COMP3008: Machine Learning
Arrhythmia

Ashley J. Robinson

School of Electronics and Computer Science
University of Southampton

May 15, 2013

1 Data handling

The data set consists of 452 instances of patient data each containing 279 attributes. This is not consist however as some attributes are not recorded for particular instances. Table 1 contains the offending attributes and the number of instances for which they are not recorded. It is clear that attribute 15 should be be neglected from the data with a missing frequency greater than 80%.

Attribute number	**Attribute name**	**Missing frequency**
11	Vector angles in degrees on front plane of T	8/452
12	Vector angles in degrees on front plane of P	22/452
13	Vector angles in degrees on front plane of QRST	1/452
14	Vector angles in degrees on front plane of J	376/452
15	Heart rate (bpm)	1/452

Table 1: Missing data

The reaming data attributes considered have a missing frequency of less than 5% so these can be considered as a positive influence towards classification. As the reaming four attributes are few in comparison to the total number no complex data completion approaches are warranted. The simplest approach is to replace the missing values with the mean of the attribute which is calculated from all instances recorded. Finally all data is normalised using equation (1) before being processed (Prügel-Bennett, 2013). Each element is replaced with a new element where μ_i is the mean and σ_i is the standard deviation of the element attribute across all instances.

$$x_{ki} \to \frac{x_{ki} - \mu_i}{\sigma_i}, \ \mu_i = \frac{1}{P}\sum_{k=1}^{P} x_{ki}, \ \sigma_i = \sqrt{\frac{1}{P-1}\sum_{k=1}^{P}(x_{ki} - \mu_{ki})^2} \tag{1}$$

2 Applicable clustering techniques

The tasks requires classification of the data using unsupervised learning techniques as the only output data is the size of each cluster and the actual classification is unknown. Table 2 contains common clustering techniques considered for this application (Leskovec and Rajaraman, 2010). K-means clustering is the most sensible choice for a preliminary investigation into this data as some small cluster sizes will prove difficult to identify with multi-class approaches. The algorithm is also simple and can be used as just a binary classifier.

Technique	**Description**
Hierarchical clustering	A bottom-up approach that starts with each data point belonging to its own cluster and gradually joining pairs of data points to form clusters.
K-means clustering	Uses centroids to separate data into a set number (K) of clusters.
Fuzzy C-means clustering	Similar to k-means but uses cluster membership functions to create clusters as apposed to hard step functions.

Table 2: Common clustering techniques

3 K-means clustering implementation

The first step initialises the algorithm by randomly selecting data points to contribute to each of the K clusters. The cluster centre is manifested as the centroid of all the points then every data point is compared with the cluster centres. Each data point is now assigned to the nearest cluster. The new cluster centres are calculated using their respective assigned data points and the whole assignment process is repeated until there is no change in the cluster vectors. All MATLAB code for this implementation is contained in appendix A.

3.1 Low dimension verification

A naive approach to the data set suggests that the condition of a heart can be determined by just the age, weight and heart rate of the patient. This is of course not a useful approach to arrhythmia classification but does produce a three dimensional pattern on which to test the algorithm. It can be seen in figure 1 that the method successfully classifies data; from visual feedback on data clustering. This increases confidence in the algorithms performance in higher dimensions.

3.2 Tuning

The aim for this classification is to assign input patterns to the classes that have been identified. The 13 classes for which respective data is contained in the database can be either be individually learnt or 12 of these classes can be grouped as positive for arrhythmia. The approach is to attempt binary classification.

Initial tests yield varied results as the algorithm will converge on different mean values for the two cluster centres. These is because of the random initialisation. This problem can be overcome by iterating over the entire algorithm. The size of the two clusters is know so it can always be assumed that the cluster for which the most data points belongs to is the normal class. A list of the cluster centres can be compiled for the two clusters and then the centroid of all these calculated. This is effectively continuing the algorithm by taking another centroid of data points.

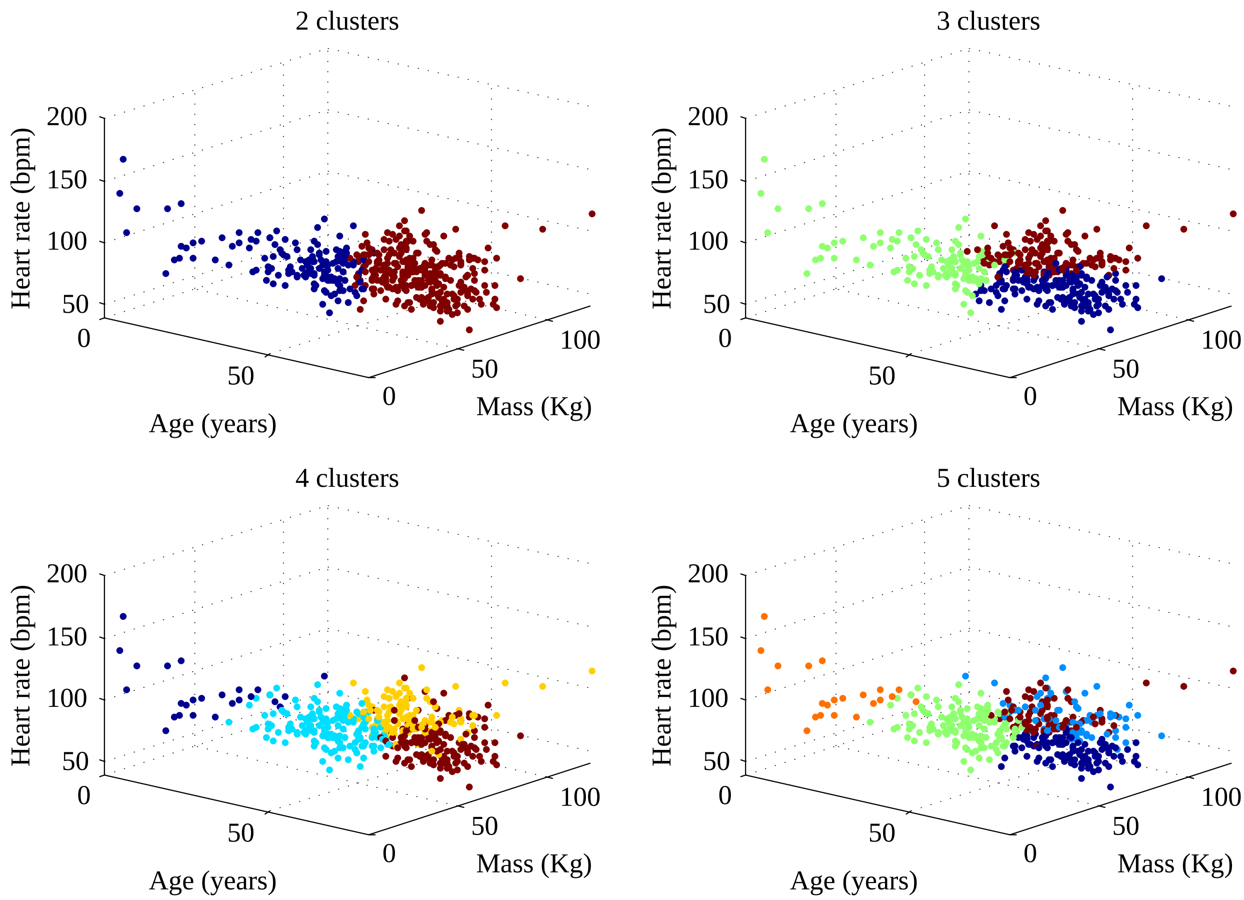


FIGURE 1: K-means clustering on three dimensional data

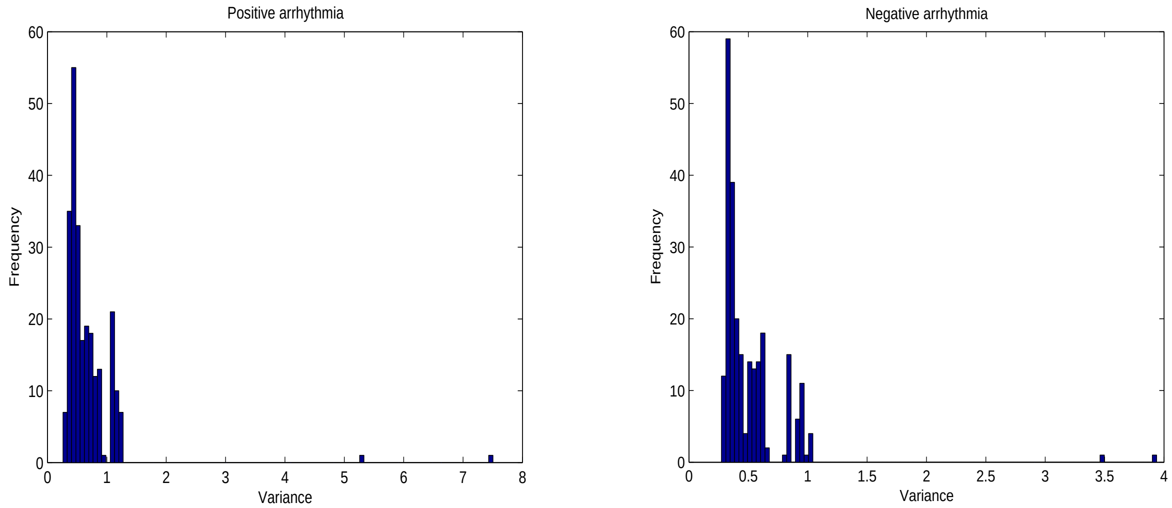
Occasionally initialisation may lead to cluster sizes that are almost identical in which case it is hard guarantee a correct classification of the centre. This can be overcome by setting a threshold for clusters with similar sizes then simply choosing to disregard the outcome of that iteration.

4 Performance

Using the mean output of all the cluster centres the variance can be calculated by comparing the individual runs to the final solution. Equation (2) contains the mean output clusters ($\hat{\mathbf{y}}_i$) from n iterations. The variance is the Euclidean distance between the mean cluster centres ($\hat{\mathbf{y}}_i$) and each of the iteration cluster centres ($\mathbf{y}_{ik}$).

The two histograms in figure 2 display the variance of the algorithm output over 250 iterations. Both exhibit grouping of variance. In fact the groupings represent a convergence on the wrong cluster. Using these incorrectly classified values to calculate the mean, and therefore the variance, throws all values off.

$$V_{ik} = ||\hat{\mathbf{y}}_i - \mathbf{y}_{ik}||, \quad \hat{\mathbf{y}}_i = \frac{1}{n}\sum_{k=1}^{n} \mathbf{y}_{ik}, \quad i \in 1, 2 \tag{2}$$

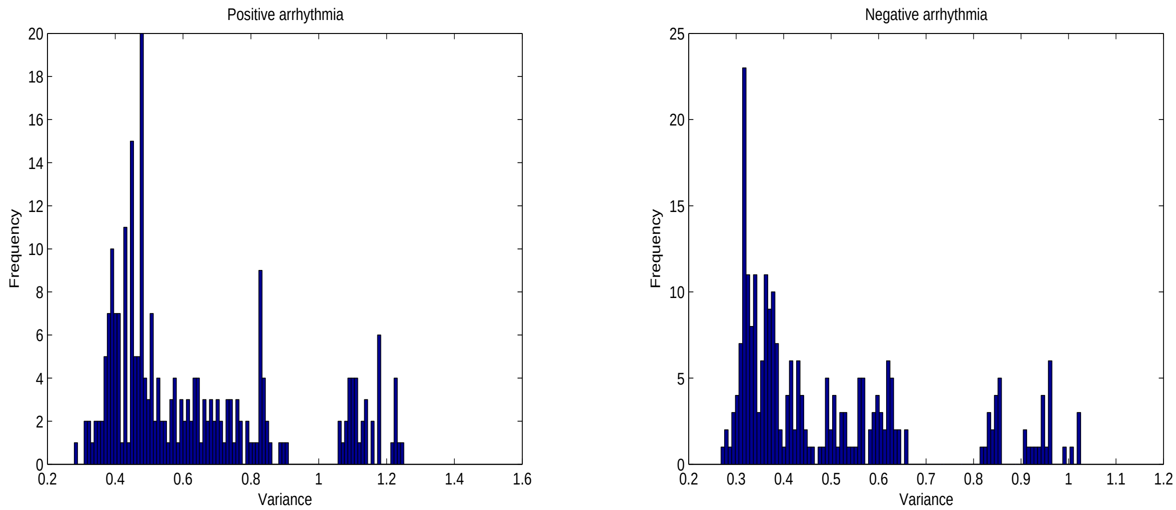


(a) Positive class

(b) Negative Class

FIGURE 2: Histograms of variance through algorithm iterations

The outliers are removed from figure 2 and the variance is recalculated[1] for 248 iterations in figure 3. This also shows a grouping because of the unpopulated variance bins in the middle but this instead is down to the nature of the data. The algorithm will converge in different centres depending on the initialisation.



(a) Positive class

(b) Negative Class

FIGURE 3: Fixed histograms of variance through algorithm iterations with outliers removed

The mean variance for this data is 0.4259 for the negative clusters and 0.5509 for the positive clusters. This is an estimation on the performance and reveals a need for improvement but shows adequate classification. The positive skew of both histograms in figure 3 verifies the performance.

[1] Code for kicking data points and iterating the algorithm is contained in listing 8

5 Conclusions and future work

While actual classification of the data in unknown this approach yielded good results after analysis of the variance. The algorithm would need adapting to reject wrongly classified output itself. This would remove the higher values of variance and improve the mean variance of the solution. The quantity threshold for assuming data belongs to a class also needs further investigation.

A much better approach could be produced using supervised learning. Labelled data could be obtained from another data set or through medical analysis of this data set.

References

Jure Leskovec and Anand Rajaraman. CS345a:Data Mining, 2010. Stanford University, http://www.stanford.edu/class/cs345a/slides/12-clustering.pdf.

Adam Prügel-Bennett. COMP3008 Machine Learning, 2013. Southampton University.

A Code listings

```
%Ashley Robinson
%10/04/12
%COMP3008
%K_Means_Clustering.m
clear
% K >= 2
K = 2;
%Load the data
data = LoadData('arrhythmia.data');%Load the data from text file
data = Normalise(data);
instances = size(data,2);%These are handy to know
attributes = size(data,1);
%populate all means
count = 0;
while(min(count) == 0)
   %initialise
   assigned_cluster = randi([1,K],1,instances);
   [old_cluster_mean] = Calculate_Means(assigned_cluster,K,data);
   assigned_cluster = Assign_New_Cluster(old_cluster_mean,K,data);%Assign data to random
    cluster means
   [cluster_mean] = Calculate_Means(assigned_cluster,K,data);
   %Converge
   i = 0;
   while(sum(abs(sum(old_cluster_mean - cluster_mean)  ~= zeros(1,K))))%Test for same means
      old_cluster_mean = cluster_mean;
      assigned_cluster = Assign_New_Cluster(cluster_mean,K,data);
      [cluster_mean] = Calculate_Means(assigned_cluster,K,data);
      i = i + 1;
      Output(i,K,instances,assigned_cluster);%User output
   end
   count = Cluster_Count(K,instances,assigned_cluster);%How many data points to each cluster?
end
```

Listing 1: Main algorithm

```
%Ashley Robinson
%10/04/12
%COMP3008
%LoadData.m
function[data] = LoadData(Path_to_data)
   %Open data file in same directory
   arrhythmia_data = fopen(Path_to_data);
   %global init
   instance = 0;
   while ~feof(arrhythmia_data)%While not at the end of the file
      instance = instance + 1;
      line = fgetl(arrhythmia_data);%Read a line from the file
      start = 1;
      %disp(line)%Just to see some output
      attribute = 0;
      for j=1:length(line)
         %Look for CSVs
         if (line(j) == ',')
            attribute = attribute + 1;
            if (line(start:(j - 1)) == '?')
               miss_mask(attribute,instance) = 1;%take a note of missing
               data(attribute,instance) = 0;
            else
               miss_mask(attribute,instance) = 0;%need to keep dimensions up
               data(attribute,instance) = str2num(line(start:(j - 1)));
            end
            start = j + 1;%One in front of the comma
         end
      end
   end
   fclose(arrhythmia_data);
   %Replace unkown data with the mean of that attribute
   for i=1:attribute
```

```
      total = 0;
      count = 0;
      for j=1:instance
         if(miss_mask(i,j) == 0)
            total = total + data(i,j);
            count = count + 1;
         end
      end
      total = total./count;%The mean of this attribute
      miss_mask(i,:) = miss_mask(i,:).*total;
   end
   data = data + miss_mask;%Missing values are zero so add miss mask
   %Remove attribute 14
   att_point = 1;
   for i=1:attribute
      if(i ~= 14)
         for j=1:instance
            temp(att_point,j) = data(i,j);
         end
         att_point = att_point + 1;
      end
   end
   data = temp;
   %All info now in matrix
   info =  sprintf('Instances: %d \nAttributes: %d ',size(data,2), size(data,1));
   disp(info)
end
```

Listing 2: Load the data from the raw text file

```
%Ashley Robinson
%21/04/12
%COMP3008
%Normalise.m
function[new_data] = Normalise(data)
   instances = size(data,2);%These are handy to know
   attributes = size(data,1);
   for i=1:attributes
      mu = data(i,1);
      for j=2:instances
         mu = mu + data(i,j);
      end
      mu = mu./instances;
      sigma = (data(i,1) - mu)^2;
      for j=2:instances
         sigma = sigma + (data(i,j) - mu)^2;
      end
      sigma = sqrt(sigma./(instances - 1));
      for j=1:instances
         if(sigma == 0)
            new_data(i,j) = 0;
         else
            new_data(i,j) = (data(i,j) - mu)./sigma;
      end
   end
end
```

Listing 3: Normalise the data

```
%Ashley Robinson
%10/04/12
%COMP3008
%Calculate_Means.m
function[cluster_mean] = Calculate_Means(assigned_cluster,K,data)
   instances = size(data,2);
   attributes = size(data,1);
   for i=1:K
      cluster_mean(:,i) = zeros(attributes,1);
      count = 0;
      for j=1:instances
         if(assigned_cluster(j) == i)
```

```
            count = count + 1;%Keep track of the number of instances in the cluster
            cluster_mean(:,i) = cluster_mean(:,i) + data(:,j);%Total all the vectors
         end
      end
      if(count ~= 0)
         cluster_mean(:,i) = cluster_mean(:,i)./count;%calculate mean
      end
   end
end
```

Listing 4: Calculate the mean of a cluster of data points

```
%Ashley Robinson
%10/04/12
%COMP3008
%Assign_New_Cluster.m
function[assigned_cluster] = Assign_New_Cluster(cluster_mean,K,data)
   for i=1:size(data,2)%instances
      assigned_cluster(i) = 1;%init
      dist = Calculate_Distance(cluster_mean(:,1),data(:,i));
      for j=2:K
         test = Calculate_Distance(cluster_mean(:,j),data(:,i));
         if(test < dist)
            assigned_cluster(i) = j;
            dist = test;
         end
      end
   end
end
```

Listing 5: Assign each data point to the nearest cluster cluster

```
%Ashley Robinson
%10/04/12
%COMP3008
%Output.m
function[] = Output(iter,K,instances,assigned_cluster)
   count = Cluster_Count(K,instances,assigned_cluster);%How many data points to each cluster
   disp(sprintf('Iteration: %d',iter));
   for i=1:K
      disp(sprintf('              Cluster %d size: %d ',i,count(i)))
   end
end
```

Listing 6: View some useful information about what is going on

```
%Ashley Robinson
%10/04/12
%COMP3008
%Cluster_Count.m
function[count] = Cluster_Count(K,instances,assigned_cluster)
   count = zeros(K,1);
   for i=1:instances
      for j=1:K
         if(assigned_cluster(i) == j)
            count(j) = count(j) + 1;
         end
      end
   end
end
```

Listing 7: How many data points are associated with each cluster

```
%Ashley Robinson
%10/04/12
%COMP3008
%Iterate_K_Means.m
clear
%Binary fixed
```

```
K = 2;
iterations = 250;

%Load the data
data = LoadData('arrhythmia.data');%Load the data from text file
data = Normalise(data);
instances = size(data,2);%These are handy to know
attributes = size(data,1);


for iter=1:iterations
    info = sprintf('Iteration: %d',iter)
    disp(info)
   %populate all means
   count = 0;
   while(min(count) == 0)
      %initialise
      assigned_cluster = randi([1,K],1,instances);
      [old_cluster_mean] = Calculate_Means(assigned_cluster,K,data);
      assigned_cluster = Assign_New_Cluster(old_cluster_mean,K,data);%Assign data to random cluster means
      [cluster_mean] = Calculate_Means(assigned_cluster,K,data);
      %Converge
      i = 0;
      while(sum(abs(sum(old_cluster_mean - cluster_mean)  ~= zeros(1,K))))%Test for same means
         old_cluster_mean = cluster_mean;
         assigned_cluster = Assign_New_Cluster(cluster_mean,K,data);
         [cluster_mean] = Calculate_Means(assigned_cluster,K,data);
         i = i + 1;
         %Output(i,K,instances,assigned_cluster);%User output
      end
      count = Cluster_Count(K,instances,assigned_cluster)%How many data points to each cluster?
      if(abs(count(1) - count(2)) < 20)
          count = 0;
          disp('close')
      end
   end
   cluster(:,:,iter) = cluster_mean;
end

%First mean holds good patients
assigned_cluster = Assign_New_Cluster(cluster(:,:,1),K,data);
count = Cluster_Count(K,instances,assigned_cluster);
if(count(1) > count(2))
   mean_cluster(:,1) = cluster(:,1,1);
   mean_cluster(:,2) = cluster(:,2,1);
else
   mean_cluster(:,2) = cluster(:,1,1);
   mean_cluster(:,1) = cluster(:,2,1);
end

for iter=2:iterations
   assigned_cluster = Assign_New_Cluster(cluster(:,:,iter),K,data);
   count = Cluster_Count(K,instances,assigned_cluster);

   if(count(1) > count(2))
      mean_cluster(:,1) = mean_cluster(:,1) + cluster(:,1,iter);
      mean_cluster(:,2) = mean_cluster(:,2) + cluster(:,2,iter);
   else
      mean_cluster(:,2) = mean_cluster(:,2) + cluster(:,1,iter);
      mean_cluster(:,1) = mean_cluster(:,1) + cluster(:,2,iter);
   end
end
mean_cluster = mean_cluster./iterations
assigned_cluster = Assign_New_Cluster(mean_cluster,K,data);
count = Cluster_Count(K,instances,assigned_cluster)

for iter=1:iterations
   assigned_cluster = Assign_New_Cluster(cluster(:,:,iter),K,data);
   count = Cluster_Count(K,instances,assigned_cluster);
```

```
    if(count(1) > count(2))
       variance(iter,1) = Calculate_Distance(mean_cluster(:,1),cluster(:,1,iter));
       variance(iter,2) = Calculate_Distance(mean_cluster(:,2),cluster(:,2,iter));
    else
       variance(iter,2) = Calculate_Distance(mean_cluster(:,2),cluster(:,1,iter));
       variance(iter,1) = Calculate_Distance(mean_cluster(:,1),cluster(:,2,iter));
    end
end
figure(1)
hist(variance(:,1),100)
xlabel('Variance')
ylabel('Frequency')
title('Cluster 1')
figure(2)
hist(variance(:,2),100)
title('Cluster 2')
xlabel('Variance')
ylabel('Frequency')
variance


kick1 =input('kick 1 (0 to quit):')
kick2 =input('kick 2:')
while(sum(kick1) ~= 0)
%------------variance fix
j = 0;
for iter=1:iterations
    if(sum(variance(iter,1) > kick1) | sum(variance(iter,2) > kick2))
        disp('out')
    else
        j = j + 1;
        new_cluster(:,:,j) = cluster(:,:,iter);
    end
end

%First mean holds good patients
assigned_cluster = Assign_New_Cluster(new_cluster(:,:,1),K,data);
count = Cluster_Count(K,instances,assigned_cluster);
if(count(1) > count(2))
    mean_cluster(:,1) = new_cluster(:,1,1);
    mean_cluster(:,2) = new_cluster(:,2,1);
else
    mean_cluster(:,2) = new_cluster(:,1,1);
    mean_cluster(:,1) = new_cluster(:,2,1);
end

for iter=2:j
    assigned_cluster = Assign_New_Cluster(new_cluster(:,:,iter),K,data);
    count = Cluster_Count(K,instances,assigned_cluster);

    if(count(1) > count(2))
       mean_cluster(:,1) = mean_cluster(:,1) + new_cluster(:,1,iter);
       mean_cluster(:,2) = mean_cluster(:,2) + new_cluster(:,2,iter);
    else
       mean_cluster(:,2) = mean_cluster(:,2) + new_cluster(:,1,iter);
       mean_cluster(:,1) = mean_cluster(:,1) + new_cluster(:,2,iter);
    end
end
mean_cluster = mean_cluster./j
assigned_cluster = Assign_New_Cluster(mean_cluster,K,data);
count = Cluster_Count(K,instances,assigned_cluster)

for iter=1:j
    assigned_cluster = Assign_New_Cluster(new_cluster(:,:,iter),K,data);
    count = Cluster_Count(K,instances,assigned_cluster);
    if(count(1) > count(2))
       new_variance(iter,1) = Calculate_Distance(mean_cluster(:,1),new_cluster(:,1,iter));
       new_variance(iter,2) = Calculate_Distance(mean_cluster(:,2),new_cluster(:,2,iter));
    else
       new_variance(iter,2) = Calculate_Distance(mean_cluster(:,2),new_cluster(:,1,iter));
       new_variance(iter,1) = Calculate_Distance(mean_cluster(:,1),new_cluster(:,2,iter));
```

```
   end
end
figure(3)
hist(new_variance(:,1),100)
xlabel('Variance')
ylabel('Frequency')
title('Cluster 1')
figure(4)
hist(new_variance(:,2),100)
title('Cluster 2')
xlabel('Variance')
ylabel('Frequency')
new_variance
kick1 =input('kick 1 (0 to quit):')
kick2 =input('kick 2:')
end
```

Listing 8: Used to iterate over the algorithm and kick output from the variance calculation