International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 18 (2015) pp 38880-38884

CODED EKSTRAP Clustering Algorithm

Terence Johnson
PhD scholar (Information Technology), AMET University, Chennai, India
Assistant Professor, Dept. of Computer Engineering, Agnel Institute of Technology & Design, Goa, India
ykterence@rediffmail.com

Dr. Santosh Kumar Singh
Head, Dept. of Information Technology, Thakur College of Science and Commerce
Kandivali (E), Mumbai, India, sksingh14@gmail.com

Valerie Menezes
Assistant Professor, Dept. of Computer Engineering, Agnel Institute of Technology & Design, Goa, India
vmm@aitdgoa.edu.in

Abstract
In the COsine Distance and Euclidean Distance based Enhanced K STRAnge Points (CODED EKSTRAP) clustering algorithm, an incremental strategy for cluster formation is put forward in which the minimum, maximum and K equidistant strangest values of the input set are calculated using the cosine distance measure. Once the furthest values of the input set equal to the user defined number of clusters K are found, the remaining values of the input set are then assigned into clusters formed by the K Strange input points using the Euclidean distance measure. The CODED EKSTRAP clustering algorithm is an extension of the Enhanced K Strange Points clustering algorithm and can be used in applications requiring the use of multiple distance measures based on the requirements of the operations involved.

Keywords: Enhanced k-strange points clustering, Cosine distance, Euclidean distance.

Introduction

The task of extracting knowledge from vast amounts of data is handled in Data mining [1]. The clustering technique of data mining separates data into groups such that members of a group are unique in some way and differ from members of other groups thereby forming clusters [2]. A cluster is thus, is a set of values that are very similar to each other within their own cluster and different from the values in other clusters [3]. A standard clustering algorithm will output clusters with a reasonably high intra cluster similarity as well as reasonably low inter cluster similarity [4]. The main goal of clustering is to find sets of similar data items where items are modeled as points in multidimensional space. The distance metrics are used to determine the similarity and dissimilarity between data items [5]. Clustering is applied to a wide variety of domains such as survey of markets, customer segmentation, business intelligence, decision making systems, genetics, bio-medicine, geo-spatial informatics, environmental engineering and much more [6]. Clustering is also used to reveal the implicit groupings in a structure of the input set [7].

Enhanced K Strange Points Clustering

The Enhanced K Strange points clustering algorithm [8] initially selects the first of the K strange values as the minimum of the input set and then finds the next strange value which is furthest from the minimum thus giving two values from the dataset which are at maximum distance from each other using the Euclidean distance measure. It then locates a third value which is furthest from the minimum and maximum such that the sum of the distances between these 3 values is greater than any other combination. If required, the location of the third value is corrected by placing it almost maximally and equally spaced from the minimum and maximum. This process is continued until K values equal to the number of user defined clusters are found. It then outputs K clusters by accumulating the remaining n-K values in the input set into clusters formed by these K Strange values using the Euclidean distance measure [9].

Cosine Distance and Euclidean Distance

It is necessary to have a distance measure for any clustering process in order to determine the closeness of values within clusters. In applications which involve a large number of dimensions describing the input, we can use the cosine similarity measure which is a space of random infinite features with structure that generalizes two or three-dimensional Euclidean space used to find the maximally separated equidistant strange values. The cosine distance specifies the angle between vectors of all dimensions as shown in Fig. 1.

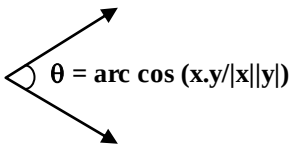

Fig.1. Geometric interpretation of inner product

The Cosine distance is given as:

$$d_c(T_i, T_j) = \frac{\sum_{K=1}^{N} T_{ik} T_{jk}}{\sqrt{\sum_{K=1}^{N} T_{ik}^2 \sum_{k=1}^{N} T_{jk}^2}}$$

The Euclidean distance [10] is used to put the remaining points from the dataset into clusters formed by the maximally separated equidistant k strange points. The Euclidean distance between the points i(a_1,b_1 ,c_1,d_1) and j(a_2,b_2,c_2,d_2) is given by:

$$d(i,j) = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 + (c_1 - c_2)^2 + (d_1 - d_2)^2}$$

CODED EKSTRAP – The Proposed Extension

The COsine Distance and Euclidean Distance based Enhanced K STRAnge Points (CODED EKSTRAP) clustering algorithm begins by finding the minimum of the dataset. Since the cosine distance measure is being used for finding strange points which are furthest apart from each other, we need to set the value of the origin as (1,1) instead of (0,0) while searching for the minimum of the dataset. This is because the value of the cosine distance will yield a zero when finding the distance of any prospective minimum point from the origin (0,0). To avoid this scenario the value of the origin is set as (1,1) and then a point which is at minimum distance from (1,1) is found. This is denoted as the temporary minimum as seen in Fig. 2.

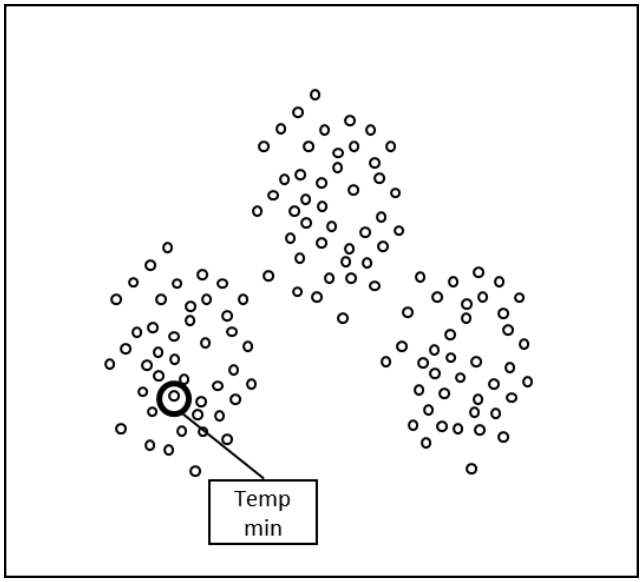

Fig. 2. Temp min of dataset.

Now the search for the actual minimum begins by looking for any point in the dataset which is less than the temporary minimum using dictionary sorting. In dictionary sorting the first dimension of the temporary minimum is checked with the first dimension of all points in the dataset to see if there is any point whose first dimension value is less than that of the temporary minimum.

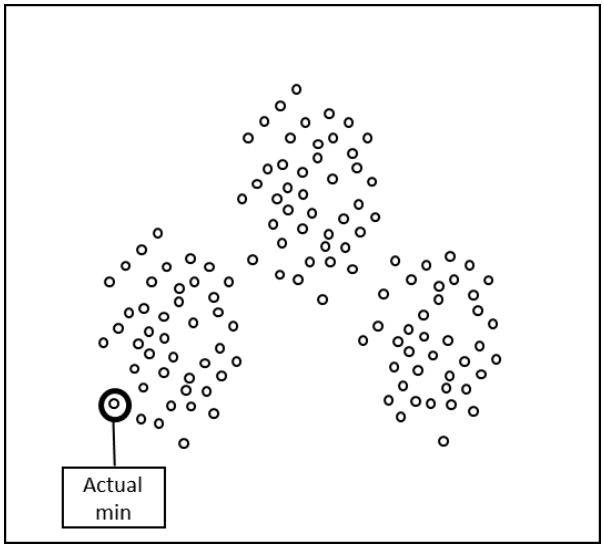

Fig. 3. Actual min of dataset.

If there is any such point, then that point will be the actual minimum. If the comparison shows that the values in the first dimension of the prospective point and temporary minimum are equal then the value in the next dimension is checked and this process is repeated till the actual minimum is found as shown in Fig. 3. The algorithm then finds a point which is at maximum distance from the minimum. It then locates a third point from the dataset which is maximally separated from the two strange points located in the previous steps. It then corrects the location of the third strange point by finding a central point between the third strange point and the minimum or maximum depending on which point the uncorrected third strange point is closest to. If the clustering requirement is to find K=3 clusters from the dataset then the similarity is found using the cosine distance measure. The Euclidean distance is then used to output the user defined K = 3 clusters using these 3 furthest strange values as shown in Fig. 4.

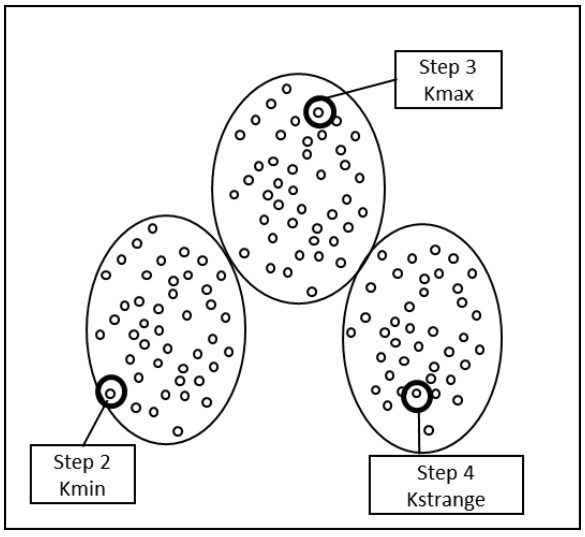

Fig. 4. Formation of K=3 clusters.

For the user defined clustering requirement of K = M clusters, the sequence of steps for finding the M Strange points from each other is continued and then the remaining values in the input set are grouped into clusters formed by these K = M strange values based on the Euclidean distance measure.

The CODED EKSTRAP Clustering Algorithm

Input: The cluster requirement of K = M and a repository having n data items D = {D_1, D_2, D_3, D_4,…….. D_n}
Output: A group of K clusters.

Steps

- First the origin is set to 1 for the number of dimensions and this value of the origin is used to find the temporary minimum of the dataset using the cosine distance formula.
- Then the actual minimum of the dataset is found using dictionary sorting and designated as K_{min}
- Next, a point which is farthest from K_{min} is found by using the cosine distance measure.
- After this we locate a third point s which is farthest from K_{min} and K_{max} using the Cosine distance.

  If s is equidistant to K_{min} and K_{max}, then s is the third strange point

  If s is closer to K_{min} than to K_{max}, then s is corrected using the formula

$$Kstrn = Kstrn_{prev} + X\left(\frac{\left|K\max - Kstrn_{prev}\right|}{K-1}\right)$$

  If s is closer to K_{max} than to K_{min}, then s is corrected using the formula

$$Kstrn = abs\left(Kstn_{prev} - X\left(\frac{\left|Kstrn_{prev} - K\min\right|}{K-1}\right)\right)$$

  where

  K= number of clusters

  X ranges from K=1,2,3,…..K-2

  $Kstrn_{prev}$ = uncorrected value of s and Kstrn = corrected value of s
- Repeat these steps until the K points equaling the number of required clusters mentioned in the problem are found.
- Accumulate the remaining values of the input set into clusters formed by the K non collinear furthest strange data items using the Euclidean distance measure.
- Output K=M clusters.

Implementation of the proposed algorithm

Consider a clustering requirement for 3 clusters of any dataset.
Step I: Finding the temporary minimum of the dataset
For this the origin is initialized as follows.

```
double orig[][]=new double[1][4];
   for(int v=0; v<1; v++){
      for(int w=0; w<4; w++){
         orig[v][w]=1;
      }
}
```

Then the temporary minimum is found as shown below.

```
for (loop var <= SizeOfData){
      nrCosD = nrCosineDist (origin, data);
      drCosD = drCosineDist (origin, data);
      cosD = nrCosD / drCosD;
      ed[k] = cosD;
      if(ed[k]>tempMin)
         tempMin = ed[k];
      TKmin=tempMin;
}
```

Step II: Finding the actual minimum

```
Kmin = TKmin
for(loop var <= SizeOfData){
      if(data[0][0]<Kmin[0][0]){
         Kmin = data;}}
      else if(data[0][0]= =Kmin[0][0])){
         if(data[0][1]<Kmin[0][1]){
            Kmin = data;}}
      else if(data[0][1]= =Kmin[0][1])){
         if(data[0][2]<Kmin[0][2]){
            Kmin = data;}}
      else if(data[0][2]= =Kmin[0][2])){
         if(data[0][3]<Kmin[0][3]){
            Kmin = data;}
}
```

Step III: Finding the maximum Kmax from Kmin

```
for(loop var <= SizeOfData){
      nrCosD = nrCosineDist (Kmin, data);
      drCosD = drCosineDist (origin, data);
      cosD = nrCosD / drCosD;
      ed[k] = cosD;
      if(ed[k]<max){
         max = ed[k];}
      Kmax=max;
}
```

Step IV: Finding the third strange point

```
for(loop var <= SizeOfData){
      dist=max+cosDist(Kmin, data) +cosDist(data, Kmax);
      if(dist<finalMax){
         finalMax = dist;
         Kstr = data;}
}
```

The third strange point is now found corrected as shown below.

```
f1= Math.abs(Kmin-s);
f2= Math.abs(Kmax-s);
```

for (**loop var < (K - 2)**){
if (f1= =f2)
$KStr_{final} = S_{prev;}$
else if (f1<f2)
$KStr_{final} = S_{prev} + X*(abs\ (Kmax- S_{prev}))/(K-1);$
else
$KStr_{final} = abs(S_{prev} - X*(abs\ (S_{prev} - Kmin))/(K-1));$}
where K = number of clusters and
X ranges from K=1,2,3,…..K-2.
S_{prev} = uncorrected value of s
$KStr_{final}$ = third strange point
As K=3 Strange points are found we stop searching for any more strange points.

Step V: Assigning points to respective clusters
The assignment of the leftover n-K values in the input set into is done using Euclidean distance measure.
for (**loop var <= SizeOfData**){
if((*dist*(k_{min},p)<=*dist*(k_{max},p))&(*dist*(k_{min},p)<=*dist*(k_{str},p)))
Assign p to Cluster 1
else if((*dist*(k_{str},p)<=*dist*(k_{min},p))&((*dist*(k_{str},p)<=*dist* (k_{max},p)))
Assign p to Cluster 2
else if((*dist*(k_{max},p)<=*dist*(k_{min},p))&(*dist*(k_{max},p)<=*dist*(k_{str},p)))
Assign p to Cluster 3}
where
p = data point.

Step V: Output K clusters

Experimental Results
The algorithm is tested with a 2D array dataset of 100000 points each with 4 columns and executed to give the results

```
int arrayRow = 100000;
int arrayCol = 4;

int K = 3; //number of required clusters

//Input as random data set
for(int i=0; i<arrayRow; i++){
    for(int j=0; j<arrayCol; j++){
        data[i][j]= (int)((Math.random()*10000) + 100);
    }
    System.out.println();
}
```

```
Problems @ Javadoc Console
<terminated> KStrangeCosineRandom4D [Java Application] C:\Program
K-Strange Points are:
Kstr1 = 100.0 2114.0 327.0 5083.0
Kstr2 = 8719.0 132.0 1906.0 112.0
Kstr3 = 4494.0  13899.0  4576.0  553.0
----------------------------------------------------------
Number of points in Cluster1 = 35062
Number of points in Cluster2 = 34230
Number of points in Cluster3 = 30708

----------------------------------------------------------
CODED EKSTRAP Clustering (K=3 clusters)
For 100000 4D Random points took: = 203.0 milliseconds
----------------------------------------------------------
```

Fig. 5. Output of CODED EKSTRAP clustering.

Once the K Strange values are determined, the algorithm then groups the remaining n-K values in the input set into clusters formed by those K-Strange values after which the K clusters are output as shown in Fig. 5.

Comparison with K-Means & Enhanced K Strange
In addition to Fig. 5 showing the results of the CODED EKSTRAP clustering algorithm, the results of the K-Means and Enhanced K-Strange points clustering algorithm are shown below in Fig. 6 and Fig. 7 for comparison of the three algorithms.

```
Problems @ Javadoc Console
<terminated> KM4DRandomDataset3Clusters [Java Application] C:\Program Files\Java\jd
Number of points in Cluster1 = 31181.0
Number of points in Cluster2 = 31314.0
Number of points in Cluster3 = 37505.0
----------------------------------------------------------
K-Means Clustering(3 Clusters)
For a Random Dataset of [100000][4] points took: = 795 milliseconds
----------------------------------------------------------
```

Fig. 6. Output of K-Means clustering.

```
Problems @ Javadoc Console
<terminated> EnhancedKStrangeRandom4D [Java Application] C:\Program Files\Java\
Number of points in Cluster1 = 33221
Number of points in Cluster2 = 47090
Number of points in Cluster3 = 19689
-----------------------------------------------
Enhanced K-Strange Points Clustering(3 Clusters)
For Random Dataset of [100000][4] points took: = 31 milliseconds
-----------------------------------------------
```

Fig. 7. Output of Enhanced K Strange points clustering.

Table I shows the results of the K-Means, Enhanced K-Strange points and the CODED EKSTRAP clustering algorithm for random datasets of 1000, 10000, and 100000 data points each with 4 dimensions for 3 clusters.

TABLE I. **COMPARISON OF EXECUTION TIMES**

Running time of algorithms (*milliseconds*) for growing data size			
Data Points	***K-Means Clustering***	***Enhanced K Strange Clustering***	***CODED EKSTRAP***
[1000][4]	16	0	0
[10000][4]	187	31	31
[100000][4]	795	31	203

From the table it is clear that the CODED EKSTRAP clustering algorithm proposed in this paper performs on par

with the Enhanced K-Strange points clustering algorithm for 10000 data points. But it takes a little more time for 100000 data points. It however easily outperforms the K-Means clustering algorithms for all sizes of inputs.

Conclusion

This paper is an extension to the Enhanced K Strange points clustering algorithm where in the cosine distance and Euclidean distance measures are used in the clustering process. It is observed that the CODED EKSTRAP clustering algorithm performs far better than the K Means clustering algorithm and gives almost the same results as the Enhanced K Strange Points clustering algorithm. The implementation shows that clustering can be done in an easy way using this method. The objective of this paper was to extend the enhancement to the K Strange points clustering algorithm using the Cosine and Euclidean distance measures.

References

[1] D. J. Hand, Heikki Mannila and Padhraic Smyth, Principles of Data Mining, MIT Press, 2001.

[2] Terence Johnson and Jervin Zen Lobo, "Collinear clustering algorithm in lower dimensions," IOSR Journal of Computer Engineering, ISSN: 2278-0661, ISBN: 2278-8727, vol. 6, Issue 5, pp. 08-11, Nov-Dec 2012.

[3] Sajid Nagi, Dhruba K. Bhattacharya and Jugal K. Kalita, "A preview on subspace clustering of high dimensional data," International Journal of Computer and Technology, ISSN: 22773061 vol. 6, no. 3, May 2013, pp. 441-448

[4] Santosh Kumar Singh and Terence Johnson, "Improved collinear clustering algorithm in lower dimensions," Proceedings of Second International Conference on Emerging Research in Computing, Information, Communication and Applications, ERCICA 2014, Elsevier publications, ISBN 9789351072638, Vol 3, pp 343-348.

[5] Terence Johnson, "Bisecting collinear clustering algorithm," International Journal of Computer Science Engineering and Information Technology Research, © TJPRC Pvt. Ltd, ISSN: 2249-6831, vol. 3, Issue 5, Dec. 2013, pp. 43-46

[6] Sunita Jahirabadkar and Parag Kulkarni, "SCAF-An efficient approach to classify subspace clustering," International Journal of Data Mining and Knowledge Management Process, vol. 3, no. 2, March 2013.

[7] Jiawei Han and Micheline Kamber, Data Mining – Concepts and Techniques, Elsevier.

[8] Santosh Kumar Singh and Terence Johnson, "Enhanced K Strange Points Clustering Algorithm", Proceedings of the '2nd International Research Conference on Emerging Information Technology and Engineering Solutions' *EITES 2015*, 978-1-4799-1838-6/15, pp 32-37, IEEE Computer Society Conference Publishing Services, © 2015 IEEE, DOI 10.1109/EITES.2015.14

[9] Santosh Kumar Singh and Terence Johnson, "K-strange points clustering algorithm ," Proceedings of International Conference on Computational Intelligence in Data Mining, 2014, Print ISBN 978-81-322-2204-0, Online ISBN 978-81-322-2205-7, Smart Innovation, Systems and Technologies, Vol 31, ISSN 2190-3018, Springer publications, pp 415-425

[10] A. Alfakih, A. Khandani, and H. Wolkowicz, "Solving, Euclidean distance matrix completion problems". Comput. Optim. Appl., 12(1999), pp.13-30