EE241 Final Project
Fall 2020

due: February 12, Saturday, 23:55

Reminder: By returning this homework assignment you have agreed that you, in person, are fully responsible for the consequences of violating the rules of conduct. As a university student, you are expected to act maturely and responsibly. In short, **do not cheat!** Your submission will be checked automatically (and manually, if needed) for plagiarism.

After doing - before submitting - your homework, you must write the following statement on paper by hand, include your name and the date, sign and scan the paper (into a pdf file, e.g., by using a mobile scanner application), and include the pdf file among the other files to be submitted: **"I have neither given nor received any unauthorized aid on this assignment."**

Please upload all the relevant files as a single zip archive on Moodle until February 12, Saturday, 23:55.
No late homeworks!

MATLAB

Implementing the *k*-Nearest Neighbor (*k*-NN) classifier algorithm

Every *1*x*d* (or *d*x*1*) vector can be considered as a point in the d-dimensional space. Assume that each such point is an instance (e.g., a fish) belonging to a class (e.g., sardines or tuna fish), and each element of the vector is the value of some attribute (e.g., *d=2, x-axis*: length of the fish, *y-axis*: width of the fish). Then imagine: the instances of the two classes, sardines and tuna fish, would form two visually separable clusters in the two-dimensional space, sardines lying closer to the origin.

The simplest classifier is the *k*-Nearest Neighbor (*k*-NN) classifier. Given an instance with an unknown class label, *k*-NN classifier looks at *k* closest neighbors (with known labels) of this vector in the space and decides that this vector comes from class 1 if most of the neighbors are coming from class 1 (similarly, decides for class 2 if the majority votes for class 2, and so on...)

The set of instances with the known labels is called the *training set* since they are used to train the classifier. To validate the classifier, we use a different set called the *validation set*. Actually, we also know the class labels of the validation instances, however we behave as if we do not know them; we let the algorithm to predict them using the training instances, then we compare the predicted labels with the known labels, calculate the correct classification rate per class, and build the confusion table.

A typical confusion table is given below.

	Class 1 (predicted)	Class 2 (predicted)
Class 1 (actual)	Percentage of correctly classified class 1 instances over all class 1 instances. (Correct classification rate of class 1)	Percentage of instances classified as class 2 while they come from class 1, over all class 1 instances.
Class 2 (actual)	Percentage of instances classified as class 1 while they come from class 2, over all class 2 instances.	Percentage of correctly classified class 2 instances over all class 2 instances. (Correct classification rate of class 2)

We can summarize the overall success of the classifier using the total correct classification rate (TCC, also: *accuracy*) given as

$$TCC\ (\%) = \frac{\#all\_correctly\_classified\_instances}{\#all\_instances}$$

EE241 Final Project
Fall 2020

due: February 12, Saturday, 23:55

Your job

Write a function that will predict the unknown class label using the *k*-NN classifier algorithm. You can assume there will be two classes. Use the Euclidean distance measure. The function should accept as input:

- the training set (the set including those instances with known labels, e.g., a *30*x*2* matrix for *15* sardines and *15* tuna fish, *d*=*2*),
- the label vector (the class labels of the training instances, e.g., a *30*x*1* vector),
- the instance with the unknown label (e.g., a *1*x*2* vector),
- the value of *k*,

and give as the output:

- the predicted class label of the instance.

Write a script that will call this function:

- to predict the class labels of all the instances given in the validation set,
- to calculate *TCC* per given *k* value,
- to plot *TCC* vs *k,*
- to print (on the command window):
    - the *k* value with the maximum *TCC*, and
    - the corresponding the confusion matrix.

Sweep over the values of *k* from *1* to *35* (*Hint: k should be an odd value when you have two classes*).

Notes:

- You may decide your own way of writing your script/function. Correct outcome and efficiency should be your target.
- The training and validation sets for this project are given in the mat-file named DATA.mat. The third columns have the class labels. The data is arbitrary, i.e., the two dimensions are not the length and the width of sardines and tuna fish :)
- We can talk about both the training error and the validation error in classification. Accordingly, one can build two confusion matrices: one for the training set and the other for the validation set. Here, you are asked to build the latter: the one for the validation set only.
- We do not consider the instances in the validation set “neighbors” to each-other as implied by the name of the algorithm. The training instances are the neighbors (of the validation instance) to look at, so to speak.