CS 422/622 Project 2

Due 10/18 11:59pm

Logistics: You must implement everything stated in this project description that is marked with an **implement** tag. Whenever you see the **write-up** tag, that is something that must be addressed in the write-up for the project. You may import the numpy and scipy.spatial (for euclidean distance) libraries. A test_script.py file is provided to test your functions with. **You must implement everything from scratch.**

Deliverables: You should submit a single ZIP file, containing your project code ($*$.py files) and your writeup (PDF or README.txt). Your zip file should be named lastname1_firstname2_project2.zip. For example, a zip file for Sara Smith might look like smith_sarah_project2.zip. Your code should run without errors on the ECC linux machines. If your code does not run for a particular problem, you will lose 50% on that problem. You should submit three py files, named accordingly (e.g. nearest_neighbors.py, clustering.py, and perceptron.py).

Extra Credit: Students in 422 who use LaTeX for their write-up and submit their LaTeX source files will get extra points.

1 Nearest Neighbors (30 points)

File name: nearest_neighbors.py

Implement a function in python:

`KNN_test(X_train,Y_train,X_test,Y_test,K)`

that takes training data, test data, and K as inputs. `KNN_test(X_train,Y_train,X_test,Y_test,K)` should return the accuracy on the test data. The training data and test data should have the same format as described earlier for the decision tree problems. Your function should be able to handle any dimension feature vector, with real-valued features. Remember your labels are binary, and they should be -1 for the negative class and 1 for the positive class. Two csv files are provided with test data.

Write-Up: Using the following training data, how would your algorithm classify the test points listed below with K=1, K=3, and K=5?
$test_1 = (1, 1, 1)$
$test_2 = (2, 1, -1)$
$test_3 = (0, 10, 1)$
$test_4 = (10, 10, -1)$
$test_5 = (5, 5, 1)$
$test_6 = (3, 10, -1)$
$test_7 = (9, 4, 1)$
$test_8 = (6, 2, -1)$
$test_9 = (2, 2, 1)$
$test_{10} = (8, 7, -1)$

622 Implement the following function in python:

`choose_K(X_train,Y_train,X_val,Y_val)`

that takes training data and validation data as inputs and returns a K value. This function must iterate through all possible K values and choose the best K for the given training data and validation data. K must be chosen in order to achieve the maximum accuracy on the validation data. **Write-Up**: What is the best K value for the training data above?

2 Clustering (30 points)

File name: clustering.py

Implement a function in python:

`K_Means(X,K)`

that takes feature vectors `X` and a `K` value as input and returns a numpy array of cluster centers `C`. Your function should be able to handle any dimension of feature vectors and any $K > 0$. mu is an array of initial cluster centers, with either K or 0 rows. If mu is empty, then you must intialize the cluster centers randomly. Otherwise, start with the given cluster centers.

Write-Up: Test your function on the following training data, with K=2 and K=3.

Sample	x1	x2
1	1	0
2	7	4
3	9	6
4	2	1
5	4	8
6	0	3
7	13	5
8	6	8
9	7	3
10	3	6
11	2	1
12	8	3
13	10	2
14	3	5
15	5	1
16	1	9
17	10	3
18	4	1
19	6	6
20	2	2

622 Implement the following function in python:

`K_Means_better(X,K)`

that takes feature vectors `X` and a `K` value as input and returns a numpy array of cluster centers `C`. Your function should be able to handle any dimension of feature vectors and any $K > 0$. Your function will run the above-implemented `K_Means(X,K)` function **many** times until the same set of cluster centers are returned a majority of the time. At this point, you will know that those cluster centers are likely the best ones. `K_Means_better(X,K)` will return those cluster centers.

Write-Up: Test your function with K=2 and K=3 on the above data. Plot your clusters in different colors and label the cluster centers.

3 Perceptron (30 points)

File name: perceptron.py

Implement a function in python:

`perceptron_train(X,Y)`

that takes training data as input and outputs the weights `w`, and the bias `b` of the perceptron. Your function should handle any real-valued features, with feature vectors in any dimension, and binary labels.

Implement a function in python:

```
perceptron_test(X_test, Y_test, w, b)
```

that takes testing data, the perceptron weights and bias as input and returns the accuracy on the testing data. I will use a script similar to the provided test_script. Note that I will test many different scenarios.

Write-Up: Train your perceptron on the following dataset. Using the w and b you get, plot the decision boundary.

Sample	x1	x2	y
1	-2	1	1
2	1	1	1
3	1.5	-0.5	1
4	-2	-1	-1
5	-1	-1.5	-1
6	2	-2	-1

4 Report (10 points)

622 is required to submit their report using LaTeX as a PDF and the source file. 422 can submit a README.txt or a LaTeX for extra credit.