Data Mining Project
KMeans ++

Vicențiu-Marian Ciorbaru
Master Artificial Intelligence

1. Algorithm Significance

KMeans++ represents an improvement over the KMeans algorithm. Before describing the KMeans++ improvement, we must first discuss about what the KMeans algorithm computes and more importantly how.

1.1 Definition

KMeans is an algorithm used frequently in data mining to detect groups of data points that conglomerate within a section of the data space. The aim of finding such groups (called clusters) is to try and extract a possible set of common traits, in order to detect various patterns within data.

A more formal definition of the k-means problem is to find cluster centers that minimize the intra-class variance, which means to find the objective function:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

Where:

S_i = i'th cluster

μ_i = centroid for cluster S_i

KMeans algorithm minimizes the objective function in an iterative process. The iterative process is very similar to a hill climbing algorithm. After each iteration the algorithm gets closer to reaching the local optimum point. The “moving agents” within the hill climbing algorithm are actually the centroids for each cluster. The approach is the following:

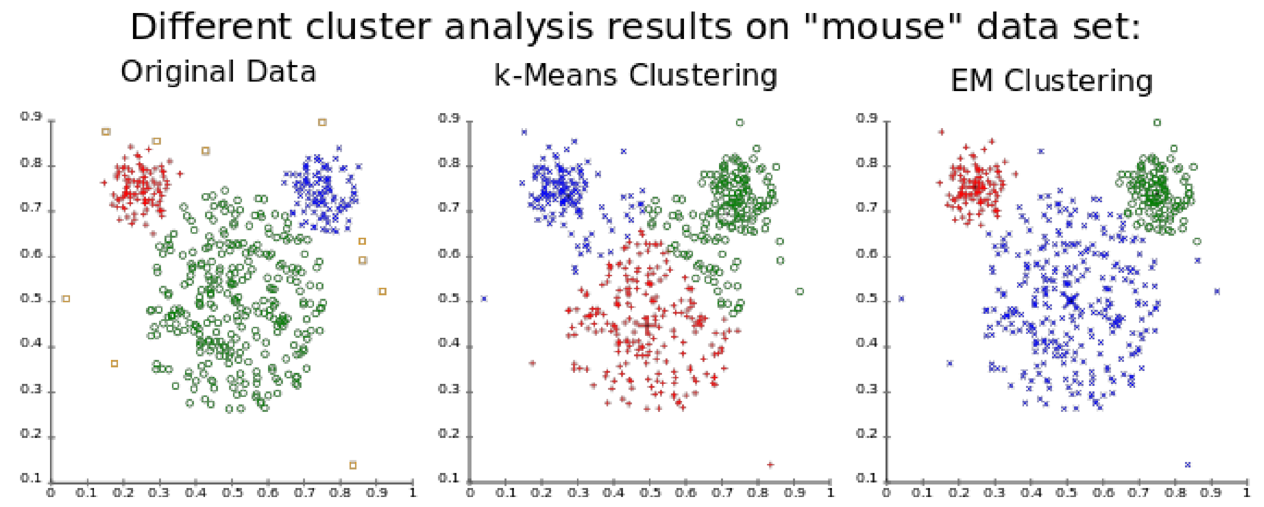
- Initialy, the centroids are chosen at random, and points are assigned to the closest centroid.

Then, the algorithm iterates using the following logic:

1. Centroids are calculated for the current cluster assignation.
2. Each data point is assigned to the cluster whose centroid is closest.
3. Repeat from step 2 till there are no more changes to the assignations.

1.2 Known results and issues

- The approach described in the last paragraph usually finishes in a timely manner, but the problem itself is NP-hard. For certain inputs the algorithm can take a very long time to converge to a solution.
- Another potential problem is that the cluster sizes tend to be equal, due to the way the objective function is defined. This causes problems with clusters that have various spread.
- Still, one of the most important issues with kMeans is that its results are very dependant on the starting centroids. With a bad choice of initial centroids, the algorithm’s performance degrades, both in terms of final results, as well as the time it takes for it to converge. This is the main drawback that KMeans++ aims to solve.



k-Means equals size cluster drawback on artificial data points.

- KMeans only works with continuous data. In case of non-continuous data, different preprocessing techniques must be used, in order to bring the data to a compatible format. For example, enumerations need to be attached a number for each item within the enumeration. This is not always possible, as there are might be no distance link between adjacent items.
- By increasing the number of attributes used to clusterize items, the significance of each attribute is diminished. This effect is known as the curse of dimensionality [1]. In essence, as the number of attributes reaches infinity, the distance between each data point becomes equal. This is a problem with all clustering algorithms that aim to use the distance between the elements as the means to classify elements.

1.3 Practical Applications

KMeans is used in many scenarios, with a various amount of success. Some of its main areas of interest are:

- Vector quantization
- Cluster analysis
- Feature learning

1.3.1 Vector quantization

Vector quantization can be summed up as taking an array of vectors, a set of n groups and finding to which group each vector belongs to. This can be translated as a means of summarizing data. By summarizing the inputs, other algorithms that do not scale as well, can be run afterwards with smaller sized input.

Examples of such uses are:

- Audio codecs
- Image compression algorithms

- Conversion of bitmap images to vector equivalents. [2]

1.3.2 Cluster analysis

There are many applications that can arise from cluster analysis. The kMeans algorithm performs the first step of the analysis, by splitting the input into clusters. Afterwards, the results can be interpreted.

Example interpretations include:

- Market research: by analyzing patterns in surveys and test panels, businesses are able to make decisions on how to attract prospective customers and where to invest additional resources.
- Social media analysis: by analyzing social media friend groups, various conclusions can be drawn. Finding significant correlations between different users aids in marketing campaigns and news feed tailoring. Groups of users can be found that prefer certain information and can afterwards have their online experience personalized with specific content.
- Crime analysis is related to social media analysis. With the use of clustering, public safety can be improved by detecting crime “hot spots” within a certain area. Analyzing temporal and geographical data results in an image of frequent areas with crime issues.
- Image segmentation is another useful result of cluster analysis. By splitting images into separate areas, one can extract the or foreground from an image, or detect the main areas of interest.

2. Algorithm general presentation

KMeans++ is comprised of two algorithms. One is the preprocessing step, that leads to selecting the initial centroids. The second step represents the actual execution of the kMeans algorithm. As the kMeans algorithm has been explained in section 1, only the preprocessing step will be addressed here:

The initial centroids are selected from the pool of data points supplied as input data. The algorithm chooses as many centroids as required and is specified by the number “k” within the input. The algorithm is as follows:

1. Select an initial point at random
2. If the number of points chosen is equal to k, stop the preprocessing algorithm and begin running kMeans with the initial centroids.
3. For each point within the dataset, calculate the distance D to its closest centroid from the list of chosen centroids.

4. Select a new point from the unselected points, with a weighted probability, proportional to D^2. Go to step 2.

The main advantage of using this approach is that the algorithm begins with the initial centroids roughly evenly space apart. Couple this with the fact that kMeans tends to produce evenly sized clusters, due to its objective function, the overall running time of the kMeans++ implementation performs on average twice as better.

The guarantee of the new method of choosing centroids is that the algorithm performs at least as well as the regular kMeans, in both speed and error. The author of the algorithm also calculated an approximation ratio for. The kMeans++ algorithm guarantees an approximation ratio O(log k) in expectation (over the randomness of the algorithm), in contrast to the original implementation which can generate randomly worse clusters than the optimum solution.

3. Practical application within this project

This project has implemented the kMeans++ algorithm and aimed to test its performance within a classification problem. The classification problem is tested using 3 datasets. The datasets are presented next.

3.1 Datasets used

Iris Flower Data Set

The Iris dataset is a multivariate data set introduced by Sir Ronald Fisher as an example of discriminant analysis. The dataset contains continuous data points. The interesting aspect of this data set is that it features 2 spatial clusters. One represents a species of plants called Iris Setosa, while the other cluster represents Iris Virginica and Iris Versicolor. The difficulty of this data set is the fact that one cluster should actually be split into 2 different clusters. The split is not immediately and is very hard to perform accurately without manual intervention.

Samples of this data set are:

Sepal length	Sepal width	Petal length	Petal width	Species*
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*

7.0	3.2	4.7	1.4	*I. versicolor*
5.0	3.6	1.4	0.2	*I. setosa*
7.7	3.8	6.7	2.2	*I. virginica*

The data set contains 150 entries.

Hepatitis Domain

The Hepatitis Domain represents a 2 class dataset. The data set features a number of 155 instances, with 19 attributes. The classes represent the fact whether a patient has the hepatitis disease or not. The attributes are mostly binary, but some contain discrete values.
The attributes are:

1. AGE: 10, 20, 30, 40, 50, 60, 70, 80
2. SEX: male, female
3. STEROID: no, yes
4. ANTIVIRALS: no, yes
5. FATIGUE: no, yes
6. MALAISE: no, yes
7. ANOREXIA: no, yes
8. LIVER BIG: no, yes
9. LIVER FIRM: no, yes
10. SPLEEN PALPABLE: no, yes
11. SPIDERS: no, yes
12. ASCITES: no, yes
13. VARICES: no, yes
14. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
15. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
16. SGOT: 13, 100, 200, 300, 400, 500,
17. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
18. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
19. HISTOLOGY: no, yes

There are missing attribute values. All the missing values are replaced with a default value of 0.

Samples of data set:
2.0 34.0 1.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 1.0 0.0 200.0 4.0 0.0

2.0 32.0 1.0 2.0 1.0 1.0 2.0 2.0 2.0 1.0 2.0 1.0 2.0 2.0 1.0 59.0 249.0
2.0 30.0 1.0 2.0 2.0 1.0 2.0 2.0 2.0 1.0 2.0 2.0 2.0 2.0 2.2 57.0 144.0 4.9 78.0
1.0 43.0 1.0 2.0 2.0 1.0 2.0 2.0 2.0 2.0 1.0 1.0 1.0 2.0 1.2 100.0 19.0 3.1 42.0

Each binary attribute is represented by a 1 or a 2.
Each continuous attribute is represented by its continuous value.

Glass Identification Database

The Glass Identification Database has been created by researcher B. German, in 1987. The data set contains a number of 214 instances. The instances feature a number of 10 attributes (including an ID for each item), all of which are continuously valued. The data set represents various percentages of chemical components within a glass material.
The attributes are:

1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron

The items represent a number of different classes. The classes are:

1. building windows float processed
2. building windows non-float processed
3. vehicle windows float processed
4. containers
5. tableware
6. headlamps

The class distribution is the following:

- 163 Window glass (building windows and vehicle windows)
    - 87 float processed
        - 70 building windows
        - 17 vehicle windows
    - 76 non-float processed
        - 76 building windows

- 51 Non-window glass
    - 13 containers
    - 9 tableware
    - 29 headlamps

3.2 Project results

The algorithm was tested with all 3 datasets to check for correctness within classification. The classification is considered correct if the cluster contains elements of only one class. The more the majority is not 100% of the elements in the class, the poorer the classification is considered.

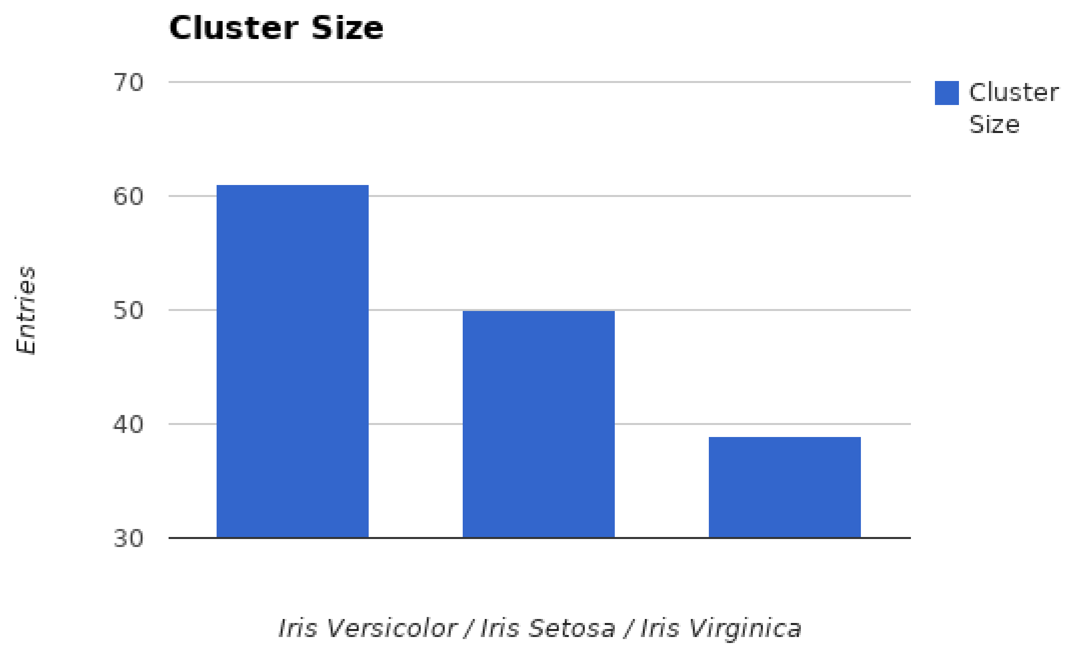
The algorithm was run, with and without the initial centroid selection improvement. Due to the sizes of the datasets, comparing runtimes is not relevant as constant factors are more relevant than the size of the input.

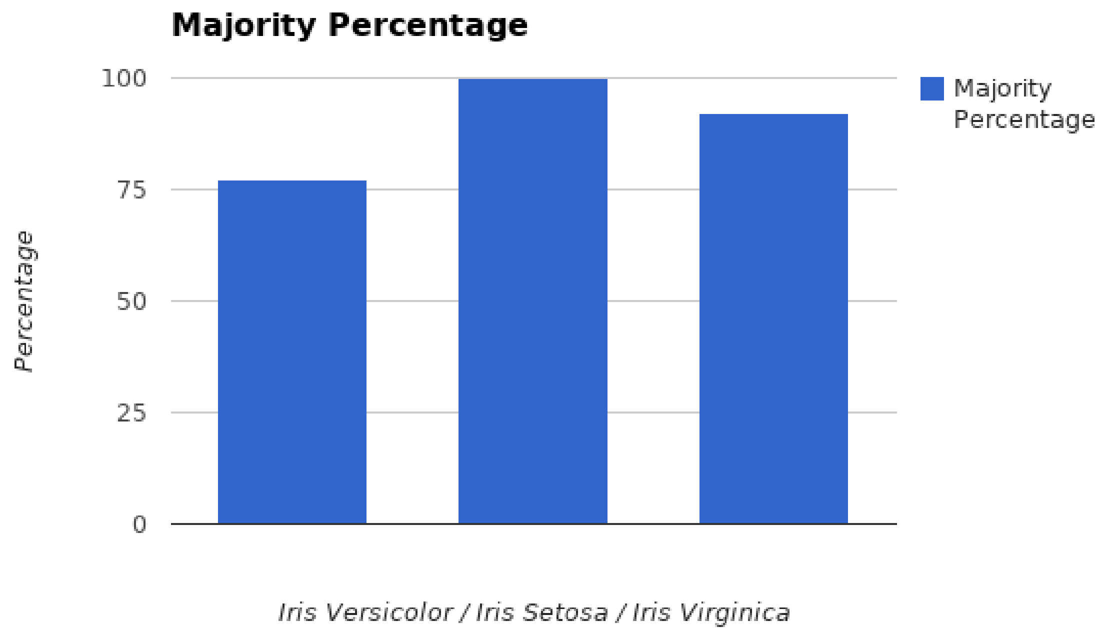
The results of running the algorithm on the data sets are presented below.

Iris Flower Data Set

Running Time	
KMeans	KMeans++
0.22	0.17

Running the Iris Flower Data Set produces surprisingly good results. The numbers are represented in the following charts:



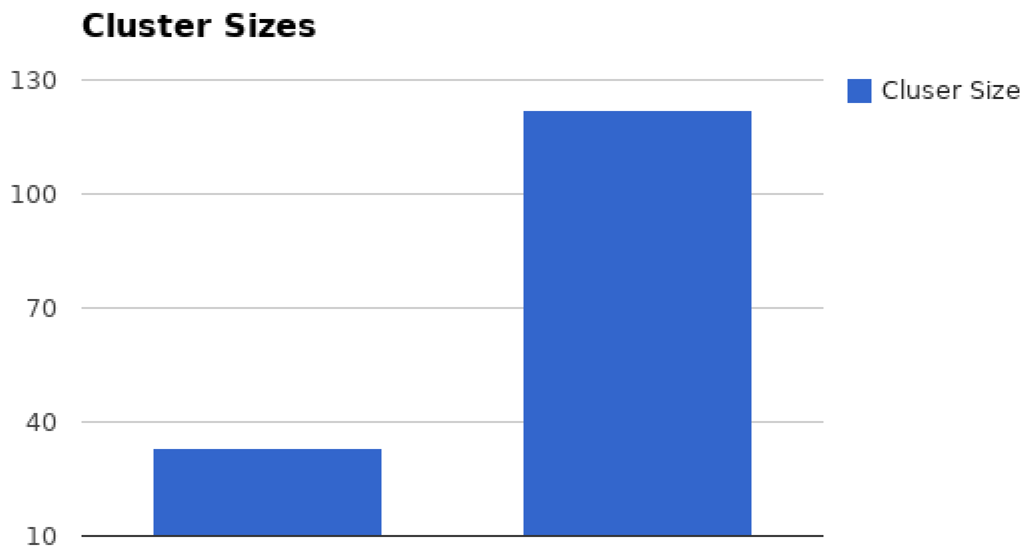


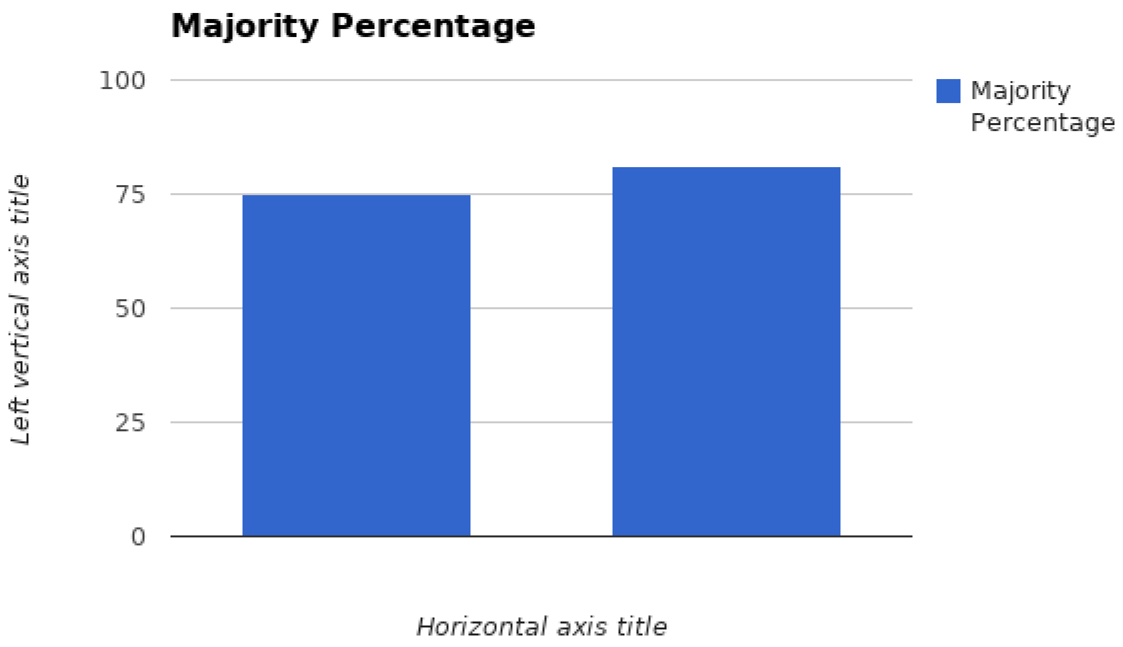
By analyzing the results for the Iris Dataset, we can determine a perfect classification for the Iris Setosa cluster. For the other two clusters, because the data is very tightly packed, there are slight errors. However the performance is well above choosing randomly. By running this classifier, greater than 75 percent of entries are classified correctly for Iris Versicolor and greater than 92 percent of entries are classified correctly as Iris Virginica.

By running the same algorithm, with both the kMeans classic version, as well as the improved version, the output results are the same.

Hepatitis Domain

Running Time	
KMeans	KMeans++
0.52	0.45





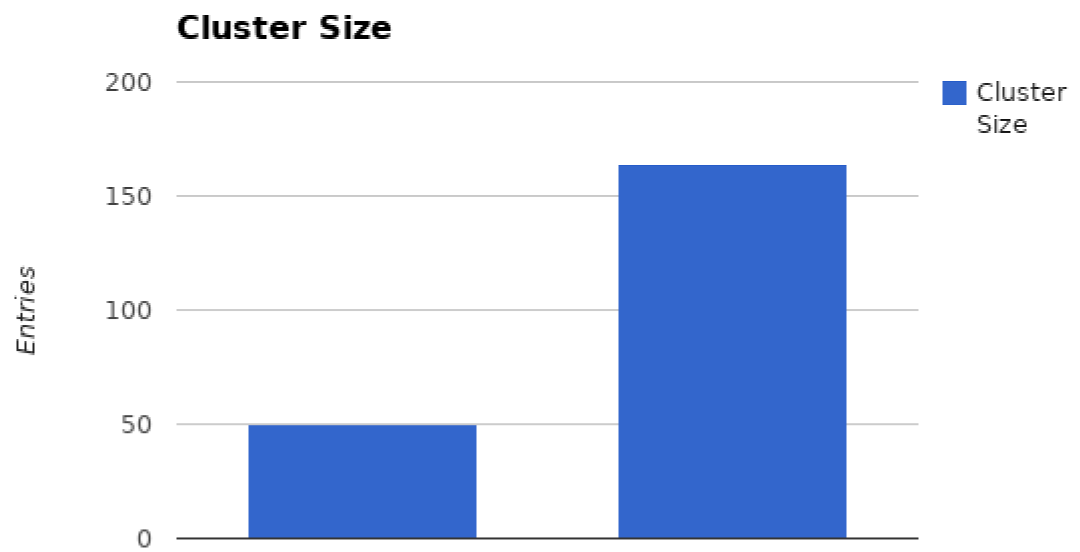
The first look at the classification looks good. There is an error of around 25% in regards to the major class within a cluster. The problem with this classification however is that both clusters have the majority of the negative class. This represents an incorrect classification.

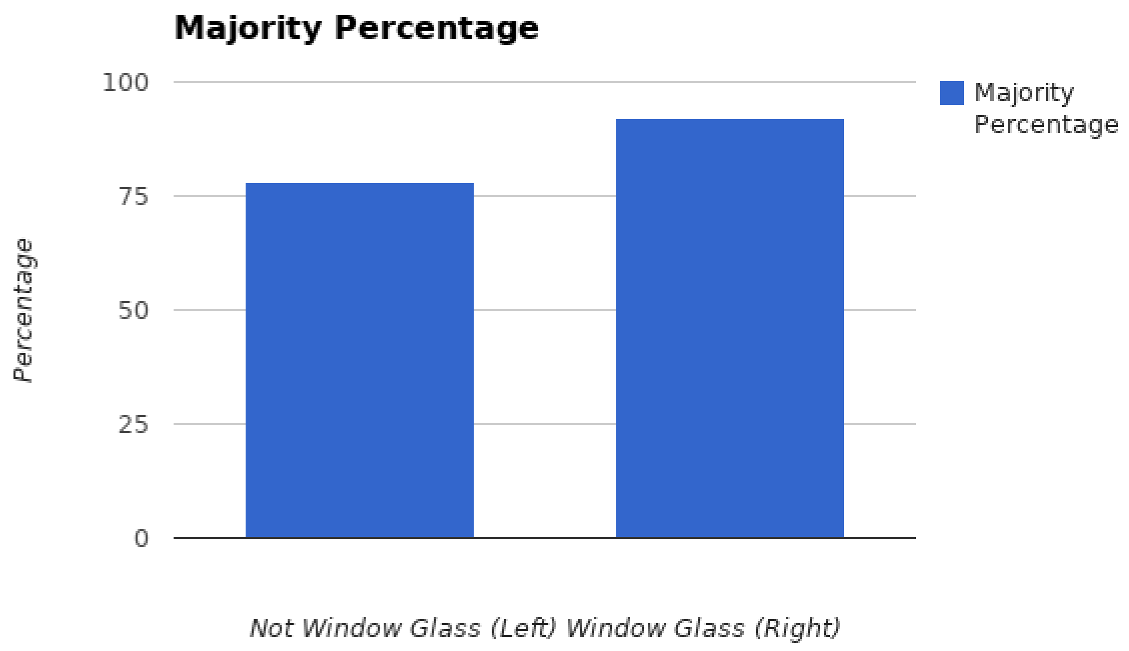
One thing to note though, is that the cluster size is proportional to the number of elements within each class. This shows that kMeans has created sensible clusters, however the grouping has reached a local optimum which however is not the global optimum.

Running the kMeans++ variant of the algorithm produces similar results. Unfortunately, due to the nature of the binary data set, the performance of the algorithm is not satisfactory. In this case, the results are highly affected by “the curse of dimensionality”. A lot of binary attributes creates a multidimensional sphere, that leaves every point almost equal in distance to all other points.

Glass Database

Running Time	
KMeans	KMeans++
0.42	0.41





Running the Glass Database set with an attempt to cluster all the 6 classes available within the dataset results in appalling results, with no better classification than a random number generator would provide. The data points are too closely packed within those classes. However, the best results are obtained by trying to classify glass within 2 classes. One class should belong to windows while the other to anything else. In this case, the results are remarkably good. Window glass gets classified correctly 88% of the time, while not window glass gets classified correctly greater than 75% of the time.

4. Conclusions

By performing the classification using both the kMeans algorithm, as well as the kMeans++ improvement, a couple of conclusions can be drawn:

- kMeans++ *improves the running time* across the board. Although more computation is done for the selection phase, this is compensated by the fact that the clusterization part converges much more quickly.
- Classification wise, the data sets are classified similarly. This can largely be attributed to the fact that the data maps relatively well to a distance function metric. All clusters are generally the same length in diameter.
- The kMeans algorithm performs surprisingly well in classifying data. The drawback is that it performs poorly with binary inputs without weights attached to them. By adding many binary attributes to an entries in a data set, the whole set becomes more predisposed to “the curse of dimensionality”. Many attributes often times impedes the classifier, instead of improving the solution.

Bibliography

[1] Yoshua Bengio (. and Yann LeCun (., "Scaling Learning Algorithms towards AI,".
[2] Bah T., "InkScape Manual - Tracing Bitmaps," http://tavmjong.free.fr/INKSCAPE/MANUAL/html/Trace-Multi.html,.
[3] Machine Learning Database
http://archive.ics.uci.edu/ml/machine-learning-databases/