Assignment 4: Hierarchical Clustering

Due: 11/8 2017 11:59 PM PST

Assignment Overview

In this assignment, you will implement the hierarchical clustering algorithm. You will use it to cluster a data set. For this assignment, we will use a data set from the UC Irvine Machine Learning Repository at: https://archive.ics.uci.edu/ml/index.html.

Bonus: this assignment can be done with either Python or Scala. There will be extra 10% bonus for Scala implementation.

Write your own code!

For this assignment to be an effective learning experience, you must write your own code! **Do no share code with other students in the class!!**

Here's why:

- The most obvious reason is that it will be a huge temptation to cheat: if you include code written by anyone else in your solution to the assignment, you will be cheating. As mentioned in the syllabus, this is a very serious offense, and may lead to you failing the class.
- However, even if you do not directly include any code you look at in your solution, it surely will influence your coding. Put another way, it will short-circuit the process of you figuring out how to solve the problem, and will thus decrease how much you learn.

So, just don't look on the web for any code relevant to this problem. Don't do it.

Format of data file

The data file that you are clustering is a database related to iris plants. A complete description can be found here: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names

You will use the file at https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data as your input file.

Each line of the csv file looks something like this: 5.1, 3.5, 1.4, 0.2, Iris-setosa

It consists of four floating point values and a text label for the type of iris plant.

The four floating point attributes correspond to:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

The string attribute is the iris class, one of the following:

-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Program Description

Your program should do the following:
1. Read the data from the file. Use only the floating-point values for the clustering. Don't discard the class information. While you can't use it for clustering, you will need it later for assigning names to the clusters and for checking the accuracy of the clusters.
2. **Apply the hierarchical algorithm to find clusters. Use Euclidean distance** as your distance measure.
3. **Assign each final cluster a name** by choosing the most frequently occurring class label of the examples in the cluster.
4. **Count the number of data points that were put in each cluster.**
5. **Find the number of data points that were put in clusters in which they didn't belong** (based on having a different class label than the cluster name).

Hierarchical algorithm:

In this assignment, you will implement Hierarchical Clustering, which should start from merging the first two closest points, then the next closest, etc. Euclidean distance is used as the distance metric, and the coordinate of centroid is defined as the average of that of all the points in the cluster.

As discussed in the lecture, **Priority Queue** can be used to build the full hierarchy. To initial a priority queue, you are allowed to use the **heapq** library in Python and **scala.collection.mutable.PriorityQueue** library in Scala. This is the only permissible external library, with the exception of math library.

Output of your program

The program will produce output of the form:

cluster <clustername1>:
(List of points in that cluster, one per line)
Number of points in this cluster: (number of points)

cluster <clustername2>:
(List of points in that cluster, one per line)
Number of points in this cluster: (number of points)

cluster <clustername3>:
(List of points in that cluster, one per line)
Number of points in this cluster: (number of points)

Number of points assigned to wrong cluster: (number of points)

Running your code

python lastname_firstname_clustering.py inputFile k
or
./bin/spark-submit --class class_name --master local[*] your_jar_file.jar inputFile k

where:
inputFile is a string indicating the name of the data file to be clustered

k is an integer representing the number of clusters

Submission Details

What you will turn in:

`lastname_firstname_clustering.py`

or

`lastname_firstname_clustering.scala, lastname_firstname_clustering.jar`

Testing your code

The sample command to execute is:
python lastname_firstname_clustering.py iris.data k
or
./bin/spark-submit --class class_name --master local[*] your_jar_file.jar inputFile k
Notice: we may use different k to test your code, so you should build the full hierarchy that works with different k values.

The sample output format is provided below:

```
cluster:Iris-versicolor
[7.0, 3.2, 4.7, 1.4, 'Iris-versicolor']
[6.4, 3.2, 4.5, 1.5, 'Iris-versicolor']
[6.9, 3.1, 4.9, 1.5, 'Iris-versicolor']
[6.5, 2.8, 4.6, 1.5, 'Iris-versicolor']
[6.3, 3.3, 4.7, 1.6, 'Iris-versicolor']
[6.6, 2.9, 4.6, 1.3, 'Iris-versicolor']
[6.1, 2.9, 4.7, 1.4, 'Iris-versicolor']
[6.7, 3.1, 4.4, 1.4, 'Iris-versicolor']
[6.1, 2.8, 4.0, 1.3, 'Iris-versicolor']
[6.1, 2.8, 4.7, 1.2, 'Iris-versicolor']
[6.4, 2.9, 4.3, 1.3, 'Iris-versicolor']
[6.6, 3.0, 4.4, 1.4, 'Iris-versicolor']
[6.8, 2.8, 4.8, 1.4, 'Iris-versicolor']
[6.7, 3.0, 5.0, 1.7, 'Iris-versicolor']
[6.0, 2.9, 4.5, 1.5, 'Iris-versicolor']
[6.0, 3.4, 4.5, 1.6, 'Iris-versicolor']
[6.7, 3.1, 4.7, 1.5, 'Iris-versicolor']
[6.1, 3.0, 4.6, 1.4, 'Iris-versicolor']
[6.2, 2.9, 4.3, 1.3, 'Iris-versicolor']
Number of points in this cluster:19

cluster:Iris-virginica
[6.1, 2.6, 5.6, 1.4, 'Iris-virginica']
Number of points in this cluster:1

Number of points wrongly assigned:4
```

Grading Criteria

1. If your code cannot be run with the commands you provide, your submission will not be graded.
2. If your program generates more than one file, there will be 20% penalty.

3. There will be 20% penalty for late submission.
4. Format of the output should exactly match the example described in the assignment. There will be 10% deduction if it's not matching.
5. There will be several test cases provided to test your code and verify your result. If your result is not correct, there will be 20% penalty for each test case.