Summarize your observation.

3. **(30pts)** Write a function for locally weighted regression `[value weight] = myLWR(X, test, k)` that classifies the class of an input `test` given a training set X using k-NN classifier where k is the number of neighbors. The data should be generated in the same using the parameters in P2-2. As we have not discussed Gradient Descent in the class yet, you can simply use Least Squares to estimate the weights. That is, to find optimal W^* for $Y = XW$, you may simple use the following:

$$(X^T X)^{-1} X^T Y = (X^T X)^{-1}(X^T X)W^* \tag{4}$$

   Use ℓ_2-norm of vectors to compute distance between different samples. Try out different $k = 1, 2, 3, 5, 10, 20, 50, 100$ to test your k-NN classifier. Show the changes of accuracy in average ℓ_2-norm w.r.t. the k. Summarize your observation.