CS 540-3: Homework 3

Clustering

100 pts total

Release date: February 29, 2016
Due date: March 11, 2016 at 11:59PM

Directions: Please submit *one* PDF file and KMeans.java file to Moodle by the deadline specified. Failure to convert your solutions to a single PDF file will result in the deduction of points. Your code also **MUST** be executable on the CS Linux machines with the command specified. Failure to observe this requirement will *result in a 0 for the programming portion of the assignment*!!! Please be sure you understand the course policies on plagiarism before consulting with other students. See either the TA or instructor if you have any questions.

1 Hierarchical Clustering [20 pts]

Consider the following set of singleton clusters:
$\{\{6\}, \{8\}, \{10\}, \{20\}, \{26\}, \{30\}, \{32\}, \{33\}\}$
Recall that in performing Hierarchical Agglomerative Clustering (HAC), there are three linkages that are typically used in calculating inter-cluster distance: single linkage, complete linkage, and average linkage. Assume that in merging clusters ties are broken by merging the two clusters that result in the smallest sum for the points in the cluster. So, for instance, if we have the three clusters {1}, {2}, and {3}, the first two would be merged first since they result in a sum of 3, whereas merging the last two would result in a sum of 5.

a) Starting with the eight singleton clusters in the dataset above, perform HAC until you are left with one cluster. For full credit, show each step in the clustering process. Do this with:

   (i) Single linkage

   (ii) Complete linkage

   (iii) Average linkage

b) For your clustering using single linkage, also draw the resulting binary tree.

c) Suppose we wanted to know what the best number of final clusters would be. Briefly describe one way you might determine this. *Hint:* Consider how you would first determine how "good" each clustering is.

2 k Nearest Neighbor [10 pts]

Consider the data in Table 1, where the first three columns represent features, and the last column represents a classification label.

Weight (lbs)	Height (in)	Shoe Size	*Age*
90	52	7	10
130	69	9.5	20
50	45	6	10
63	51.5	6.5	10
145	70	11	20
160	69.5	10	20

Table 1: **Weight, Height, Shoe Size, Age**

a) Using classification-based k-NN with $k = 3$, predict the label for the instances [100, 50, 7, ?] and [120, 90, 9, ?]. Use Euclidean distance for a distance metric. In the event of a tie (i.e., two neighbors are equidistant from the instance in question and only one can be used), use the neighbor that appears first in the table. Show all work for credit.

b) Repeat part (a), but use regression-based k-NN.

3 Implementation of k Means [70 pts]

In this question, you will be building a k-means clustering algorithm with Euclidean distance. We are providing a skeleton code "KMeans.zip" for you at the course Web page. Please download, unzip this file, and open the file `KMeans.java`. You will see the code in Fig 1.

```
public class KMeans {
    public KMeansResult cluster(
        double[][] centroids,
        double[][] instances,
        double threshold) {
    /* ... YOUR CODE GOES HERE ... */
    }
}
```

Figure 1: **KMeans.java**

Your contribution is the implementation of this `cluster` method. It accepts an array of initial centroids, an array of instances, and a threshold for stopping the iterations of the algorithm. It must return an object of type `KMeansResult`, as shown in Fig 2. It is defined in `KMeansResult.java`. We will use this object to verify the correctness of your clustering algorithm.

```
public class KMeansResult {
    double[][] centroids;
    double[] distortionIterations;
    int[] clusterAssignment;
}
```

Figure 2: **KMeansResult.java**

The `centroids` is an array of the final coordinate of your centroids. The `distortionIterations` will return an array of each successive distortion your algorithm records on each iteration, where distortion is as we discussed in class (see the lecture slides). This allows us to know both how many iterations your algorithm required and how well it clustered the data set before converging. When the difference between successive distortions, recorded on each iteration, is less than the `threshold` argument, your program should stop iterating and return an instance of `KMeansResult` that provides us with the appropriate arrays.

3.1 More about `KMeans.cluster()`

Since `KMeans.cluster()` is the core part of this question, we will give a more detailed explanation of its prototype.

3.1.1 Parameters

You are required to pass three arguments to the method `cluster`. Their detailed meanings are given as follows:

- `double[][] centroids`
  A two-dimensional array of the initial position of the centroids. Each row corresponds to a centroid and each column to a feature dimension.
- `double[][] instances`
  The data set you will be clustering. Each row corresponds to an instance, and this array will have the same "width as the centroids array.
- `double threshold`
  The threshold used to determine when to stop iterations. More specially, if at iteration i the relative change in distortion between successive iterations is less than the `threshold`, i.e., $|\frac{distortion(i)-distortion(i-1)}{distortion(i-1)}| < threshold$, then your algorithm should terminate and return its results.

3.1.2 Results

You are required to return a variable with the type `KMeansResult`. The detailed meanings of its each field are given as follows:

- `double[][] centroids`
  The position of the final centroids after your clustering is complete.
- `double[] distortionIterations`
  An array of the distortions your algorithm records on successive iterations.
- `int[] clusterAssignment`
  An array of the index of the centroid assigned to each row of the instances method argument. That is, the integers in this array will point

to a row in the returned `double[][] centroids array`. So, for example, if we wished to know the coordinate, in the feature space, of the centroid assigned to the first instance, we would call `centroids[clusterAssignment[0]]`. Remember that all Java arrays are 0-based.

3.1.3 Implementation details

In each iteration, you should first reallocate the clusters for each instance, then update the coordinates of centroids by averaging all instances in their corresponding clusters. It is possible that, in some iteration after reallocation, there is a centroid c which does not match to any instances. In this case, we call the centroid of this cluster an *orphaned centroid*. This is a problem, because you would have no instances assigned to the centroid to average over to determine the centroids new location. To solve this, we specify you to implement the following behaviors in this scenario:

1. Search among all the instances for the instance `x` whose distance is farthest from its assigned centroid.
2. Choose `x`'s position as the position of `c`, the orphaned centroid.
3. Reallocate again the cluster assignments for all `x`.
4. Check if there is still a orphaned centroid. If so repeat step 1 to 4 until all orphaned centroids have been removed.

Once you have removed all possible orphaned centroids, you should update the centroids coordinates by averaging over all the instances assigned to it. After that, you are also required to calculate the distortion of an iteration, and store it into the array `distortionIterations`. Notice that we dont know the number of iterations before hand, so you may need to use a variable-length array like `Vector` or `ArrayList` first, and write the result into `distortionIterations` after all iterations.

Moreover, it is also possible that the minimum or maximum distance may correspond to multiple candidates. For example, there may be more than one cluster centroid with the same minimum distance to an instance, or several instances with the same maximum distance to their corresponding centroids. In this case, always choose the one with smaller ID, either instance ID, cluster ID, etc.

Finally, your program should be able to deal with different numbers of centroids, instances, as well as the feature dimensions, and work correctly no matter what the size of the `double[][] centroids` and `double[][] instances` arrays. So do not hard-code any array lengths. The `.length` member allows you to determine the length of the array. You can determine the number of instances with `instances.length` and the number of feature dimensions with `instances[i].length`, where `i` is an integer array index.

3.2 How to test

We will test your program on multiple test cases. Your code should be able to handle test set of size 10,000, although we may also test it with smaller test sets. We will also vary centroid initialization, number of clusters, as well as threshold value, to test your programs correctness comprehensively.

The format of test command will like this:

```
java HW3 data_file init_centroid_file threshold output_flag,
```

where `data_file` contains the features of all instances, `init_centroid_file` gives the initial positions of a certain number of centroids, and `threshold` has same meaning as mentioned. The `output_flag` is an integer argument, controlling what the program outputs:

1. 1 - The array of centroids
2. 2 - The array of assignments
3. 3 - The array of distortions

In order to facilitate your debugging, we will provide you a sample input and output files. They are `words0.txt` and `initCentroids0.txt` for the input, and `output0_1.txt`, `output0_2.txt`, `output0_3.txt` for the output, as seen in the zip file. Here is an example command:

```
java HW3 words0.txt initCentroids0.txt 1e-4 3
```

You are **NOT** responsible for any file input or console output. We have written the class `HW3` for you, which will load the data and pass it to the method you are implementing.

As part of our testing process, we will call `javac *.java` and then call the main method of `HW3` with parameters of our choosing.

3.3 Deliverables

1. Hand in (to Moodle) your modified version of the `KMeans.java` code skeleton with your implementation of the k-means clustering algorithm.

2. Hand in (also to Moodle) *one PDF* of your written solutions and (optionally) any comments about your k-means program.