Data Mining report

Implement the robust K-median algorithm for incomplete data

Student: Duy Dien Nguyen

Professor: Massimo Cafaro

1. Introduction

In the fields of statistics, data mining, and machine learning, it is common for the datasets under consideration to contain multiple observations with missing feature values. Such incomplete data occur in a variety of application domains due to various reasons, such as improper collection of datasets, high cost of obtaining some feature values, and missing answers in the questionnaire. For example, in online shopping, users usually rate only a small fraction of the available items, resulting in a large amount of missing feature values [1].

Clustering analysis is considered an effective method to extract useful features and explore potential data patterns. Due to missing feature values, there is an urgent need to cluster incomplete data in many fields such as market research, information retrieval, image processing, and clinical medicine. The basic approach to clustering incomplete data is the two-step method: First estimates the missing feature values by imputation. Then applies the clustering methods for complete data: Hierachical algorithms, Partioning algorithms,…

One of the problems with traditional imputation methods such as direct imputation and iterative imputation is the assumption that the missing feature value can be well estimated by a single value. It is usually difficult to obtain accurate estimates of the missing values, so clustering methods based on imputation are sensitive to the accuracy of the estimate. To address this problem, many techniques for missing values estimation and clustering incomplete data are mentioned in [1]. One of them is the use of nearest neighbor intervals to represent the missing values and extend FCM by defining a new interval distance function for interval data in [2]. It has been shown that interval data is an effective way to deal with missing values, and it has also been used to propose effective clustering methods.

Based on the interval data representation, the authors present robust K-median and K-means clustering algorithms in [1], the improved interval construction method based on pre-classification is used to obtain the interval data for missing values. Unlike the existing algorithms that use either the interval distance function or optimal imputation, the authors reformulate the clustering problem as a minimax robust optimization problem based on interval data.

This project aims to explore and implement the robust K-median algorithm for incomplete data proposed by the authors in [1]. The algorithm, the implementation process, and the results will be discussed in the next sections.

2. Robust K-median algorithm

Consider the problem of clustering a set of n objects $I = \{1, \ldots, n\}$ into K clusters. For each object $i \in I$, we have a set of m features $\{x_{ij} : j \in J\}$, where x_{ij} describes the jth features of the object i quantitatively. Let $x_i = (x_{i1}, \ldots, x_{im})^T$ be the feature vector of the object i and $X = (x_1, \ldots, x_n)$ be the feature matrix or dataset.

We refer to a data set X as an incomplete dataset if it contains at least one missing feature value for some objects. That is, there exists at least one $i \in I$ and $j \in J$, such that x_{ij} = ? (or null). To describe the missing dataset, for any $i \in I$, the authors partition the feature set of i into two subsets:

$$J_i^0 = \{j : x_{ij} = ?, \forall j \in J\}, J_i^1 = J \setminus J_i^0$$

As mentioned before, in practice, it is difficult to obtain accurate estimations of missing feature values. Thus, the authors represent missing values by intervals. Specifically, for any $i \in I$, they use an interval $[(1-\theta)x_{ij}^-, (1+\theta)x_{ij}^+]$ to represent unknown missing feature value where $j \in J_i^0$ and use $\overline{x_{ij}}$ to represent known feature value where $j \in J_i^1$. To simplify notations, the authors use $\overline{x_{ij}} = ((1-\theta)x_{ij}^- + (1+\theta)x_{ij}^+)/2$ and $\delta_{ij} = ((1+\theta)x_{ij}^+ - (1-\theta)x_{ij}^-)/2$ for any $j \in J_i^0$ and $\delta_{ij} = 0$ for any $j \in J_i^1$.

Then, the authors defined the robust cluster objective function:

$$\begin{aligned} J^R(U,V) &= \sum_{i\in I}\sum_{j\in J} \max\left\{\sum_{k=1}^{K} u_{ik}\left|\overline{x}_{ij} + y_{ij} - v_{kj}\right| : -\delta_{ij} \le y_{ij} \le \delta_{ij}\right\} \\ &= \sum_{i\in I}\sum_{j\in J} \max\left\{\sum_{k=1}^{K} u_{ik}\left|\overline{x}_{ij} - \delta_{ij} - v_{kj}\right|, \sum_{k=1}^{K} u_{ik}\left|\overline{x}_{ij} + \delta_{ij} - v_{kj}\right|\right\} \\ &= \sum_{i\in I}\sum_{j\in J}\sum_{k=1}^{K} u_{ik} \max\left\{\left|\overline{x}_{ij} - \delta_{ij} - v_{kj}\right|, \left|\overline{x}_{ij} + \delta_{ij} - v_{kj}\right|\right\}, \end{aligned}$$

For $k = 1, \ldots, K$, $V_k \in R^m$ is the kth cluster prototypes and, for any $i \in I$, u_{ik} indicates whether the object i belongs to the kth cluster. Let the cluster prototype matrix $V = [V_1, \ldots, V_K] \in R^{m\times K}$ and the membership matrix $U = [u_1, \ldots, u_n] \in R^{K\times n}$, where $V_i = (V_{i1}, \ldots, V_{im})^T$ and $u_i = (u_{i1}, \ldots, u_{iK})^T$.

Since $\max\{|x - y|, |x + y|\} = |x| + |y|$ and $\delta_{ij} = 0$, for any $i \in I$ and $j \in J_i^1$, we further have:

$$J^R(U,V) = \sum_{k=1}^{K}\sum_{i\in I}\sum_{j\in J} u_{ik}\left|\overline{x}_{ij} - v_{kj}\right| + \sum_{k=1}^{K}\sum_{i\in I}\sum_{j\in J_i^0} u_{ik}\delta_{ij}$$

The above equation shows that the existence of missing values increases the objective function of the cluster. Based on this, the robust K-median clustering algorithm can be described as follows:

Robust K-median algorithm

Input: The feature matrix $\overline{X}$, interval size δ_{ij} ($i \in I, j \in J$) and number of cluster K.

Output: The cluster prototype matrix V^* and membership matrix U^*.

Step 1 (initialization): Set iteration index $t = 0$ and randomly select K different rows from $\overline{X}$ as the initial cluster prototypes $\{v_k^t : k = 1, \ldots, K\}$.

Step 2 (Assign point to its nearest centroid): Let $t = t + 1$ and update U^t by fixing V^{t-1}.

For any $i \in I$, randomly select $k^* \in \arg\min\{\sum_{j\in J} |\overline{x}_{ij} - v_{kj}^{t-1}| + \sum_{j\in Ji0} \delta_{ij} : k = 1, \ldots, K\}$, and set $u_{ik*}^t = 1$ and, for any $k \neq k^*$, set $u_{ik}^t \ t = 0$.

Step 3 (Update centroid): Update V^t by fixing U^t:

For any $k = 1, \ldots, K$, let $I_k = \{i \in I : u_{ik}^t = 1\}$. For any $j \in J$, set v_{kj}^t as the median of $\{\overline{x}_{ij} : i \in I_k\}$.

Step 4 (Stop criterion): If $u_{ik}^t = u_{ik}^{t-1}$ for any $i \in I$ and $k = 1, \ldots, K$, then stop and return to $U^* = U^t$ and $V^* = V^t$; otherwise, go to Step 2.

3. Implementation

Step 1: Generate missing values for dataset.

In this step, I randomly remove the value of some features of the data from the original data set to obtain a new data set with missing values. The function receives as input the original data set, the percentage of the data whose feature value is to be removed, and the output file. To make the incomplete data manageable, we have to ensure that the following conditions are met [1]:

(1) Each object retains at least one feature.

(2) Each feature has at least one value that is present in the incomplete data set.

```
for index, item in enumerate(data):
  number_of_remove_index = random.randint(1,2)
  index_to_remove = random.sample(range(0, 4), number_of_remove_index)
  newDatum = np.zeros(dimensions)
  if index in datum_index:
    for dim in range(dimensions):
      if dim in index_to_remove:
        newDatum[dim] = float(np.nan)
      else:
        newDatum[dim] = item[dim]
  else:
    newDatum = item
  writer.writerow(newDatum)
```

Fig. 3.1. Randomly remove feature value of data

Step 2: Use the KNN algorithm to find k nearest neighbors of the points containing missing attribute values, and then find the interval for those missing attribute values. The range of values of the missing attribute can be determined by the minimum and maximum of the corresponding attribute values of the neighbors.

```
# Get the most similar (nearest) neighbors
def get_neighbors(data, test_row, num_neighbors):
  distances = list()
  for train_row in data:
    dist = euclidean_distance(test_row, train_row)
    distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
  neighbors = list()
  for i in range(num_neighbors):
    neighbors.append(distances[i+1][0]) #use i+1 to skip itself
  return neighbors
```

Fig. 3.2. Get k nearest neighbors of a point

After that, we obtain new feature matrix and delta value and save as 2 new data files based on the formula that is presented in section 2.2.

```
for neighbor in neighbors:
  if neighbor[dim] > max_val:
    max_val = neighbor[dim]
  if neighbor[dim] < min_val:
    min_val = neighbor[dim]

value = "{:.1f}".format((max_val*(1+theta) + min_val*(1-theta))/2)
delta = "{:.1f}".format((max_val*(1+theta) - min_val*(1-theta))/2)
```

Fig. 3.3. Find interval and return new estimated value and delta for each missing feature

Step 3: Apply robust K-median clustering algorithm in the section 2.2 for new dataset.

The robust K-median algorithm reads 2 new files (feature matrix and delta value) that generated in the previous step and k (number of clusters) as input. The output is 2 new files: the cluster ID predicted for each point and the final position of the centroid of each cluster.

```
PS C:\Users\dienn\OneDrive\Desktop\Robust-Kmedian> ./robust_kmedian Co
mpleteData_miss=20_theta=0.1.csv Delta_miss=20_theta=0.1.csv 3 results
Clustering completed in iteration : 12

Cluster 1 centroid : 5.82985 2.70896 4.35522 1.46567
Cluster 2 centroid : 4.99388 3.40816 1.47347 0.320408
Cluster 3 centroid : 6.84118 3.08235 5.7 2.08235
```

Fig 3.4. Running an experiment

4. Experimental results

In this section, I will run and compare the robust K-Median on the Iris dataset with different values of theta and percentage of missing feature values.

4.1. Dataset and Experimental Setup

The Iris dataset is one of the most widely used datasets for performance testing of machine learning algorithms. The dataset consists of 150 objects and each object has four features of Iris flowers, including sepal length, sepal width, petal length, and petal width. The Iris data includes three clusters, Setosa, Versicolour, and Virginica, and each cluster contains 50 objects.

First, I generate new datasets with missing values by randomly selecting a certain percentage (0%, 10%, 20% and 40%) of the data points and remove some of their attributes.

1	sepal.length	sepal.width	petal.length	petal.width
2	5.1	3.5	1.4	0.2
3	4.9	3	1.4	0.2
4	4.7	3.2	1.3	0.2
5	4.6	3.1	1.5	0.2
6	5	3.6	1.4	0.2
7	5.4	3.9	1.7	nan
8	nan	3.4	1.4	0.3
9	5	3.4	1.5	0.2
10	nan	2.9	nan	0.2

Fig. 4.1. Example of generated Iris dataset with 20% of rows contains missing value

Second, according to the numerical experiments in [1], k = 6 is a good choice. Then I construct the interval and then estimate the value and the corresponding delta value for the missing value, estimating from the 6 nearest neighbors as the formula in Section 2.

1	sepal.length	sepal.width	petal.length	petal.width
2	5.1	3.5	1.4	0.2
3	4.9	3	1.4	0.2
4	4.7	3.2	1.3	0.2
5	4.6	3.1	1.5	0.2
6	5	3.6	1.4	0.2
7	5.4	3.9	1.7	0.3
8	2.9	3.4	1.4	0.3
9	5	3.4	1.5	0.2
10	5	2.9	1.5	0.2

1	sepal.length	sepal.width	petal.length	petal.width
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0.1
8	2.9	0	0	0
9	0	0	0	0
10	1	0	0.3	0

Fig. 4.2. Example of completed Iris dataset and corresponding delta after estimated.

4.2. Result and Discussion

In this section, I will test and compare the performance of the implemented robust K-median on the Iris data set under different missing rates from 0% to 40%.

Tables 1 report the averaged performance of robust K-median algorithm on the incomplete Iris data. The first column in the table gives the missing rate. The second to fifth columns give the averaged misclassification rates by comparison with the true clustering result with different values of θ ranging from 0.05 to 0.15.

Table 1. Performance of Robust K-median algorithm on the IRIS data

	Misclassification rate (%)			
Theta values	0	0.05	0.1	0.15
0	11.3	11.3	11.3	11.3
10	12.6	12.6	12.6	13.3
20	14	14	14.6	16
40	15.3	15.3	18.6	18

The table 2 gives the mean prototype error of the implemented algorithm, which is calculated by:

$$\|V^* - \tilde{V}\|_2 = \sum_{k=1}^{K} \sum_{j \in J} |v_{kj}^* - \tilde{v}_{kj}|_2$$

where V^* represents the cluster prototypes given by robust K-median algorithm and $\tilde{V}$ is the actual cluster prototypes of the Iris data set without missing values that present in [2].

Table 2. Mean prototype error

	Mean prototype error			
Theta values	0	0.05	0.1	0.15
0	0.015	0.015	0.015	0.015
10	0.017	0.016	0.015	0.010
20	0.032	0.022	0.015	0.015
40	0.063	0.058	0.050	0.046

From Tables 1 and 2, we have the following observations:

- When there is no missing value, that is, the missing rate is equal to zero, the robust K-median algorithm give the same results. As the missing rate increases, in most cases, both the misclassification rate and mean prototype error also increase.
- The experimental results also show that the interval size affects the performance of the robust K-median algorithm. When the value of θ increases from 0.05 to 0.15, the misclassification rate increases in most cases, but the mean prototype error decreases.

References

[1] Jinhua Li, Shiji Song, Yuli Zhang, Zhen Zhou, "Robust K-Median and K-Means Clustering Algorithms for Incomplete Data", Mathematical Problems in Engineering, vol. 2016, Article ID 4321928, 8 pages, 2016. https://doi.org/10.1155/2016/4321928

[2] D. Li, H. Gu, and L. Zhang, “A fuzzy c-means clustering algorithm based on nearest neighbor intervals for incomplete data,” Expert Systems with Applications, vol. 37, no. 10, pp. 6942–6947, 2010. https://doi.org/10.1016/j.eswa.2010.03.028