CS 422 Homework Assignment 3

Due: 11:59pm, Thursday, March 29th

This assignment is scored out of 70. It consists of 6 questions. When you submit, you are required to create a folder with your name (Last name first, then First name), CS422, HW3, e.g., LastName_FirstName_CS422_HW3. Type your answers into a text file (**only `.txt`, `.doc`, and `.pdf` file formats are accepted**) and save it in this folder. Put all your Java programs (`*.java`) as well as output files in the same folder. Zip this folder, and submit it as one file to Desire2Learn. Do not hand in any printouts. Triple check your assignment before you submit. **If you submit multiple times, only your latest version will be graded and its timestamp will be used to determine whether a late penalty should be applied.**

Short Answers

P1. (10pts) In the following data set, the attribute $status$ is the class label. Use the Bayesian classifier to predict the class for the tuple $X = (department = system,\ age = 31 \ldots 40,\ salary \leq 40K)$. Show all your work.

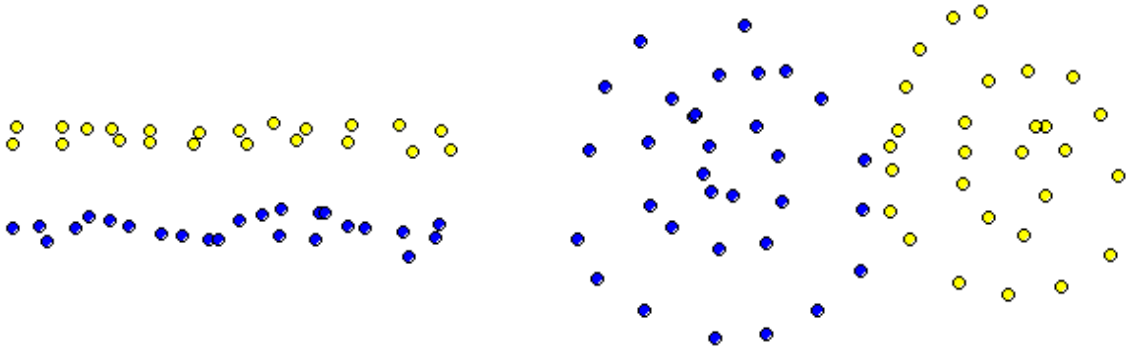
department	*age*	*salary*	*status*
sales	31...40	41K...50K	senior
sales	<=30	<=40K	junior
sales	31...40	<=40K	junior
systems	<=30	41K...50K	junior
marketing	31...40	>50K	senior
systems	<=30	41K...50K	junior
systems	>40	>50K	senior
marketing	31...40	41K...50K	senior
marketing	31...40	41K...50K	junior
systems	>40	<=40K	senior
sales	<=30	<=40K	junior

P2. (5pts) A report shows that 13% of the undergraduate students smoke while this number is 25% for the graduate students. If in a university, 75% of the students are undergraduate students and the rest are graduate students, what is the probability that a student who smokes is an undergraduate student? Show all your work.

P3. (6pts) Explain why an SVM algorithm has high classification accuracy in high-dimensional space? Also can you show that an SVM is a special case of a rule-based classifier?

P4. (5pts) Give an example of a data set consisting of three natural clusters, for which k-means would almost always find the correct clusters, but bisecting k-means would not.

P5. (6pts) For each of the following two examples, which clustering strategy would you think can produce two nicely separated clusters: k-means, hierarchical clustering using minimum distance, hierarchical clustering using maximum distance, or hierarchical clustering using mean distance. Explain your answers.



Example 1 Example 2

Programming Questions

P5. (38pts) A high-level summary of the k-means clustering algorithm is shown as follows:

---

k-means clustering algorithm

---

1. Select k samples as the initial centroids.
2. **repeat**
3. Form k clusters by assigning all samples to their closest centroid.
4. Recompute the centroid of each cluster.
5. **until** the SSE does not reduce**.**

In this algorithm, k samples are selected as the initial cluster centroids. For each sample in the data set, the Euclidean distances between the sample and k centroids are calculated. The sample is then assigned to the cluster with the shortest distance. Note that each sample belongs to only one cluster.

After all samples are assigned to their clusters, the sum of squared error (SSE) is measured on the current clustering. The SSE is defined as follows:

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2$$

where SSE is the sum of the squared error for all samples in the data set; p is a data sample in cluster C_i, whose centroid is c_i (both p and c_i are multi-dimensional).

The algorithm will then start a loop to update the centroids on the current clustering, re-assign the samples to their corresponding clusters, and measure the SSE. The loop is terminated if the SSE does not reduce anymore.

In this programming question, you will implement the above k-means clustering algorithm and test your program on the UCI Iris data set. The detailed information for this data set is available at https://archive.ics.uci.edu/ml/datasets/iris. You are recommended to read the data description before you get started.

The Iris data set contains 150 iris samples. Each sample has 5 attributes:

Sepal length	Sepal width	Petal length	Petal width	Class

Attributes 1~4 describe the properties of the sample the last attribute is the class label. Because this data set is used for cluster analysis, all class labels were removed. The preprocessed data set is stored in the text file "**`iris.txt`**". The k-means algorithm runs on this data set to minimize the clustering SSE.

a. Completing the `Homework3` class

You are provided with two files "**`Homework3.java`**" and "**`TestHomework3.java`**". You are required complete the following four methods in the former:

(Note: you should NOT change the contents of the parameter arrays in any of the following methods!)

`int[] assignSamples(int k, double[][] centroids, double[][] dataset)`
This method takes as parameter an integer for the number of clusters, a two-dimensional **`double`** array for the cluster centroids, and a two-dimensional **`double`** array for the data set that contains test samples. It returns a one-dimensional **`int`** array that contains the cluster indices for all samples.
The array **`centroids`** has **`k`** rows and 4 columns. Here is an example for a 2-cluster problem:

```
{5.2, 4.1, 1.5, 0.1}
{4.5, 1.8, 3.2, 1.1}
```

You can call the **`euclidean`** method that is provided to you to obtain the **Euclidean distance** between the test sample and every centroid in order to find the closest cluster. The sample is then assigned to this cluster and the assignment is recorded in the returned array. The following is an example for the returned array (each slot indicates the cluster index for the corresponding sample):

```
{0, 1, 1, ... , 0}
```

In this array, sample 0 (the samples in the provided data set are indexed from 0 to 149) is assigned to cluster 0, samples 1 and 2 are assigned to cluster 1, whereas the last sample is assigned to cluster 0.

`double[][] updateCentroids(int k, int[] clusters, double[][] dataset)`
This method takes as parameter an integer for the number of clusters, a one-dimensional **`int`** array that contains the cluster indices for all samples, and a two-dimensional **`double`** array for the data set that contains test samples. It returns a two-dimensional **`double`** array that contains the updated cluster centroids with the same format as above. **Each updated cluster centroid is the mean of all samples in that cluster.**

`double calculateSSE(int[] clusters, double[][] centroids, double[][] dataset)`

This method takes as parameter a one-dimensional **`int`** array that contains the cluster indices for all samples, a two-dimensional **`double`** array for the cluster centroids, and a two-dimensional **`double`** array for the data set that contains test samples. It returns the SSE for the given clusters.

`double kMeans(int k, double[][] centroids, double[][] dataset)`

This method implements the k-means clustering algorithm. It takes an integer for the number of clusters, a two-dimensional **`double`** array for the cluster centroids, and a two-dimensional **`double`** array for the data set that contains test samples. It returns the final SSE after the clustering process converges, i.e., the SSE stops reducing. **This method should leverage the above methods in order to implement the algorithm. It should NOT re-implement any functionalities that have been implemented in the above methods.**

Note that you are only supposed to touch the above methods. You are NOT allowed to create any other methods, instance variables, or make any changes to methods other than the above methods or files other than "`Homework3.java`". Points will be taken off if you fail to follow this rule.

b. Code Testing

You are provided with a test driver implemented by "**`TestHomework3.java`**" (**Do not make any changes to this file!**) so there is no need to write your own. You are also given a text file "**`iris.txt`**" that contains the test samples.

Depending on your programming environment, the data file might need to be placed in different folders so that you test driver can read it. For jGRASP, you can leave the data file in the same folder as your java files. For NetBeans, you should place it in your project folder in which you see directories like **`build`**, **`nbproject`**, and **`src`**, etc.

Once you have completed the required methods, you can run the test. You should create a plain text file named "**`output.txt`**", copy and paste the output (if your code crashes or does not compile, copy and paste the error messages) to this file and save it.

Grading Rubric:

Code does not compile: -10
Code compiles but crashes when executed: -5
Changes were made to things other than the required methods: -5
Has output file: 5
`assignSamples` changes the content of the array parameters: -5
`assignSamples` does not use Euclidean distance: -5
`updateCentroids` changes the content of the array parameters: -5
`calculateSSE` changes the content of the array parameters: -5
`kMeans` changes the content of the array parameters: -5
`kMeans` does not leverage the other three methods: -5
Code passes 11 test cases: 33 (each test case worth 3 points)

Sample Output:

```
Test 1: assignSamples - [Passed]
   Centroids:
   5.70        4.40        1.50        0.40
   5.50        2.40        3.80        1.10

   Test records:
   4.50        1.80        3.20        1.10
   5.20        4.10        1.50        0.10
 --------------
 Expected:[1, 0]
 Yours: [1, 0]

Test 2: assignSamples - [Passed]
   Centroids:
   5.70        4.40        1.50        0.40
   5.50        2.40        3.80        1.10

   Test records:
   5.80        2.00        3.00        0.20
   5.20        4.10        1.50        0.10
   5.90        4.20        1.20        1.50
   5.40        3.00        4.50        1.50
 --------------
 Expected:[1, 0, 0, 1]
 Yours: [1, 0, 0, 1]

Test 3: assignSamples - [Passed]
   Centroids:
   5.00        3.40        1.60        0.40
   6.00        2.70        5.10        1.60
   6.50        3.00        5.50        1.80

   Test records:
   5.00        3.70        1.00        1.80
   5.90        3.20        4.80        1.80
   6.10        3.00        4.60        1.40
   6.70        3.10        5.60        2.40
 --------------
 Expected:[0, 1, 1, 2]
 Yours: [0, 1, 1, 2]

...

Test 11: kMeans(Initial centroid indicies = [20, 90, 140]) - [Passed]
 --------------
 Expected: 78.945
 Yours: 78.945

Total test cases: 11
Correct: 11
Wrong: 0
```