Lab Report: Lab3 - Association Analysis -2

Udaya Shanker Mohanan Nair(udamo524), Dhanush Kumar Reddy Narayana Reddy(dhana004)

2025-04-07

Introduction

In this experiment, we explored how different clustering algorithms identify structure within the **Monk1 dataset**, and why the clusters found may not align with the true class labels. We used both clustering and association rule analysis to examine the underlying logic of the data.

SimpleKMeans

SimpleKMeans in **Weka** is an implementation of the classic **K-Means** clustering algorithm. It partitions the data set into k clusters by minimizing the distance between data points and their assigned cluster centroids. It works best with numerical data and assumes that clusters are spherical and evenly sized.

SimpleKMeans with 2 Clusters

The **SimpleKMeans algorithm** was applied to the **Monk1 dataset** with **k = 2** clusters using the Euclidean distance metric. The algorithm converged after just 3 iterations, producing a within-cluster sum of squared errors of *358.0*, indicating the compactness of clusters. The clustering process was efficient, taking only *0.01 seconds* to complete. The resulting clusters contained *77 instances (62%)* in **Cluster 0,** with *47 instances (38%)* in **Cluster 1**. Analyzing the cluster centroids, we observe that Cluster 0 typically has values like **attributes#1 = 1** and **attributes#2 = 2**, while Cluster 1 differs with features such as **attributes#1 = 3**, **attributes #3 = 2**, and **attributes#6 = 1**. These centroids reflect the average attribute values within each cluster, highlighting how the algorithm grouped instances based on numerical similarity.

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     monk1-weka.filters.unsupervised.attribute.Remove-R7
Instances:    124
Attributes:   6
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Test mode:evaluate on training data

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

```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 358.0
Missing values globally replaced with mean/mode

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%)
```

SimpleKMeans with 3 Clusters

The **SimpleKmeans algorithm** was executed with **k = 3** on the **Monk1 dataset** using Euclidean distance. It converged in just 3 iterations and achieved a within-cluster sum of squared errors of *314.0*, which is slightly lower than the 2-cluster (358.0), indicating a tighter clustering. The instance were divided into **Cluster 0:59(48%)**, **Cluster 1:38(31%)**, and **Cluster 2:27(22%)**. From the cluster centroids, we observe that each cluster emphasizes different combinations of attribute values- for example, *attributes#1 = 1*, in **Cluster 0** vs *attributes #1 = 3* in **Cluster 1** - highlighting how the algorithm groups instances by numerical similarities. However, since clustering is unsupervised, these clusters don't necessarily align with the actual class labels of the data.

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     monk1-weka.filters.unsupervised.attribute.Remove-R7
Instances:    124
Attributes:   6
              attribute#1
              attribute#2
              attribute#3
              attribute#4
```

```
              attribute#5
              attribute#6
Test mode:evaluate on training data

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


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 314.0
Missing values globally replaced with mean/mode

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 seconds

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

Clustered Instances

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

EM

EM with 2 Clusters

The **Expectation-Maximization(EM)** algorithm was applied to the **Monk1 dataset** with the number of clusters set to 2. The model was trained on *124 instances with 6 attributes*, using a probabilistic approach to assign instances based on likelihood. It completed in *0.04 seconds* with a final *log likelihood of -6.00606*, indicating a moderate fit. The instances were nearly evenly distributed between *Cluster 0 (48%)* and *Cluster 1 (52%)*. Cluster characteristics reveal notable differences in attribute distributions. For instance, **Cluster 0** had higher probabilities fro *attributes#1 = 1* and *attributes#2 = 3*, while **Cluster 1** leaned more toward *attributes#1 = 2* and *attributes#2 = 1*. Similarly, *attribute#6 = 1* appeared more frequently in **Cluster 1**. Unlike KMeans, EM provides soft clustering, reflecting the underlying statistical patterns in the data rather than being rigid. partitions.

```
=== Run information ===

Scheme:weka.clusterers.EM -I 100 -N 2 -M 1.0E-6 -S 100
Relation:     monk1-weka.filters.unsupervised.attribute.Remove-R7
Instances:    124
Attributes:   6
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Test mode:evaluate on training data

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


EM
==

Number of clusters: 2

              Cluster
Attribute           0       1
                (0.5)   (0.5)
==============================
attribute#1
  1             29.4776 17.5224
  2              11.643  32.357
  3             24.2423 14.7577
  [total]       65.3629 64.6371
attribute#2
  1              10.008  26.992
  2             19.4842 24.5158
  3             35.8707 13.1293
  [total]       65.3629 64.6371
attribute#3
  1             31.1592 35.8408
  2             33.2037 27.7963
  [total]       64.3629 63.6371
attribute#4
  1             17.2532 26.7468
  2             16.5682 24.4318
  3             31.5415 13.4585
  [total]       65.3629 64.6371
attribute#5
  1             13.9353 17.0647
  2             19.0736 13.9264
  3             10.9633 21.0367
  4             22.3906 13.6094
  [total]       66.3629 65.6371
attribute#6
  1             25.2844 32.7156
```

```
  2            39.0785 30.9215
  [total]      64.3629 63.6371


Time taken to build model (full training data) : 0.04 seconds

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

Clustered Instances

0       59 ( 48%)
1       65 ( 52%)


Log likelihood: -6.00606
```

EM with 3 Clusters

The **EM algorithm** was executed on the **Monk1 dataset** using 3 clusters, completing the training in 0.02 seconds with a slightly improved *log likelihood of -5.96262* compared to the 2-cluster model. The 124 instances were divided into *Cluster 0 (40%)*, *Cluster 1 (17%)*, and *Cluster 2 (43%)*. Cluster 1 is significantly smaller, capturing a more specific pattern in the data. Attribute distributions reveal distinct cluster behaviors: for example, *attributes#1 = 2* is more dominant in **Cluster 2**, while *attributes#1 = 3* show up strongly in **Cluster 0**. *Attributes#5 = 1* is highly concentrated in **Cluster 1**, hinting at a unique subgroup. Meanwhile, **Cluster 2** shows high values for *attributes#2 = 1* and *attributes#5 = 3*, suggesting different attribute combinations. Compared to the 2-cluster model, this setup captures more granular structures, possibly reflecting underlying class-like separations in the Monk1 dataset more effectively through soft clustering.

```
=== Run information ===

Scheme:weka.clusterers.EM -I 100 -N 3 -M 1.0E-6 -S 100
Relation:     monk1-weka.filters.unsupervised.attribute.Remove-R7
Instances:    124
Attributes:   6
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Test mode:evaluate on training data

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


EM
==

Number of clusters: 3

              Cluster
Attribute           0       1       2
```

```
                (0.4)   (0.2)  (0.39)
======================================
attribute#1
  1            24.2508  3.6254 20.1238
  2             7.7929 10.2602  26.947
  3            20.8313 14.2613  4.9074
  [total]       52.875 28.1468 51.9782
attribute#2
  1             8.5876  8.3912 21.0211
  2            12.1224 10.3865 22.4911
  3             32.165  9.3691  8.4659
  [total]       52.875 28.1468 51.9782
attribute#3
  1            24.5556 15.7371 27.7073
  2            27.3194 11.4097 23.2709
  [total]       51.875 27.1468 50.9782
attribute#4
  1            11.8655 14.0256 19.1089
  2            14.2182  8.5015 19.2803
  3            26.7913  5.6197 13.5891
  [total]       52.875 28.1468 51.9782
attribute#5
  1             9.4906 17.4189  5.0905
  2            16.8562  4.5329 12.6109
  3             8.7413  3.5023 20.7564
  4            18.7869  3.6927 14.5204
  [total]       53.875 29.1468 52.9782
attribute#6
  1            17.9585 16.5912 24.4502
  2            33.9165 10.5555  26.528
  [total]       51.875 27.1468 50.9782


Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0       50 ( 40%)
1       21 ( 17%)
2       53 ( 43%)


Log likelihood: -5.96262
```

Association Analysis

Here, we applied the Aprior algorithm in Weka with settings:

1. Minimum support: 0.05 (at least 6 instances)
2. Minimum confidence: 0.9

3. Maximum number of rules: 19
4. Class attributes: class(set using -c last)

This setup generated 19 rules, many of which perfectly predicted the *class=1* label.

Among the 19 rules, we identified 4 non-redundant rules that together perfectly describe instance of *class = 1*:

1. attribute#5 = 1 => class = 1 (29 instances, confidence: 1)
2. attribute#1 = 3 ^ attributes#2 = 3 => class = 1 (17 instances, confidence: 1)
3. attribute#1 = 2 ^ attributes#2 = 2 => class = 1 (15 instances, confidence: 1)
4. attribute#1 = 1 ^ attributes#2 = 1 => class = 1 (9 instances, confidence: 1)

```
=== 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
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)
```

```
 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)
```

Questions

Does Clustering perform poorly on MONK1? Why or why not?

Yes, clustering performs poorly on the MONK1 data set in terms of recovering the original class labels. This is not because clustering algorithms are bad, but because:

1. They are unsupervised and do not have access to class labels.
2. The data set is better understood through rule-based (symbolic) relationships, not geometric ones.
3. Only association rule learning of supervised learning can uncover the exact logic behind the class label assignment.