International Journal of Database Theory and Application
Vol.9, No.9 (2016), pp.167-176
http://dx.doi.org/10.14257/ijdta.2016.9.9.16

An Extended K-Means Algorithm using MapReduce Framework for Mixed Datasets

Anupama Chadha[*] and Suresh Kumar[**]

[*]*Faculty of Computer Applications, MRIU, Faridabad, India*
anupamaluthra@gmail.com
[**]*Faculty of Engineering and Technology, MRIU, Faridabad, India*
suresh.fet@mriu.edu.in

Abstract

K-Means is a famous partition based clustering algorithm. Various extensions of K-Means have been proposed depending on the type of datasets being handled. Popular ones include K-Modes for categorical data and K-Prototype for mixed numerical and categorical data. The K-Means and its extensions suffer from one major limitation that is dependency on prior input of number of clusters K. Sometimes it becomes practically impossible to correctly estimate the optimum number of clusters in advance. Various ways have been suggested in literature to overcome this limitation for numerical data. But for categorical and mixed data work is still in progress. In this paper, we introduce a new algorithm based on the K-Means that takes mixed dataset as an input and generates appropriate number of clusters on the run using MapReduce programming style. The new algorithm not only overcomes the limitation of providing the value of K initially but also reduces the computation time using MapReduce framework.

Keywords: *Clustering, K-Means, Mixed dataset, Generating clusters on the run, MapReduce framework*

1. Introduction

Clustering is a technique of dividing the given dataset into groups or clusters such that the objects in one group are more similar to each other than the objects in the other group. Earlier, clustering techniques were developed focusing on a single type of attributes, either numerical or categorical. Since mixed dataset is common in real life, so techniques need to be developed to group this type of data. The techniques used only for numerical data or for categorical data cannot be directly applied to cluster mixed data as they differ in their behavior. The numerical data is continuous whereas categorical data is discontinuous and disordered.

K-Prototype is a variant of K-Means that can be used with numeric or categorical datasets. K-Prototype extends the idea of K-Means by applying Euclidean distance to numeric attributes and Binary distance to categorical attributes. However, the Binary distance for categorical attributes does not represent the real situation as the categorical values may have some other degree of difference rather than just 0 or 1. So, various extensions of the K-Prototype have been investigated. All these extensions suffer from the limitation of inputting the number of clusters required. Attempts have been made in the literature to deal with the limitation of providing number of clusters required as an input in K-Prototype algorithm.

Cheung *et al*. [4] presented a similarity metric that can be applied to categorical, numerical, and mixed attributes. Based on this similarity metric an iterative clustering algorithm is developed to overcome the limitation of inputting K. The limitation of this approach is that it requires some initial value of K which should not be less than the original value of K.

Liang *et al.* [3] extended K-Prototype algorithm by proposing a generalized mechanism for characterizing within-cluster entropy and between-cluster entropy to identify the worst cluster in a mixed dataset, an effective cluster validity index to evaluate the clustering results and the K-Prototype algorithm with a new dissimilarity measure. Experimental results on both synthetic and real data with mixed attributes show that the proposed algorithm is superior to the other algorithms both in detecting the number of clusters and in obtaining better clustering results. The limitation of the algorithm is that it requires input parameters representing the minimum and maximum number of clusters that can be generated from the dataset.

The algorithms discussed above require some initial value of K or some other parameter as input. In this paper, we extend K-Prototype algorithm to overcome the limitation of providing number of clusters required or any other parameter as an input using MapReduce framework.

In order to deal with huge datasets parallel and distributed computing models are becoming popular. MapReduce is one of these models which is becoming popular for its simplicity. As the name suggests this model uses two functions: Map and Reduce. In this model the dataset is divided into small chunks. These chunks are independently input to the various machines called Data nodes or Slave nodes. One machine sits on top of these machines to control the distribution of chunks of data. This machine is called Name node or Master node. The small chunks of data are sent to the data nodes by the master node. On these data nodes Map or Reduce functions are run. The nodes which run Map function read the data and output them in the form of (key, value) pair. The nodes which run the Reduce function take these (key, value) pairs as inputs and combine them on the basis of the key values and produce the final output and return it to the master node as shown in Figure 1.

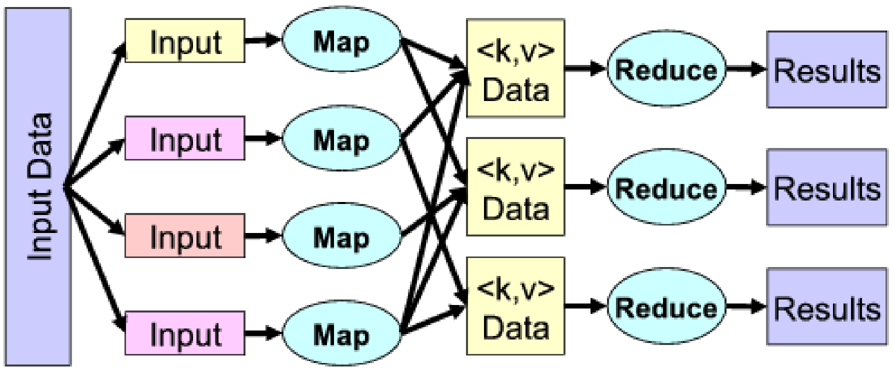


Figure 1. MapReduce Framework [2]

The work in this paper is an extension of our previous work [1]. In our previous work, we have suggested an algorithm based on the K-Means for mixed datasets which does not require K as input. In our present work we have extended our previous work by implementing that algorithm on the MapReduce framework.

2. An Extended K-Means Algorithm

In our previous work we have extended the K-Means algorithm which does not require an initial estimation and specification of number of clusters (K) as input for mixed datasets, and hence can generate much more meaningful clusters. The extended algorithm is discussed again in this section so that its implementation on MapReduce framework can be understood better.

The extended algorithm starts by dividing the given dataset into some initial clusters based on the most significant attribute in the dataset. These initial clusters are then checked for outliers based on some objective function. The set of outliers obtained are then taken as a new dataset and this process repeats until all the objects are assigned to some clusters satisfying the objective function.

The methods of finding the most significant attribute, finding the initial clusters, finding the centroids and finding the distance of an object from its centroid are discussed in detail in our previous work.

2.1. The Pseudocode of the Extended K-Means Algorithm

```
Input: Dataset (D) of n objects with m mixed attributes
Output: clusters or groups distributing the objects in the given dataset

    1. Find the most significant attribute.
    2. Create initial clusters.
    3. Find the centroids of all the clusters.
    4. Find the distance of every object from the respective centroids. Find the
       average of all the distance values for every cluster. The minimum of these
       average distance values (other than zero) is taken as dm.
    5. Find the outliers in the initial clusters according to the objective function as
       given in Eq.(1).
                    $d(X_i, C_j) <= d_m$ not an outlier      (1)
             where $d(X_i, C_j)$ is the distance between object $X_i$ and centroid $C_j$,
                    $X_i=x_{i1}, x_{i2}, \ldots x_{im}$,   $C_j=c_{i1}, c_{i2}, \ldots c_{im}$,
                 $d_m$= The minimum of all the average distances.
       The distance of every object is calculated from its centroid in every cluster.
       The average distance is calculated for each cluster.

    6. Let B={Y1, Y2,…..Yp} be the set of outliers obtained in step 5.
    7. If set B contains one object then
           Create a new cluster containing that object
       Else
           Repeat
                 a. Assume this set B as a new dataset (D)
                 b. Perform steps 1 to 7
           until (B==Φ)
```

3. An Extended K-Means Algorithm using MapReduce Framework

In Section 2, we discussed the extended K-Means algorithm which does not require the input as K. In this section the algorithm discussed in Section 2 has been modified to implement in the MapReduce framework. The algorithm is divided into three parts: part1 is the Startup algorithm, part 2 is the Mapper function and part 3 is the Reducer function.

The Startup algorithm initiates the process by dividing the dataset in various clusters depending on the most significant attribute. Also the centroids of these clusters are found by the Startup algorithm. The centroids and the objects belonging to the centroids are then sent to various Mappers. The Mappers calculate the distance of the objects from their centroids and also the average of all these distances. The centroids, objects allocated to the centroids, the distance of the objects from their centroids and the average distance from each Mapper is sent to the Reducer. The Reducer then finds out the value of d_m and the outliers in each cluster depending on the objective function mentioned in step 5 of the algorithm in Section 2.1. These outliers are then sent to the Startup algorithm. The Startup algorithm checks if there is one outlier then a new cluster is created for that outlier else the whole process is repeated taking the outliers as a new dataset.

The use of MapReduce framework will reduce the processing time considerably as the task of finding the distance of objects from their respective centroids is distributed among the various Data nodes running the Map function.

Figure 2 shows the flowchart of the extended K-Means algorithm in MapReduce framework.

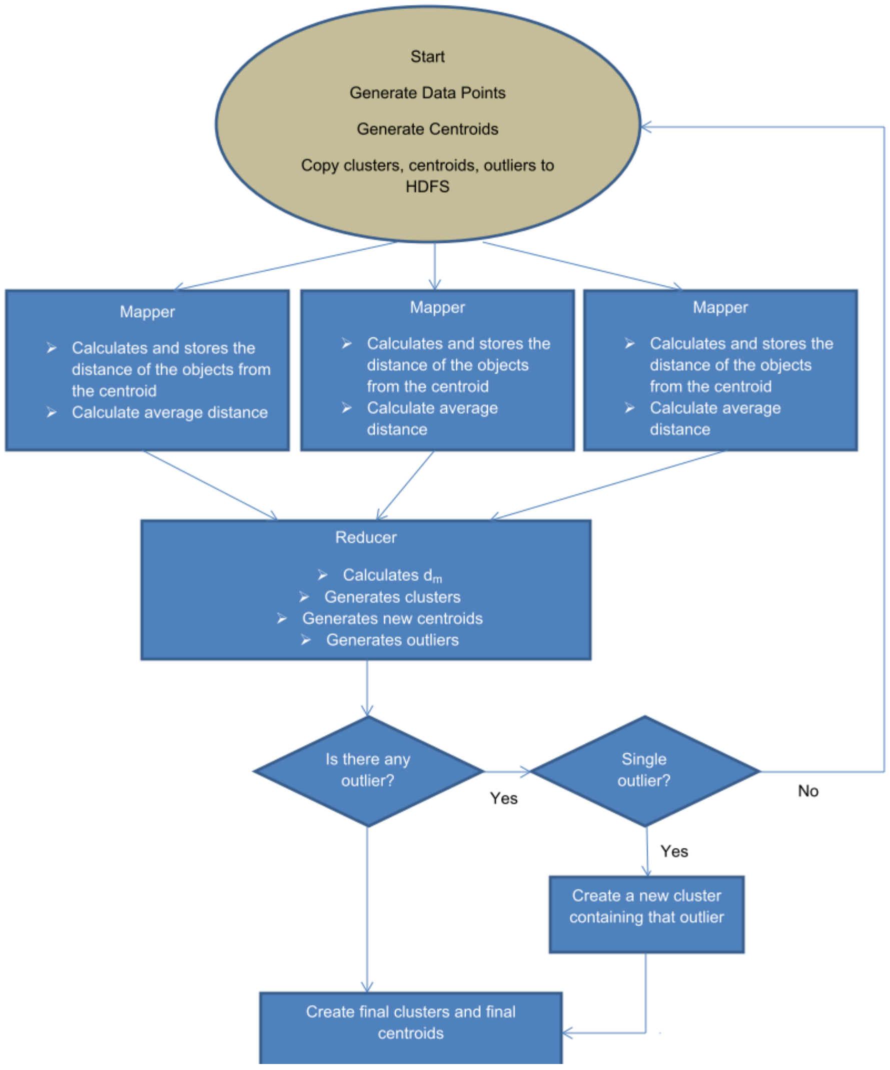

Start
Generate Data Points
Generate Centroids
Copy clusters, centroids, outliers to HDFS
Mapper
Calculates and stores the distance of the objects from the centroid
Calculate average distance
Mapper
Calculates and stores the distance of the objects from the centroid
Calculate average distance
Mapper
Calculates and stores the distance of the objects from the centroid
Calculate average distance
Reducer
Calculates d_m
Generates clusters
Generates new centroids
Generates outliers
Is there any outlier?
Yes
Single outlier?
No
Yes
Create a new cluster containing that outlier
Create final clusters and final centroids

Figure 2. Extended K-Means Algorithm in MapReduce Framework

4. Psuedocode of the Extended K-Means Algorithm using MapReduce Framework

```
Algorithm1 (Startup)

Input:
D: Dataset (D) of n objects with m mixed attributes

Output:
Final clusters with their centroids

Method:
  1.    Find the most significant attribute.
  2.    Create initial clusters.
  3.    Find the centroids of all the clusters.

  4.    Load (X, initialcentroids)
  5.    centroids, finalclusters, outliers<= Perform MapReduce

  6.    If outliers== Φ then
                End
        Else
                 If outliers==1 then

                        Create a new cluster containing that object and end
                Else
                        Repeat
                                a.      Assume this set B as a new dataset (D)
                                b.      Perform steps 1 to 6
                        Until (B==Φ)
```

```
Algorithm 2 (Mapper)
        Input:
            • A subset of m dimensional objects X={$x_1$, $x_2$, …….$x_n$} in each mapper
            • Centroids ($c_1$,$c_2$,…..$c_k$)
        Output:
        In the form of key, value where key=centroid, value=objects assigned to each
        centroid, the distance of the objects from their centroids, average distance
        Method:
        1. Find the distance of every object from the respective centroids. Find the
           average of all the distance values for every cluster.
        2. Output<< (centroids, X, the distance of the objects from their centroids,
           average distance)
```

```
Algorithm 3 (Reducer)
        Input:
                (key, value) pair, where key= centroid, value=objects assigned to each
                centroid, the distance of the objects from their centroids, average distance
        Output:
            Centroids ($c_1$, $c_2$,….$c_k$), clusters, outliers
Method:
    1.  Find the minimum of the average distances received from the Mappers. The
        minimum of these values is taken as $d_m$.
```

2. Find the outliers in the initial clusters according to the objective function as given in Eq.(1).
3. Output<< (centroids, clusters, outliers)

5. Illustrative Example

The above algorithm is explained using a dataset as shown in Table 1.

Table 1. Dataset with Mixed Attributes

Alpha	Beta	Gamma
A	C	3.2
A	C	3.0
A	D	3.2
B	D	5.0
B	C	4.2
B	D	3.5
C	D	4.8

As per our Startup algorithm:

Step 1. To find the most significant attribute, discretize attribute Gamma. After discretizing Table 1 transforms into Table 2.

Discretization is done as:

$$interval\ width = (Max\ value - Min\ value)/N \tag{2}$$

The numeric attribute Gamma will be discretized into three values ‘a’, ‘b’ and ‘c’ because the maximum number of values contained by a categorical attribute ‘Alpha’ in the dataset is three. For attribute Gamma, Taking N=3, Max value=5.0, Min value=3.

Interval width=(8.0-3.5)/3= 0.6

Table 2. Dataset after Discretizing Table 1

Alpha	Beta	Gamma
A	C	c
A	C	c
A	D	c
B	D	a
B	C	b
B	D	c
C	D	a

Calculate the distance between various values of the attributes of Table 2. Table 3 shows all the distance values.

Table 3. Distance between Various Values of Attributes of Table 2

Alpha	Beta	Gamma
d(A, B)=1/2 d(A, C)=5/6 d(B, C)=1/3	d(C,D)=11/24	d(a, b)=3/4 d(a, c)=5/8 d(b, c)=5/8

The significance values of attributes Alpha, Beta and Gamma are shown in Table 4.

Table 4. Significance of Attributes of Table 2

Alpha	Beta	Gamma
0.61	0.46	0.67

Step 2. Choosing Gamma as the most significant attribute, initial clusters are obtained as:

Cluster1	Cluster2
{(A ,C, c), (A, C, c), (A, D, c), (B, D, a), (B, D, c), (C, D, a)}	{(B, C, b)}

Create initial clusters by keeping the values 'a' and 'b' of attribute Gamma into separate clusters as the distance between them is more than the distance between 'a', 'c' and the distance between 'b', 'c'. The objects with value 'c' for attribute Gamma can be kept in the cluster with value 'a' or value 'b' because the distance between value 'a' and 'c' is same as the distance between value 'b' and 'c'.

Step 3. In order to find the centroids of the clusters, the attribute Gamma is normalized as shown in Table 5.

Table 5. Objects in Cluster1 and Cluster2 after Normalizing Attribute Gamma

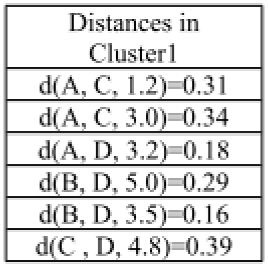
Cluster1			Cluster2		
Alpha	Beta	Gamma	Alpha	Beta	Gamma
A	C	0.1	B	C	0.45
A	C	0			
A	D	0.1			
B	D	1			
B	D	0.25			
C	D	0.9			

Now centroid of cluster1:
(1/4(A, 3B), 0.34, 0.3)
centroid of cluster2:
(1/2(2C), 0.5, 0.5), that is, (C, 0.5, 0.5)

Step 4. Load each centroid and the objects allocated to them in separate Mappers.

Step 5. Every Mapper calculates the distance of the objects from their centroid. Also it calculates the average of these distances. The distance of the objects from their respective centroids is shown in Table 6.

Table 6. Distance of the Objects from their Respective Centroids



Distances in Cluster1
d(A, C, 1.2)=0.31
d(A, C, 3.0)=0.34
d(A, D, 3.2)=0.18
d(B, D, 5.0)=0.29
d(B, D, 3.5)=0.16
d(C , D, 4.8)=0.39

The average of all the distances in cluster1 is 0.28. As cluster2 contains one object, so the minimum distance d_m=0.28.

Step 5. There are four outliers in cluster1:

(A, C, 3.2), (A, C, 3.0), (B, D, 5.0), (C, D, 4.8)

Therefore, the transformed cluster1 is:

{(A, D, 3.2), (B, D, 3.5)}

Step 6. Take B={(A, C, 3.2), (A, C, 3.0), (B, D, 5.0), (C, D, 4.8)}

Step 7. Repeat steps 1-7 on the above dataset. Finally the following four clusters are obtained.

Cluster1: {(A, D, 3.2), (B, D, 3.5)}
Cluster2: {(B, C, 4.2)}
Cluster3: {(A, C, 3.2), (A, C, 3.0)}
Cluster4: {(B, D, 5.0), (C, D, 4.8)}

The clusters obtained above are compared with the clusters obtained using K-Prototype algorithm, with K=4, using software RapidMiner as shown in Table 7.

Table 7. Results of Dataset of Table 1

	K-Prototype	**Extended Algorithm**
Cluster1	{(B, D, 3.5)}	{(A, D, 3.2), (B, D, 3.5)}
Cluster2	{(B, C, 4.2)}	{(B, C, 4.2)}
Cluster3	{(A, C, 3.2), (A, C, 3.0), (A, D, 3.2)}	{(A, C, 3.2), (A, C, 3.0)}
Cluster4	{(B, D, 5.0), (C, D, 4.8)}	{(B, D, 5.0), (C, D, 4.8)}

Table 7 shows that the extended algorithm has generated clusters by putting the most similar objects together. In the extended algorithm, the maximum attributes in which the objects match is three in cluster3, whereas in the K-Prototype algorithm the maximum attributes in which the values match is two.

6. Conclusion and Future Work

We have extended the K-Means algorithm for mixed datasets to overcome one of its major limitation of inputting the value of K. The extended algorithm uses MapReduce framework. The use of MapReduce framework has made the extended algorithm efficient in terms of time by distributing the work of distance calculation of the objects from their centroids among the data nodes on which Map function is run. In our future work we will implement this algorithm on Hadoop framework and will compare its performance to various other extensions of the K-Means algorithm for mixed datasets proposed in the literature.

References

[1] A. Chadha and S. Kumar, "Extension to the K-Means Algorithm for Automatic Generation of Clusters for Mixed Datasets", Proceedings of the International Conference on Advances on Computers, Communication and Electronic Engineering, Srinagar University, **(2015)**.
[2] http://alumni.cs.ucr.edu/~jdou/misco/figs/mapreduce.png. 08/08/2016
[3] J. Liang, X. Zhao, D. Li, F. Cao and C. Dang, "Determining the Number of Clusters using Information Entropy for Mixed Data", Pattern Recognition, vol. 45, no. 6, **(2012)**, pp. 2251–2265.
[4] Y. Cheung and H. Jia, "Categorical-and-Numerical-Attribute Data Clustering based on a Unified Similarity Metric without Knowing Cluster Number", Pattern Recognition, vol. 46, **(2013)**, pp. 2228–2238.