Fall 2015 **CSC 466:** Knowledge Discovery from Data Alexander Dekhtyar

Lab 5: Clustering

Due date: Friday, December 4, 11:59pm..

Overview

In this assignment you will use unsupervised learning techniques to discover clusters in a number of simple datasets. You will implement two cluster analysis methods, **k-means clustering** and **agglomerative hierarchical clustering**.

Assignment Preparation

This is a pair programming assignment. Each student teams up with a partner. While this assignment can be easily split into two parts (one person does k-means clustering algorithm, the other — hierarchical clustering algorithm), I **strongly recommend** that both students take active part in implementing both algorithms.

Note: Open-source versions of clustering algorithms are plentiful. It is considered cheating to re-use code for clustering algorithms that was written by someone else. (You may use code from other *explicitly acknowledged sources* for other purposes in the program, e.g., for parsing input).

Data

We provide multiple **simple datasets** for use in this assignment. Each dataset consists of two files: a CSV (comma-separated values) file containing the actual data and a text `.txt` file containing the *header* of the dataset, specifying the names of the attributes.

Most of the datasets were adopted from

http://people.sc.fsu.edu/∼burkardt/datasets/hartigan/hartigan.html which contains a list of datasets from [1] and [2]. Some of these datasets contain real data, some — are synthesized. One more dataset, Iris, is a classical machine learning dataset[1] from the University of California, Irvince Machine Learning repository[2] The remaining data files come from the FATAL ACCIDENTS dataset, collected by Peter Oyler[3]. The dataset contains information about automobile accidents in California that resulted in fatalities. All CSV files for all datasets have exactly the same format.

CSV file format. The first line of each CSV file is analogous to the restrictions file in the **Lab 3** assignment. It is a **binary vector** identifying which columns of the CSV file shall be used for clustering. In some CSV files, one of the columns is used as a `rowId`, and should be excluded from the analysis, while in some other files, all columns contain data. In the Iris dataset, the last column (labeled as 0 by the restrictions vector) is actually a class label, which can be used as ground truth to validate the quality of the clustering.

0 in position i of the first row means that column i should be ignored in the cluster analysis.

1 in position 1 of the first row means that column i should be used in the cluster analysis.

The remaining rows of the CSV file contain data points.

Text files. Text files need not be processed by your programs. Text files contain the original headers for each dataset, split from the CSV data to simplify parsing. The headers describe the nature of the dataset, contain references to its origins and specify the names of each column in the dataset. If you want to use attribute headers in your program (e.g., for generating output), you may either add a header file name as a parameter to your program, or deduce the header file name (it will always be `header_` followed by the name of the data file w/o extension, followed by `.txt`).

List of Datasets

The following datasets are *officially* provided to you for this assignment. Each file listed below, as well as a `zip` archive of the entire collection is available at the following URL[4]:

http://wiki.csc.calpoly.edu/csc466-2009/wiki/Lab4Data

---

[1] This is the most often cited by machine learning papers dataset in existence.

[2] http://archive.ics.uci.edu/ml/index.html

[3] Peter took the first version of CSC 466. The FATAL ACCIDENTS dataset was his data analytical project.

[4] I used an already existing page from the previous course's wiki and added data to it. You have no write permissions to the wiki, but should have full read rights.

CSV File	Header File	Dataset	Source
`4clusters.csv`	`header_4clusters.txt`	4 clusters in 2D space (synthetic)	Spaeth[2]
`mammal_milk.csv`	`header_mammal_milk.txt`	constituents of mammal milk	Hartigan[1]
`economy.csv`	`header_economy.txt`	profit vs. equity in sectors of economy	Hartigan[1]
`planets.csv`	`header_planets.txt`	sightings of minor planets	Hartigan[1]
`iris.csv`	`header_iris.txt`	measurements of different Iris flowers	UCI ML repository
`many_clusters.csv`	`many_clusters_headers.txt`	many clusters in 2D space (synthetic)	Spaeth[2]
`birth_death_rate.csv`	`header_birth_death_rate.txt`	birth and death rates of countries	Hartigan[1]
`AccidentsSet01.csv`	`header_AccidentsSet01.csv`	fatal automotive accidents	P. Oyler
`AccidentsSet02.csv`	`header_AccidentsSet02.csv`	fatal automotive accidents	P. Oyler
`AccidentsSet03.csv`	`header_AccidentsSet03.csv`	fatal automotive accidents	P. Oyler

More datasets may be made available by the instructor, as well as can be manufactured and used by each team, but only the datasets listed above will be used as part of your lab deliverables.

Lab Assignment

You will investigate the behavior of two common clustering procedures, ***k*-means clustering** and **agglomerative hierarchical clustering** on the datasets provided to you. Part of your assignment involves implementation of the two clustering procedures. In addition to this, you will study the datasets using your implementations of the clustering algorithms, and will submit the results you have obtained.

For simplicity, the lab refers to Java programs whenever specific pieces of code are mentioned. **However, each team is free to implement clustering algorithms in their programming language of choice** *(provided that the instructions on how to run their programs are submitted as part of the lab deliverables).*

k-means Clustering

You will implement the ***k*-means clustering** algorithm as a Java program `kmeans.java`.

Input. The program shall take as input two parameters:

```
java kmeans <Filename> <k>
```

`<Filename>` is the name of the CSV file containing the input dataset.

`<k>` is the number of clusters the program has to produce.

Internals. There is a variety of versions of the ***k*-means clustering algorithm**. Each team is encouraged to (a) select the specific version and, possibly (b) experiment with different versions [5] to discover the one that produces better clusters (or works faster).

While the datasets you are working with are small, you should implement a version of the **disk-based k-means clustering algorithm** in a sense that your implementation **access each data point exactly once per iteration of the algorithm**. You **are allowed** to store all data in main memory, since the datasets are very small.

Among the different decisions that each team can make are:

- Choice of **initial cluster centroids**. Random, SelectCentroids procedure, pre-selected, other variations. . . .
- Choice of **centroid recomputation method**. Typically, the **mean** point of the cluster (hence the "**k-means**" in the name of the method), but you can use **medians** or **modes** if so desired.
- **Stopping criterion.** Minimum point reassignment, minimum centroid change, threshold on sum of squared error (SSE) (for the latter — you need to find a good threshold).
- **Distance measure.** While Eucledean distance makes sense for some of the datasets (the 2D synthetic ones, e.g.), you may choose to try other distance measures to see if your cluster organization changes/becomes more pronounced.
- **Dealing with outliers.** Attempt to detect outliers vs. no outlier detection.

Output. The output of your program shall consist of two things:

1. Description of the detected clusters.
2. Evaluation of the computed clustering of the data.

Since our datasets are small, describe detected clusters **by listing all points belonging to them** (this also simplifies grading/comparison). If data points come with `rowIds`, you are allowed to output just them.

Because except for the Iris dataset we lack well-established (and suppliable to your program) **ground truth** on clusters, only **internal evaluation measures** for clusters need to be computed and reported. (For the Iris dataset, you can compute the accuracy measures for each cluster if there is a clear cluster-to-class mapping.)

For each cluster, compute and report:

1. Number of points in the cluster.

---

[5] This is where two people can work in parallel on the same program.

2. Coordinates of its centroid.
3. Maximum, minimum, and the average distance from a point to cluster centroid.
4. Sum of Squared Errors (SSE) for the points in the cluster.

You may *additionally* choose to report any other internal measures, e.g., **inter-cluster distances**.

Here is an example of an output for a single cluster (w/o the SSE):

```
Cluster 0:
Center: 26.25,28.0,
Max Dist. to Center: 10.077822185373186
Min Dist. to Center: 3.010398644698074
Avg Dist. to Center: 6.569312544990502
4 Points:
25.0,38.0,
32.0,27.0,
26.0,25.0,
22.0,22.0,
```

(Your output does not have to match this, but should report the information in an easy-to-read way).

Hierarchical Clustering

You will implement the **agglomerative hierarchical clustering** method as a `hclustering.java` program.

Input. Your program shall take two parameters as input:

```
java hclustering <Filename> [<threshold>]
```

`<Filename>` is the name of the CSV file containing the input dataset.

`<threshold>` is the *optional* threshold at which your program will "cut" the cluster hierarchy to report the clusters.

If `<threshold>` parameter is **specified** in the input, your program shall produce both the cluster hierarchy, and the appropriate list of clusters cut at the specified threshold.

If `<threshold>` parameter is **not specified** in the input, your program shall produce the cluster hierarchy **alone**.

Internals. Commonly, **hierarchical clustering algorithms** take as input the distance matrix for the points in the input dataset. You algorithm will work with the same data as the **k-means clustering algorithm**, i.e., the actual dataset. Your implementation of the hierarchical clustering method will be responsible for computing the distances between all points in the dataset.

Additionally, as with k**-means clustering**, in your implementation of the **agglomerative hierarchical clustering** algorithm, you can select the specific details of implementation that you prefer, or experiment with multiple options.

Among the features you have to select are:

- **Distance measure for points.** See k**-means clustering algorithm** discussion.
- **Distance measure for clusters.** Select between **single-link**, **complete-link** and **average-link** methods (you can also try **centroid** or **Ward's** methods).

Output. `hclustering.java` shall produce as output the following information:

- The **dendrogram of the cluster hierarchy** constructed by the algorithm. This hierarchy is independent of the threshold and shall be produced each time the program is run. It is useful to output the dendrogram into a separate file.
- The **actual clusters** obtained by applying the input threshold to the computed **cluster hierarchy**.
- The **evaluation** measures specifying the quality of the obtained clusters.

The latter two types of output should be produced **only if** the threshold value is specified in the input. If it is not specified, only the tree is produced and returned.

Dendrogram. Create an XML version of the cluster hierarchy dendrogram and output it. You need to output the dendrogram XML to `stdout`, but you may, if you want, also create an XML file to store it.

The XML representing the dendrogram is simple and straightforward. The dendrogram is a binary labeled tree. You will use three XML elements:

- `<tree>` to indicate the root node of the dendrogram.
- `<node>` for all inner nodes.
- `<leaf>` for all leaf nodes containing individual data points. Leaf elements are `EMPTY`.

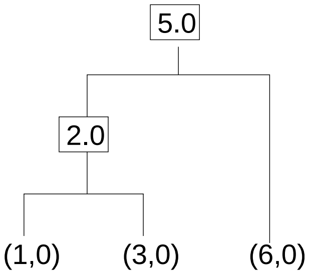
The `<tree>` and `<node>` elements will contain an attribute `height` representing the height label of the node (the distance between the two clusters that are merged).

The `<leaf>` elements will contain an attribute `data` whose contents shall be the data point from the dataset represented by the leaf node (or its `rowId`). For the sake of completeness, feel free to add a `height` attribute with the value of 0 to the `<leaf>` elements.

The `<tree>` and all `<node>` elements shall have exactly two children: either `<node>` or `<leaf>` elements.

Here is an example of a small XML output and the dendrogram it represents.

```
<tree height = "5.0" >
   <node height = "2.0">
      <leaf height= "0" data ="1, 0"/>
      <leaf height= "0" data ="3, 0"/>
   </node>
   <leaf height = "0" data = "6,0"/>
</tree>
```



Clusters and measures. Whenever a threshold is specified, your program shall, in addition to producing the full dendrogram, also report the clusters that are formed when the dendrogram is cut at the specified threshold. The clusters and their parameters shall be reported in the same way as your **k-means clustering algorithm** implementation does it.

Study of the datasets

You will use your implementations of the **k-means clustering** and **agglomerative hierarchical clustering** methods to study the six datasets provided to you and to find the *best* (in your opinion) clusters in the data.

The results of your study will be compiled in the report that each team will prepare and submit.

Use both methods. For each dataset provided to you, you need to use both methods (programs) to discover the best clusters. Your report will indicate the best results you achieved using each of the two programs.

Best clustering. Each implementation comes with one *predefined* parameter (k, number of clusters, for k-means clustering; threshold for hierarchical clustering), which will affect the output of the respective program. In addition, each team may choose to implement multiple choices for various parts of each algorithm (e.g., you may implement all three *linking* techniques for the hierarchical clustering, or a number of different ways to select the initial centroids for the k-means clustering).

You will run your programs on each dataset with different settings of the input parameters (and any choices that you have implemented) to determine which runs produce the best clustering result.

The evaluation measures computed by your programs will help you assess the quality of the output.

Additionally, you may choose to determine the **ground truth** for some (or all) of the datasets and compare the outputs to the results you obtain. Ground truth for the Iris dataset is given in the dataset itsel. For some other datasets it can be determined by investigating/visualizing the dataset using any tools available to you. 2D and some 3D datasets, for example, can be plotted as scatterplots using Excel or MatLab, and visual observation may help you determine the clusters that should be detected by your program. (For some datasets, like the `economy` dataset, which has 10 variables, simple visualization may be infeasible).

Report. Each team will prepare a report of discoveries. The report will consist of the following information:

1. Study Design. Provide brief description of your implementations of the two clustering methods. Specify which features (such as distance measures, centroid chocies, etc.) you have implemented and used in the study.
2. Results. For each dataset, and for each algorithm provide the description of the best result achieved. Specify the input parameters (and features) of the program that lead to the best result, and provide a snapshot of the obtained output (feel free to paste the output verbatim, or to use a more succinct representation of the clusters.
3. Discussion. May be embedded into reporting of results. Mention any interesting observations about specific datasets/methods that you have made during the study.
4. Analysis. Present an overall conclusion of your results. Did both methods perform equally well? Did one method worked better on some data and the other - on different data? When did each of the algorithms perform well? What features (if any) of each algorithm lead to better performance in what cases?

Deliverables and submission instructions

This lab has both electronic and harcopy deliverables.

Electronic deliverables.

- `kmeans.java` and all supporting files.

- `hclustering.java` and all supporting files.
- `README` file specifying what has been implemented, and providing any instructions for compiling and running the code. (must contain the names of all students on the team).
- **The report.** The electronic version of the report should be submitted in PDF format.

Use the following command

```
$ handin dekhtyar lab05-466 <Files>
```

Extra Credit

Extra credit (up to 50% of the full score for the lab) will be given to any team that implements DBSCAN and incorporates the results of running DBSCAN on the given datasets in the report. Please note, that due to fairly small quantities of data, DBSCAN may not yield significantly higher quality performance than the other two methods.

As you remember, DBSCAN is guided by two parameters: ε: the radius of the neighborhood of a data point, and *minPts*: the smallest number of points in a neighborhood of a data point, for it to be considered a core point. Your analysis of work of DBSCAN shall discover the best combination/combinations of these parameters for each dataset.

Otherwise, your implementation of the DBSCAN algorithm shall behave similarly to your implementation of the k-means clustering algorithm. The inputs are the name of the data file and the values of the two parameters, while the output is the list of clusters and the analysis of each cluster.

References

[1] John Hartigan, *Clustering Algorithms*, Wiley, 1975. ISBN 0-471-35645-X.

[2] Helmut Spaeth, *Cluster Dissection and Analysis, Theory, FORTRAN Programs, Examples*, Ellis Horwood, 1985.