Journal of Advanced Computer Science & Technology, 4 (2) (2015) 231-236
www.sciencepubco.com/index.php/JACST
©Science Publishing Corporation
doi: 10.14419/jacst.v4i2.4749
Research Paper

AK-means: an automatic clustering algorithm based on K-means

Omar Kettani*, Faical Ramdani, Benaissa Tadili

Mohamed V- University, Scientific Institute, Physics of the Earth Laboratory Rabat, Morocco
**Corresponding author E-mail: kettani.o@gmail.com*

Abstract

In data mining, K-means is a simple and fast algorithm for solving clustering problems, but it requires that the user provides in advance the exact number of clusters (k), which is often not obvious. Thus, this paper intends to overcome this problem by proposing a parameter-free algorithm for automatic clustering. It is based on successive adequate restarting of K-means algorithm. Experiments conducted on several standard data sets demonstrate that the proposed approach is effective and outperforms the related well known algorithm G-means, in terms of clustering accuracy and estimation of the correct number of clusters.

Keywords: *Automatic Clustering; G-Means; K-Means; Parameter-Free Clustering.*

1. Introduction

Clustering is the process of grouping data into disjoint set called clusters such as that similarities among data members within the same cluster are maximal while similarities among data members from different clusters are minimal. The optimization of this criterion is an NP hard problem in general Euclidean space d, even when the clustering process deals with only two clusters [1]. To tackle this problem, many approximation algorithms have been proposed, aiming to find near optimal clustering solution in reasonable computational time. Most of the existing clustering algorithms depend on one or more tuning parameters, which are often difficult to determine, because they may require many empirical error-trials steps without a reliable effective result. K-means [2], the most prominent clustering algorithm has a major drawback: the user must specify the correct number of clusters in advance, which is often a difficult task when the distribution of the given data set is unknown.
In this paper, an alternative parameter free method for automatic clustering, called AK-means, is proposed. It is based on successive adequate restarting of K-means. Algorithm validation and comparative study with G-means [3], a related well known algorithm, are conducted using several real-worlds and artificial clustering data sets from the UCI Machine Learning Repository [4].
In the next section, some related works are briefly discussed. Then the proposed approach is described in Section 3. Section 4 presents application's results of this clustering method to different standard data sets and reports its performance. Finally, conclusions of the paper are summarized in Section 5.

2. Related work

Despite the fact that obtaining an optimal number of clusters k for a given data set is an NP-hard problem [5], several methods have been developed to find k automatically.
Pelleg and Moore [6] introduced the X-means algorithm, which proceed by learning k with k-means using the Bayesian Information Criterion (BIC) to score each model, and choose the model with the highest BIC score. However, this method tends to over fit when it deals with data that arise from non-spherical clusters. Tibshirani et al. [7] proposed the

Gap statistic, which compares the likelihood of a learned model with the distribution of the likelihood of models trained on data drawn from a null distribution. This method is suitable for finding a small number of clusters, but has difficulty when k increases. Hamerly and Elkan [3] proposed the G-means algorithm, based on K-means algorithm, which uses projection and a statistical test for the hypothesis that the data in a cluster come from a Gaussian distribution. This algorithm works correctly if clusters are well-separated, and fail when clusters overlap and look non-Gaussian. In our experiments, G-means tends to overestimate the number of clusters, as reported in section 4.
In the present work, an alternative approach is proposed, attempting to overcome these issues.

3. Proposed approach

The proposed algorithm starts by setting k=floor $((n)^{1/2})$; where n is the number of objects in the given data set. This choice is motivated by the fact that this number lies in the range from 2 to $(n)^{1/2}$, as reported by Pal and Bezdek in [8]. Then it applies a deterministic initialization procedure proposed by the authors in [9]. K-means algorithm is applied with these initial k centroids, and centroid of the smallest cluster is removed, then K-means restarts with the remaining centroids. At each iteration, the maximum of CH cluster validity index [10] of the current partition is stored. We used this index because it is relatively inexpensive to compute, and it generally outperforms other cluster validity indices as reported by Milligan and Cooper in [11]. This process is repeated until k=2. Finally, the algorithm outputs the optimal k and partition corresponding to the maximum value of CH stored so far. This algorithm is outlined in the pseudo-code below:

```
Algorithm AK-means

Input: $D= \{x_1, x_2. . . x_n\}$ in $R^d$

Output:   k mutually disjoint clusters $C_1 ... C_k$ such that $\bigcup_{j=1}^{k} C_j = D$

$k \leftarrow \lceil (n)^{1/2} \rceil$

$X \leftarrow D$

For j=1 to k do

        $C_j \leftarrow$ KNNsearch($x_1$, X, $\lceil n/k \rceil$)

        $c_j \leftarrow \sum_{x_i \in C_j} x_i / \lceil n/k \rceil$

        $X \leftarrow X - C_j$

End For

$[I,c] \leftarrow$ K-means (D,c,k)

$ko \leftarrow k$

$Io \leftarrow I$

$CHo \leftarrow CH(I)$

While k>2 do

        $j \leftarrow \mathrm{argMin}_{i<=k}(|C_i|)$

        $c_j \leftarrow []$

        $k \leftarrow k-1$
```

```
    [I,c]$\leftarrow$ K-means (D,c,k)
    if CHo <CH(I) then
        ko $\leftarrow$k
        Io $\leftarrow$ I
        CHo $\leftarrow$ CH (I)
    End if
End while
Output: ko and Io
```

4. Experimental results

Algorithm validation is conducted using seven real-world clustering data sets, namely breast, iris, wine, glass, ruspini, thyroid, yeast and 14 artificial generated clustering data sets from the UCI Machine Learning Repository. Data sets s1 to s4 are generated with varying complexity in terms of spatial data distributions, which have 5000 vectors scattered around 15 predefined clusters with varying degrees of overlap. Data sets a1, a2, and a3 are generated in 2-dimensional Gaussian distribution; there are 150 vectors per cluster. Dim32 to Dim528 are high-dimensional data sets with 16 Gaussian clusters.
Silhouette index [12] which measures the cohesion based on the distance between all the points in the same cluster and the separation based on the nearest neighbor distance, was used in these experiments (bigger average silhouette value indicates a higher clustering accuracy).
In these experiments, a comparative study between G-means and AK-means is conducted on these data sets using Matlab software on a computer with Intel Core 2Duo CPU 2.8 GHZ and RAM 4.0GB memory. The number of clusters found by both algorithms, the average of silhouette values and CPU running time are reported in table 1.
In our experiments, we used $\alpha = 0.0001$ the significance level of the test, for G-means script.
The results of the experiments with different data sets indicate that the proposed approach estimates the correct number of clusters, in 18 cases among 21 tested data sets. A Matlab code of the proposed approach is given in the appendix.

5. Conclusion

In this work, an algorithm was suggested for automatic clustering. This approach estimated the correct number of clusters in almost all tested data sets. This method was compared with the related well known algorithm, G-means, which improved for finding the correct number of clusters. The comparisons also showed that the proposed approach is better than G-means in terms of clustering accuracy.
In future work, it will be of interest to find a tighter upper bound on the number of clusters, instead of $n^{1/2,}$ in order to reduce the number of computation's steps of the proposed approach. Another possible algorithm's speed up is to avoid unnecessary distance calculations by exploiting the triangle inequality following the method developed by Elkan in [13]. A further possible improvement of the proposed approach will consist to try more adequate similarity measures instead of Euclidean distance, in order to enhance its clustering accuracy.

Acknowledgements

This work is supported by Mohamed V University funding from P-U. Project.

Table 1: Experimental Results of Application of G-Means and AK-Means on Different Data Sets.

		G-means			AK-means		
Dataset	k	k found	Mean Silh	CPU time (s)	k found	Mean Silh	CPU time (s)
breast	2	106	0.4226	5.1120	2	0.7542	1.0386
iris	3	3	0.7786	2.8643	3	0.7786	0.3728
glass	7	2	0.7879	0.7893	15	0.6514	0.5293
ruspini	4	4	0.9086	0.9048	4	0.9086	0.0997
thyroid	2	3	0.7773	0.9017	3	0.7773	0.4168
wine	3	3	0.5043	0.9940	3	0.5043	0.3411
yeast	10	19	0.2659	2.4021	2	0.4102	6.2920
a1	20	23	0.7337	2.0891	20	0.7892	5.2059
a2	35	40	0.7413	2.3931	35	0.7911	14.5874
a3	50	53	0.7727	3.7775	50	0.7949	27.6954
D31	31	31	0.9222	1.6490	31	0.9222	4.9055
dim32	16	111	0.4413	5.3006	16	0.9962	3.5238
dim64	16	109	0.5464	5.4418	16	0.9985	6.1691
dim128	16	111	0.6214	8.2748	16	0.9991	13.1553
dim256	16	106	0.6032	9.8462	16	0.9996	28.2731
dim528	16	109	0.5999	20.4447	16	0.9998	63.4478
R15	15	15	0.9361	1.2827	15	0.9361	0.4046
s1	15	80	0.5632	8.4497	15	0.8803	12.7067
s2	15	87	0.5563	13.4046	15	0.8009	18.1031
s3	15	73	0.5393	14.7327	15	0.6659	27.3990
s4	15	85	0.5315	25.1755	15	0.6446	26.1389

References

[1] Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering". Machine Learning 75: 245–249. http://dx.doi.org/10.1007/s10994-009-5103-0.

[2] Lloyd. S. P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory 28 (2): 129–137. http://dx.doi.org/10.1109/TIT.1982.1056489.

[3] Greg Hamerly and Charles Elkan. Learning the k in k-means. In Proceedings of the seventeenth annual conference on neural information processing systems (NIPS), pages 281–288, 2003

[4] Asuncion, A. and Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[5] H. Spath, Clustering Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood, Chichester, 1980.

[6] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In Proceedings of the 17th International Conf. on Machine Learning, pages 727–734. Morgan Kaufmann, 2000.

[7] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the Gap statistic. Journal of the Royal Statistical Society B, 63:411–423, 2001. http://dx.doi.org/10.1111/1467-9868.00293.

[8] Pal, N.R. and Bezdek, J.C. (1995) On Cluster Validity for the Fuzzy c-Means Model. IEEE Transactions on Fuzzy Systems, 3, 370-379. http://dx.doi.org/10.1109/91.413225.

[9] Kettani, O.; Tadili, B. and Ramdani, F. - A deterministic k-means algorithm based on nearest neighbor search. International Journal of Computer Applications (0975 – 8887), Vol. 63, No.15, February 2013. http://dx.doi.org/10.5120/10544-5541.

[10] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. Communications in Statistics, 3:1–27, 1974.

[11] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. Psychometrica, 50:159–179, 1985. http://dx.doi.org/10.1007/BF02294245.

[12] L. Kaufman and P. J. Rousseeuw. Finding groups in Data: "an Introduction to Cluster Analysis". Wiley, 1990. http://dx.doi.org/10.1002/9780470316801.

[13] C. Elkan, "Using the triangle inequality to accelerate k-means", ICML 2003 Conference Proceedings, p. 147#153, 2003.

Appendix

```
a Matlab code of the proposed approach
clear all;
[file,filePath] = uigetfile('*.txt');
if isequal(file, 0)
   return;
end
dname = [filePath file];
try
  a = load(dname);
catch
  set(handles.Outext1, 'String', 'Running state: incorrect data file !');
  return
end;
% a(:,3)=[]
co = 'brgmcyk';
pt =
{'bs','r^','md','go','c+','rs','m^','gd','co','b+','gs','b^','rd','bo','g+','ms','c^','cd
','mo','m+','g+','gs','b^','rd','bo','g+','ms','c^','cd','mo','m+','g+','ms','c^','cd','m
o','m+','g+','gs','b^','rd','bo','g+','ms','c^','cd','mo','m+','g+','bs','r^','md','go','
c+','rs','m^','gd','co','b+','gs','b^','rd','bo','g+','ms','c^','cd','mo','m+','g+','gs',
'b^','rd','bo','g+','ms','c^','cd','mo','m+','g+','ms','c^','cd','mo','m+','g+','gs','b^'
,'rd','bo','g+','ms','c^','cd','mo','m+','g+','bs','r^','md','go','c+','rs','m^','gd','co
','b+','gs','b^','rd','bo','g+','ms','c^','cd','mo','m+','g+','gs','b^','rd','bo','g+','m
s','c^','cd','mo','m+','g+','ms','c^','cd','mo','m+','g+','gs','b^','rd','bo','g+','ms','
c^','cd','mo','m+','g+'};
lc = length(co);
[n,p]= size(a);
k=round(sqrt(n))
m=init(a,k)
[idx,m] = kmeans(a,k,'start',m,'emptyaction','singleton')
ko=k
CHo= vCH(a,idx)
idxo=idx
while k>2
  [mD,id] =min(arrayfun(@(j) length(find(idx==j)),1:k))
   m(id(1),:)=[]
   k=k-1
   [idx,m] = kmeans(a,k,'start',m,'emptyaction','singleton')
    CH= vCH(a,idx)
      if CHo<CH
          CHo=CH
          ko=k
          idxo=idx
      end;
end
k=ko
idx=idxo
[s,h]  = silhouette(a,idx);
figure;
for j=1:k
plot(a(idx==j,1),a(idx==j,2),pt{j},'MarkerSize',5)
hold on
end
si0= mean(s);
disp(si0)
disp(k)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [CH] = vCH(data,labels)
[nrow,nc] = size(data);
labels = double(labels);
k=max(labels);
[sw,sb] = v_sumsqures(data,labels,k);
```

```
ssw = trace(sw);
ssb = trace(sb);
if k > 1
  CH = ssb/(k-1);
else
  CH =ssb;
end
CH = (nrow-k)*CH/ssw;    % Calinski-Harabasz

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [W, B] = v_sumsqures(data,labels,k)

if (size(labels, 1) == 1)
    labels = labels';
end
[ncase,m] = size(data);
Dm = mean(data);
Dm = data - Dm(ones(ncase,1),:);
T = Dm'*Dm;
W = zeros(size(T));
Dm = zeros(k,m);
for i = 1:k
   if k > 1
      Cindex = find(labels == i);
   else
      Cindex = 1:ncase;
   end
   nk = length(Cindex);
   if nk > 1
      dataC = data(Cindex,:);
      m = mean(dataC);
      Dm(i,:) = m;
      dataC = dataC - repmat(m,nk,1);
      W = W + dataC'*dataC;
      dataC = sum(dataC.^2,2);
   end
end
B = T - W;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function distances = calcdist2(data,center)
[n,dim] = size(data);
[n2,dim2] = size(center);
if n2 == 1
    distances = sum(data.^2, 2) - 2*data*center' + center*center';
elseif n2 == n
    distances = sum( (data - center).^2 ,2);
else
    error('bad number of centers');
end
distances = distances;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function C=init(a,k)
[n,p]= size(a);
Z=a
C=[]
idx=[]
for j=1:k-1
     idx=knnsearch(Z(1,:),Z,round(n/k))
     C(j,:)=mean(Z(idx,:))
     Z(idx,:)=[]
end
 C(k,:)=mean(Z(1:end,:))
```