CMPT 459 Fall 2020
Data Mining
Martin Ester
TAs: Madana Krishnan Vadakandara Krishnan
and Rhea Rodrigues

Assignment 2

Total Marks: 100

Assignment 2.1 (55 marks)

We want to cluster categorical data, i.e. data that have categorical attribute domains. The task of this assignment is to develop the equivalent of the *k* -means algorithm for categorical data, called the *k-modes* algorithm. We assume the following distance function (Hamming distance) for pairs of categorical objects:

$$dist(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i) \;\; with \;\; \delta(x_i, y_i) = \begin{cases} 0 \;\; if \;\; x_i = y_i \\ 1 \;\; else \end{cases}$$

Instead of the means, we choose the mode *m* as the representative of a cluster *C* . In every attribute, the mode takes the attribute value that is most frequent within the cluster. More formally, the mode *m* of cluster *C* is defined as follows:

$$m = (m_1, ..., m_d) \;\; \text{such that} \;\; \forall\, j, 1 \le j \le d, \; \forall a \in A_j : |\{x \in C \,|\, x_j = m_j\}| \ge |\{x \in C \,|\, x_j = a\}|$$

a) (30 marks)

Prove that *m* is the object minimizing the cluster cost

$$TD(C,m) = \sum_{p \in C} dist(p,m)$$

Hint: first formulate the intuition of the proof, then formalize it. The proof can be performed by contradiction.

Assignment 2.1

a) A mode of $m = \{m_1, m_2, m_3, \dots m_n\}$ is a vector/cluster $C = \{p_1, p_2, p_3 \dots p_n\}$ that minimises

$$TD(C, m) = \sum_{p \in C} dist(p, m)$$

$$\Rightarrow TD(C, m) = \sum_{\substack{i=1, \\ p \in C}}^{\wedge} dist(p, m)$$

here C is not neccessarily an element of m.

Let $n_{C_{k,j}}$ be the number of objects having the k^{th} category $C_{k,i}$ in attribute A_j and $fr(A_j = C_{k,j} \mid m) = \frac{n_{C_{k,i}}}{n}$ the relative frequency of category $C_{k,j}$ in m_i and n is the total number of objects in m. (Here, A_j refers to each attribute A_j describes a domain of values associated with a defined semantic & data type)

For the dissimilarity measure:

$$dist(x, y) = \sum_{i=1}^{\wedge} \delta(x_i, y_i) \text{ with } \delta(x_i, y_i) = \begin{cases} 0 \text{ if } x_i = y_i \\ 1, \text{ else} \end{cases}$$

$$\Rightarrow \sum_{\substack{i=1, \\ p \in C}}^{\wedge} dist(p, m_i) = \sum_{i=1}^{n} \delta(n_i, p_i)$$

$$\Rightarrow \sum_{\substack{i=1 \\ p \in C}}^{\wedge} dist(p, m_i) = \sum_{i=1}^{\wedge} \sum_{j=1}^{l} \delta(n_{i,j}, p_j)$$

$$= \sum_{j=1}^{l} \left(\sum_{i=1}^{n} \delta(m_{i,j}, p_i) \right)$$

$$= \sum_{i=1}^{d} n\left(1 - \frac{n_{p_j}}{n}\right)$$

$$= \sum_{i=1}^{d} n\left(1 - fr(A_j = p_j \mid m)\right)$$

Because $n(1 - fr(A_j = p_j \mid m)) \geq 0$ for $1 \leq j \leq n$,

$\sum_{i=1}^{n} \sum_{p \in C} dist(p, m_i)$ is minimised iff every $n(1 - fr(A_j = p_j \mid m))$ is minimal. Thus $fr(A_j = p_i \mid m)$ must be maximal.

This defines a way to find C from a given m as it allows k-means paradigm to be used to cluster categorical data. This implies that mode of a dataset X is not unique.

b) (25 marks)

Provide a pseudo-code for the *k* -modes algorithm with the following header:

k-modes(dataset, numberOfClusters)

To initialize the *k* cluster representatives, take a random sample of *k* objects from the dataset.

```
K-modes clustering - Categorical data

Code outline:
Algorithm: k-modes(dataset, numOfClusters)
Input: Data, k
Output: Cj where 1 <= j <= k

Step 1: Randomly select the K initial cluster centers such that Cj, j = 1,2,...,K
Step 2: Find the matching dissimilarity between the each K initial cluster modes and each data objects using the minizing cost function TD(C,m)
Step 3: Evaluate the fitness using the dissimilarity distance function (Hamming distance)
Step 4: Find the minimum mode values in each data object i.e. finding the objects nearest to the initial cluster modes.
Step 5: Assign the data objects to the nearest cluster modes.
Step 6: Update the modes by applying the frequency based method on newly formed clusters
Step 7: Recalculate the similarity between the objects and the updated modes
Step 8: Repeat step 4 - 5 until no changes in the cluster ship of the objects
```

```
Psuedocode:

key:
k = numOfClusters
X = data

Algorithm: get_distance(x, c)
Input: numOfClusters, sizeOfDataset
Output: distance

dist <- 0
IF x is not equal to c do
    dist += 1


Algorithm: kmodes(dataset, numOfClusters)
Input: X, k
Output: Cj where 1 <= j <= k

C1 <- empty array
Randomly select k cluster centres from X with n objects
C <- random k clusters
FOR i from 0 to range of n do
    minDist <- intialize with some large value
    FOR j from 0 to k do
        Calculate the distance between ith data point and jth mode vector using dissimilarity distance function and assign that data point to appropriate cluster whose cluster mode vector is closer
        to it and update mode vector of corresponding cluster and also find the
        distribution of mode categories between clusters

        dist <- get_distance(C[row index, features], X[row index, features])
        IF dist < minDist do
            minDist <- dist
            clustNum <- j
            X[i,d] <- clustNum
            C[j,d] <- clustNum
        ENDIF
    ENDFOR
    FOR j from 0 to k do
        result <- indexes where X[:d] == j
        mode_info <- array of modal values
        C[j] <-  reshape modal array
    ENDFOR
ENDFOR

WHILE c1 is not equal to c do
    c1 <- c
    FOR i from 0 to range of n do
    minDist <- intialize with some large value
        FOR j from 0 to k do
            Calculate the distance between ith data point and jth mode vector using dissimilarity distance function (6) and assign that data point to appropriate cluster whose cluster mode vector
            is closer to it and update mode vector of corresponding cluster and also find the distribution of mode categories between clusters

            dist <- get_distance(C[row index, features], X[row index, features])
            IF dist < minDist do
                minDist <- dist
                clustNum <- j
                X[i,d] <- clustNum
                C[j,d] <- clustNum
            ENDIF
        ENDFOR
        FOR j from 0 to k do
            result <- indexes where X[:d] == j
            mode_info <- array of modal values
            C[j] <-  reshape modal array
        ENDFOR
    ENDFOR
    IF old_modes is equal to new_modes
        then break the loop
    ENDIF
```

Assignment 2.2 (45 marks)

Implement the K-modes algorithm in python according to your pseudo-code.

Your program will accept two inputs: the dataset and numberOfClusters.

As output, write a new csv file in which every row consists of the example and the corresponding cluster number which was assigned to it by your algorithm.

You will be using the Mushroom Data set which you can download her e. Read the Dat a Set Description found on the website to understand the data. Note that the first column is the class label, followed by 22 columns of categorical attributes.

You must handle missing values in the dataset. Mention how you handled the missing values in your PDF Report (see submission details below).

Ans 2.2:

To handle missing data:

1. First we check for null or NaN values in the data. On the first glance it doesn't show anything, so after reading the dataset description we find that only column 11 has missing values in form of '?'
2. To start imputation, we first replace all '?' with NaN attributes so we can identify them easily.
3. Create a new column and replace 1 if the category is NAN else 0. This column is an importance column to the imputed category as with this we can keep track of what values were imputed.
4. We replace NaN attributes with mode of the column as this is the easiest way to impute the data

With this method we capture the importance of missingness in the data. Although if the missing data is too large, there might be a potential to misunderstand the data. This method comes with curse of dimensionality; if 10 columns have missing data, we'll have to create 10 additional columns to capture them hence if the number with missing data increase, then the number of columns increases to capture the missing data.

Follow these instructions for your implementation:

- Prepare the data
  - Handling the missing values (5 marks)
  - Brief description of the above in the report (5 marks)
- Initialization of centroids (5 marks)
- Assignment of observations to a cluster (5 marks)
- Compute new centroids (10 marks)
- Terminating criteria for iterations (10 marks)
- Saving results to a file (5 marks)

You can use libraries including math,numpy, scipy, random, etc. Do not use the scikit-learn libraries. Implement your own logic to compute the distance between observations. You will be marked on the correctness of your implementation.

Submission details

1. Submit a PDF Report for the solutions to Assignment 2.1. In your report, for Assignment 2.2, briefly explain how you decided to handle the missing data and why you handled it in this way.
2. For Assignment 2.2 submit a python program that reads the data set as input (assume the data is in the same directory as the program).

Psuedocode if you arrent able to read from the photo!

Psuedocode:

key:
k = numOfClusters
X = data

Algorithm: get_distance(x, c)
Input: numOfClusters, sizeOfDataset
Output: distance

dist <- 0
IF x is not equal to c do
dist += 1

Algorithm: kmodes(dataset, numOfClusters)
Input: X, k
Output: Cj where 1 <= j <= k

C1 <- empty array
Randomly select k cluster centres from X with n objects
C <- random k clusters
FOR i from 0 to range of n do
minDist <- intialize with some large value
FOR j from 0 to k do
Calculate the distance between ith data point and jth mode vector using dissimilarity distance function and assign that data point to appropriate cluster whose cluster mode vector is closer to it and update mode vector of corresponding cluster and also find the
distribution of mode categories between clusters

dist <- get_distance(C[row index, features], X[row index, features])
IF dist < minDist do
minDist <- dist
clustNum <- j
X[i,d] <- clustNum
C[j,d] <- clustNum
ENDIF
ENDFOR
FOR j from 0 to k do

```
        result <- indexes where X[:d] == j
      mode_info <- array of modal values
      C[j] <-  reshape modal array
   ENDFOR
ENDFOR

WHILE c1 is not equal to c do
c1 <- c
FOR i from 0 to range of n do
minDist <- intialize with some large value
   FOR j from 0 to k do
```

Calculate the distance between ith data point and jth mode vector using dissimilarity distance function (6) and assign that data point to appropriate cluster whose cluster mode vector is closer to it and update mode vector of corresponding cluster and also find the distribution of mode categories between clusters

```
                dist <- get_distance(C[row index, features], X[row index, features])
      IF dist < minDist do
         minDist <- dist
         clustNum <- j
         X[i,d] <- clustNum
         C[j,d] <- clustNum
      ENDIF
          ENDFOR
          FOR j from 0 to k do
          result <- indexes where X[:d] == j
      mode_info <- array of modal values
      C[j] <-  reshape modal array
          ENDFOR
ENDFOR
IF old_modes is equal to new_modes
          then break the loop
          ENDIF
ENDWHILE
```