Lab 3
732A61 Data Mining - Clustering and Association Analysis

Carles Sans Fuentes

March 6, 2017

Assignment 1

In this assignment I am going to work with the data set monk1.arff. This is an artificial dataset with 124 instances, each described by 6 discrete attributes and a binary class attribute.

Clustering

First, I am asked to cluster the data with different algorithms and number of clusters. The ones that are going to be used are the (1) Simplekmeans and (2) the XXX with (3) 3 Clusters and (4) 4 Cluster such that this are 4 options. I am going to use the clusters to class evaluation model to see whether the clustering algorithm is able to discover the class division existing in the data.

Exercise 1 Clustering

3 Clusters k-means

The algorithm "SimpleKMeans" has been applied to your data. In Weka euclidian distance is implemented in SimpleKmeans and the seed chosen for this case (ignoring the class attribute) is 10. The results are provided here below for three clusters:

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000
    -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I
    500 -num-slots 1 -S 10
Relation:     monk1
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 314.0

Initial staring points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute      Full Data          0          1          2
                   (124)       (59)       (38)       (27)
=========================================================
attribute#1            1          1          3          2
attribute#2            3          2          1          3
attribute#3            1          1          2          1
attribute#4            3          1          3          2
attribute#5            4          3          2          1
attribute#6            2          2          1          1




Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       59 ( 48%)
1       38 ( 31%)
2       27 ( 22%)
```

4 Clusters k-means

The algorithm ”SimpleKMeans” has been applied to your data for 4 clusters. The seed chosen for this case (ignoring the class attribute) is 10. The results are provided here below for four clusters:

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000
    -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I
    500 -num-slots 1 -S 10
Relation:     monk1
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 293.0

Initial staring points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1
Cluster 3: 2,3,1,3,1,2

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute      Full Data          0          1          2          3
                   (124)       (50)       (36)       (24)       (14)
====================================================================
attribute#1            1          1          3          2          2
attribute#2            3          2          1          3          3
attribute#3            1          1          2          1          1
attribute#4            3          1          3          2          3
```

```
attribute#5             4          3          2          1          1
attribute#6             2          2          1          1          2




Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       50 ( 40%)
1       36 ( 29%)
2       24 ( 19%)
3       14 ( 11%)
```

Makedensitydbasedclusterer with clusterer kmeans with 3 clusters

The algorithm "makedensitydbasedclusterer" has been applied to the data with default min standard deviation equals to 1.0E-6. As a clusterer for the algorithm, the SimpleKmeans with 3 clusters has been used as well as a seed chosen for this case (ignoring the class attribute) which is 10. The results are provided here below for three clusters:

```
=== Run information ===

Scheme:       weka.clusterers.MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.
     SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1
     -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S
     10
Relation:     monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
     SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
      -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-I7
Instances:    124
Attributes:   8
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
              cluster
Test mode:    evaluate on training data


=== Clustering model (full training set) ===

MakeDensityBasedClusterer:

Wrapped clusterer:
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 314.0

Initial staring points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1          2
                   (124)       (59)       (38)       (27)
=========================================================
attribute#1             1          1          3          2
attribute#2             3          2          1          3
attribute#3             1          1          2          1
attribute#4             3          1          3          2
attribute#5             4          3          2          1
attribute#6             2          2          1          1



```

```
Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.4724

Attribute: attribute#1
Discrete Estimator. Counts =  32 17 13  (Total = 62)
Attribute: attribute#2
Discrete Estimator. Counts =  9 34 19  (Total = 62)
Attribute: attribute#3
Discrete Estimator. Counts =  37 24  (Total = 61)
Attribute: attribute#4
Discrete Estimator. Counts =  30 14 18  (Total = 62)
Attribute: attribute#5
Discrete Estimator. Counts =  8 13 23 19  (Total = 63)
Attribute: attribute#6
Discrete Estimator. Counts =  16 45  (Total = 61)

Cluster: 1 Prior probability: 0.3071

Attribute: attribute#1
Discrete Estimator. Counts =  9 10 22  (Total = 41)
Attribute: attribute#2
Discrete Estimator. Counts =  21 7 13  (Total = 41)
Attribute: attribute#3
Discrete Estimator. Counts =  9 31  (Total = 40)
Attribute: attribute#4
Discrete Estimator. Counts =  11 9 21  (Total = 41)
Attribute: attribute#5
Discrete Estimator. Counts =  9 17 6 10  (Total = 42)
Attribute: attribute#6
Discrete Estimator. Counts =  24 16  (Total = 40)

Cluster: 2 Prior probability: 0.2205

Attribute: attribute#1
Discrete Estimator. Counts =  7 18 5  (Total = 30)
Attribute: attribute#2
Discrete Estimator. Counts =  8 4 18  (Total = 30)
Attribute: attribute#3
Discrete Estimator. Counts =  22 7  (Total = 29)
Attribute: attribute#4
Discrete Estimator. Counts =  4 19 7  (Total = 30)
Attribute: attribute#5
Discrete Estimator. Counts =  15 4 4 8  (Total = 31)
Attribute: attribute#6
Discrete Estimator. Counts =  19 10  (Total = 29)


Time taken to build model (full training data) : 0.09 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       60 ( 48%)
1       39 ( 31%)
2       25 ( 20%)


Log likelihood: -6.09108
```

Makedensitydbasedclusterer with clusterer kmeans with 4 clusters

The algorithm "makedensitydbasedclusterer" has been applied to the data with default min standard deviation equals to 1.0E-6. As a clusterer for the algorithm, the SimpleKmeans with 4 clusters has been used as well as a seed chosen for this case (ignoring the class attribute) which is 10. The results are provided here below for four clusters:

```
=== Run information ===

Scheme:       weka.clusterers.MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.
     SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1
     -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S
     10
Relation:     monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
     SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
      -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-I7
Instances:    124
Attributes:   8
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
              cluster
Test mode:    evaluate on training data


=== Clustering model (full training set) ===

MakeDensityBasedClusterer:

Wrapped clusterer:
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 293.0

Initial staring points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1
Cluster 3: 2,3,1,3,1,2

Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1          2          3
                   (124)       (50)       (36)       (24)       (14)
====================================================================
attribute#1            1          1          3          2          2
attribute#2            3          2          1          3          3
attribute#3            1          1          2          1          1
attribute#4            3          1          3          2          3
attribute#5            4          3          2          1          1
attribute#6            2          2          1          1          2



Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.3984

Attribute: attribute#1
Discrete Estimator. Counts =  28 13 12  (Total = 53)
Attribute: attribute#2
Discrete Estimator. Counts =  9 31 13  (Total = 53)
Attribute: attribute#3
Discrete Estimator. Counts =  30 22  (Total = 52)
Attribute: attribute#4
Discrete Estimator. Counts =  30 14 9  (Total = 53)
Attribute: attribute#5
Discrete Estimator. Counts =  6 11 22 15  (Total = 54)
Attribute: attribute#6
Discrete Estimator. Counts =  16 36  (Total = 52)

Cluster: 1 Prior probability: 0.2891

Attribute: attribute#1
Discrete Estimator. Counts =  9 10 20  (Total = 39)
Attribute: attribute#2
Discrete Estimator. Counts =  21 7 11  (Total = 39)
Attribute: attribute#3
Discrete Estimator. Counts =  8 30  (Total = 38)
Attribute: attribute#4
Discrete Estimator. Counts =  11 9 19  (Total = 39)
Attribute: attribute#5
Discrete Estimator. Counts =  8 16 6 10  (Total = 40)
Attribute: attribute#6
Discrete Estimator. Counts =  24 14  (Total = 38)

Cluster: 2 Prior probability: 0.1953

Attribute: attribute#1
Discrete Estimator. Counts =  7 16 4  (Total = 27)
Attribute: attribute#2
Discrete Estimator. Counts =  8 4 15  (Total = 27)
Attribute: attribute#3
Discrete Estimator. Counts =  19 7  (Total = 26)
Attribute: attribute#4
Discrete Estimator. Counts =  4 19 4  (Total = 27)
```

```
Attribute: attribute#5
Discrete Estimator. Counts =  13 4 4 7  (Total = 28)
Attribute: attribute#6
Discrete Estimator. Counts =  19 7  (Total = 26)

Cluster: 3 Prior probability: 0.1172

Attribute: attribute#1
Discrete Estimator. Counts =  5 7 5  (Total = 17)
Attribute: attribute#2
Discrete Estimator. Counts =  1 4 12  (Total = 17)
Attribute: attribute#3
Discrete Estimator. Counts =  12 4  (Total = 16)
Attribute: attribute#4
Discrete Estimator. Counts =  1 1 15  (Total = 17)
Attribute: attribute#5
Discrete Estimator. Counts =  6 4 2 6  (Total = 18)
Attribute: attribute#6
Discrete Estimator. Counts =  1 15  (Total = 16)


Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       51 ( 41%)
1       35 ( 28%)
2       23 ( 19%)
3       15 ( 12%)


Log likelihood: -6.06035
```

Explanation of the clustering part

It can be seen that three and 4 clusters have been created using the algorithm K-means and the Makedensitydbasedclusterer with the clusterer kmeans with 3 and 4 clusters. Still, not a clear cluster result has been reached. This makes sense somehow because I have ignored the class label which classifies data in two clusters. Thus, classifying for 3 or 4 clusters using algorithms does not provide with much information (at least in this case) neither with a good clustering. Also, initial classification is almost exactly the same (it always changed by one in both algorithms) for the different algorithms and clusters. This might mean that the algorithm does not work good in order to be able to change from one state to another. Here below I will try just before starting with the association to check for the kmeans algorithm with two clusters and compare it as well with the class classification.

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000
    -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I
    500 -num-slots 1 -S 10
Relation:     monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
    SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
     -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-I7-
    weka.filters.unsupervised.attribute.Remove-R8
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 3
```

```
Within cluster sum of squared errors: 358.0

Initial staring points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                          Cluster#
Attribute      Full Data          0           1
                   (124)       (77)        (47)
==============================================
attribute#1            1          1           3
attribute#2            3          2           3
attribute#3            1          1           2
attribute#4            3          1           3
attribute#5            4          4           2
attribute#6            2          2           1




Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       77 ( 62%)
1       47 ( 38%)
```

It still happens the same as before, leading to almost no changes from the initial points and to a non-clear clustering. The clustering algorithm cannot find a good cluster class division because the data is classified according to rules and not according to distances between points. For this reason, trying to predict clusters using distances to create the centroid does not give a good classification, because the classification is not based on distances (discrete data), but on categorical data. Therefore, the algorithms used are not good for our case.

Exercise 2

In this part I am going to perform association analysis to find a set of rules that are able to accurately predict the class label from the rest of the attributes. It is recommended by the lab to use a minimum support of 0.05 and a maximum number of rules of 19, which is the information that is going to be used for that. The apriori algorithm is used and it founds the following results:

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 19 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.05 -S -1.0 -c -1
Relation:     monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
    SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
     -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-I7-
    weka.filters.unsupervised.attribute.Remove-R8
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
              class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.05 (6 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 19

Generated sets of large itemsets:

Size of set of large itemsets L(1): 19

Size of set of large itemsets L(2): 151

Size of set of large itemsets L(3): 378

Size of set of large itemsets L(4): 125

Size of set of large itemsets L(5): 6

Best rules found:

 1. attribute#5=1 29 ==> class=1 29    <conf:(1)> lift:(2) lev:(0.12) [14] conv:(14.5)
 2. attribute#1=3 attribute#2=3 17 ==> class=1 17    <conf:(1)> lift:(2) lev:(0.07) [8] conv
     :(8.5)
 3. attribute#3=1 attribute#5=1 17 ==> class=1 17    <conf:(1)> lift:(2) lev:(0.07) [8] conv
     :(8.5)
 4. attribute#5=1 attribute#6=1 16 ==> class=1 16    <conf:(1)> lift:(2) lev:(0.06) [8] conv
     :(8)
 5. attribute#1=2 attribute#2=2 15 ==> class=1 15    <conf:(1)> lift:(2) lev:(0.06) [7] conv
     :(7.5)
 6. attribute#1=3 attribute#5=1 13 ==> class=1 13    <conf:(1)> lift:(2) lev:(0.05) [6] conv
     :(6.5)
 7. attribute#5=1 attribute#6=2 13 ==> class=1 13    <conf:(1)> lift:(2) lev:(0.05) [6] conv
     :(6.5)
 8. attribute#2=3 attribute#5=1 12 ==> class=1 12    <conf:(1)> lift:(2) lev:(0.05) [6] conv
     :(6)
 9. attribute#3=2 attribute#5=1 12 ==> class=1 12    <conf:(1)> lift:(2) lev:(0.05) [6] conv
     :(6)
10. attribute#1=3 attribute#2=3 attribute#6=2 12 ==> class=1 12    <conf:(1)> lift:(2) lev
    :(0.05) [6] conv:(6)
11. attribute#4=1 attribute#5=1 11 ==> class=1 11    <conf:(1)> lift:(2) lev:(0.04) [5] conv
    :(5.5)
12. attribute#1=2 attribute#5=1 10 ==> class=1 10    <conf:(1)> lift:(2) lev:(0.04) [5] conv
    :(5)
13. attribute#2=2 attribute#5=1 10 ==> class=1 10    <conf:(1)> lift:(2) lev:(0.04) [5] conv
    :(5)
14. attribute#1=1 attribute#2=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04) [4] conv
    :(4.5)
15. attribute#4=2 attribute#5=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04) [4] conv
    :(4.5)
16. attribute#4=3 attribute#5=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04) [4] conv
    :(4.5)
17. attribute#1=2 attribute#2=2 attribute#3=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04)
     [4] conv:(4.5)
18. attribute#1=3 attribute#2=3 attribute#3=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04)
     [4] conv:(4.5)
19. attribute#3=1 attribute#5=1 attribute#6=1 9 ==> class=1 9    <conf:(1)> lift:(2) lev:(0.04)
     [4] conv:(4.5)
```

Now I am going to set the variable "car" in the a priori algorithm to true setting as well the index to the class variable to get the rules directly:

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 19 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.05 -S -1.0 -A -c
    7
Relation:     monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
    SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
     -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-I7-
    weka.filters.unsupervised.attribute.Remove-R8
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
              class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.05 (6 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 19

Generated sets of large itemsets:

Size of set of large itemsets L(1): 33

Size of set of large itemsets L(2): 141

Size of set of large itemsets L(3): 59

Size of set of large itemsets L(4): 1

Best rules found:

 1. attribute#5=1 29 ==> class=1 29    conf:(1)
 2. attribute#1=3 attribute#2=3 17 ==> class=1 17    conf:(1)
 3. attribute#3=1 attribute#5=1 17 ==> class=1 17    conf:(1)
 4. attribute#5=1 attribute#6=1 16 ==> class=1 16    conf:(1)
 5. attribute#1=2 attribute#2=2 15 ==> class=1 15    conf:(1)
 6. attribute#1=3 attribute#5=1 13 ==> class=1 13    conf:(1)
 7. attribute#5=1 attribute#6=2 13 ==> class=1 13    conf:(1)
 8. attribute#2=3 attribute#5=1 12 ==> class=1 12    conf:(1)
 9. attribute#3=2 attribute#5=1 12 ==> class=1 12    conf:(1)
10. attribute#1=3 attribute#2=3 attribute#6=2 12 ==> class=1 12    conf:(1)
11. attribute#4=1 attribute#5=1 11 ==> class=1 11    conf:(1)
12. attribute#1=2 attribute#5=1 10 ==> class=1 10    conf:(1)
13. attribute#2=2 attribute#5=1 10 ==> class=1 10    conf:(1)
14. attribute#1=1 attribute#2=1 9 ==> class=1 9    conf:(1)
15. attribute#4=2 attribute#5=1 9 ==> class=1 9    conf:(1)
16. attribute#4=3 attribute#5=1 9 ==> class=1 9    conf:(1)
17. attribute#1=2 attribute#2=2 attribute#3=1 9 ==> class=1 9    conf:(1)
18. attribute#1=3 attribute#2=3 attribute#3=1 9 ==> class=1 9    conf:(1)
19. attribute#3=1 attribute#5=1 attribute#6=1 9 ==> class=1 9    conf:(1)
```

Eliminating association rules

Giving the fact that the rules in our case are monotonous, e.g if the set A is true, for any superset of A called B, then the rule for A still holds, we can eliminate from the first rule all other supersets getting the following 4 set of rules:

```
1. attribute#5=1 29 ==> class=1 29    conf:(1)
2. attribute#1=3 attribute#2=3 17 ==> class=1 17    conf:(1)
5. attribute#1=2 attribute#2=2 15 ==> class=1 15    conf:(1)
14. attribute#1=1 attribute#2=1 9 ==> class=1 9    conf:(1)
```

Explanation of classification

This tell us that (1) whenever attribute5 is classified as 1 or (2) attribute1 and attribute2 are classified as 3 or (3) attribute1 and attribute2 are classified as 2 or (4) attribute1 and attribute2 are classified as 1, the classication for the data will be 1. I can simply it even more to 2 rules: (1) whenever attribute5 is classified as 1 or (2) whenever attribute1 and attribute2 are classified equally the the same number.

Conclusion

As previously said, trying to predict clusters using distances to create the centroid does not give a good classification, because the classification of the data is not based on distances (discrete data), but on categorical data. Therefore, the algorithms used are not good for our case, unless this data is treated somehow such that the algorithm does not look for the centroid in the way the kmeans or the makedensitybasedclustered is used but taking into account how cluster from categorical data. For this reason, I will say that the clustering algorithms fail or perform poorly for the monk1 dataset.