Generating K-Prototype Points for Unsupervised Learning

Amir Ahmad, Shehroz S. Khan,
Solid State Physics Laboratory, DRDO, Delhi, India-110054
Scientific Analysis Group, DRDO, Delhi, India-110054

Abstract

Traditionally initial cluster centers for K-Means clustering algorithm are generated randomly which may lead K-Means to get trapped in local minima. In this paper we present a method to generate K-prototype points that are quite near to the desired cluster centers. The effectiveness of the method has been evaluated experimentally on several numeric data sets.

Keywords: Data Mining, Cost function, K-means clustering, Normal distribution, Cluster centers

1 Introduction

Clustering aims to find the natural grouping of the data. The objective of clustering is to find convenient and valid organization of data into 'similar' groups. Clustering has a variety of applications in different fields like data mining and knowledge discovery [1], data compression and vector quantization [2], finance [3], and bio-informatics [4].

K-means clustering algorithm [5], is one of the most popular clustering algorithm. A priori knowledge of number of clusters is a must for K-means clustering.

*Email address: amirahmad01@yahoo.com.in
algorithm. K-means is defined over numeric data \([6][7]\) since it computes the mean of the clusters. K-means algorithm calculates cluster centers iteratively. Let \(X=\{x_1, x_2, \ldots, x_n\}\) be a \(m\) dimensional data set having \(n\) patterns contained in a pattern space \(S\). Therefore, we seek the \(K\)-regions (clusters) \(S_1, S_2, \ldots, S_K\) such that every \(x_i, i=1,2,\ldots, n\), falls into one of these regions and no \(x_i\) falls in two regions. K-means algorithm is based on the minimization of cost function defined as squared distance from all points in a cluster domain to the cluster center, that is
\[
\min_{x \in S_j(t)} \sum (x - z_j)^2
\]
where \(S_j(t)\) is the cluster domain for cluster center \(z_j\) at the \(t^{th}\) iteration. The clustering procedure of K-means algorithm is illustrated as under :

1. Randomly choose \(K\)-prototype points as the initial cluster centers
2. Distribute the pattern samples \(x\) among the chosen cluster domains according to the following criteria :
\[
x \epsilon S_j(t) \quad \text{ if } (x - z_j(t))^2 < (x - z_i(t))^2
\]
for all \(i=1,2,\ldots,K\) and \(i \neq j\)
3. Update the cluster centers as
\[
z_j(t+1) = \frac{\sum_{x \epsilon S_j(t)} x}{|N_j|} \quad j = 1, 2, \ldots, K
\]
where \(|N_j|\) is the number of data items in the \(j^{th}\) cluster. These adjusted centers will minimize the sum of squared distances from all points in \(S_j(t)\) to the new cluster centers.
4. Repeat step 2 and 3 till the algorithm converges.

K-means does not guarantee unique clustering because we get different results with randomly chosen initial clusters [8]. Machine learning practitioners find it difficult to rely on the results obtained using randomly selected initial cluster centers. The K-means algorithm gave better results only when the initial partitions was close to the final solution [9]. Several attempts have been reported
to generate K-prototype points that can be used as initial cluster centers. A recursive method for initializing the means by running K clustering problems is discussed by Duda, R.O., et al [7]. Bradley, P.S. et al [10] reported that the values of initial means along any one of the m coordinate axes is determined by selecting the K densest "bins" along that coordinate. Bradley, P.S. et al [11] proposes a procedure that refines the initial point to a point likely to be close to the modes of the joint probability density of the data. Mitra, P. et al [12] suggested a method to extract prototype points based on density based multi-scale data condensation.

The paper is organized as follows. In section 2, we present the proposed algorithm to generate K-prototype points. We discuss some experimental results on real world numeric data sets in section 3. Section 4 presents the conclusion.

2 K-Prototype Generation

In K-Means clustering algorithms the procedure adopted for choosing initial cluster centers is extremely important as it has a direct impact on the formation of final clusters. Our proposed algorithm generates the K-prototype points that are close to the desired cluster centers. As there are no universally accepted method for selecting initial cluster center [13], we compare the results against the standard method of randomly choosing initial starting points.

The proposed algorithm is based on normal distribution. We assume that each of the attributes of the pattern space is normally distributed. For K-fixed clusters we divide the normal curve into K partitions such that the area under these partitions is equal. We then take the midpoints of the interval of each of these partitions. This has been done to scrap the outliers and to keep the centers as far as possible. The area corresponding to the mid-points can be defined in the following way:

Area from $-\infty$ to $s^{th}$ mid-point (from left side) = $\frac{2s-1}{2K}$, $s=1,2,\ldots,K$

We now calculate the percentile [14] corresponding to these areas. We compute the attribute values corresponding to these percentiles using mean and
standard deviation of the attribute. These attribute values will be treated as one of the coordinate of the initial \( K \)-center. This process has to be adopted for all the attributes to get the \( K \)-cluster centers coordinate. The starting point will be a combination of these center coordinates.

For generating this combination, first we assign the class label of every pattern using the distance between the \( i^{th} \) attribute value of the patterns and \( i^{th} \) coordinate of the center using partitioning around the moving centers [15]. Now we perform the \( K \)-means clustering procedure to get the final clusters corresponding to this exercise. Repeat the above procedure for all the attributes. We have \( m \) (number of attributes) classes associated with every pattern. We call it as a pattern string. These classes of pattern string may or may not be same. For \( n \) patterns we will have \( n \) such pattern strings, where \( i^{th} \) entry of the \( j^{th} \) pattern string corresponds to the class of the \( j^{th} \) pattern when the initial centers were based on \( i^{th} \) attribute. We select the \( K \) most frequent pattern strings out of these \( n \) pattern strings. The \( i^{th} \) value of the pattern string will be replaced by the initial point as computed above.

### 2.1 Proposed Algorithm

**Input:**
D - The set of \( n \) data elements described with attributes \( A_1, A_2, \ldots, A_m \), where \( m= \) no. of attributes and all attributes are numeric
K - predefined number of clusters

**Output:**
Initial centers of clusters \( \hat{C}_{ij} \), \( i=1,2,\ldots,K \) and \( j=1,2,\ldots,m \)

begin

1. For each attribute \( A_j \) repeat step 2 to 7

2. Compute mean \( (\mu_j) \) and standard deviation \( (\sigma_j) \)

3. Compute percentile \( z_s \), corresponding to area from \( -\infty \) to \( z_s = \frac{2s-1}{2K} \),
   where \( s=1,2,\ldots,K \)
4. Compute attribute value corresponding to these percentiles using means and standard deviation of the attribute as

\[ x_s = z_s \cdot \sigma_j + \mu_j = C_{sj} \]

where \( C_{sj} \) is associated with \( s \) class value

5. Create initial partitions using Euclidean distance between \( x_s \) and \( A_j^{th} \) attribute of all patterns (The assigned class label is treated as the class of the pattern)

6. Execute K-means on complete data set

7. Store the class labels as \( S_{ij} \) where \( t=1,2,\ldots,n \)

8. Generate pattern string, \( P_t \) corresponding to every pattern by storing the class labels. every pattern string will have \( m \) class labels. The \( j^{th} \) value of pattern string \( P_t = S_{tj} \)

9. Choose the \( K \) most frequent \( P_t \)'s. Break ties randomly

10. Replace the \( j^{th} \) value of string \( P_{k'} \) \( \{k' = 1,2,\ldots,K\} \), with \( C_{rj} \) where \( r=\text{Integer value of } j^{th} \) value of string \( P_{k'} \)

11. This combination of \( C_{rj} \) for \( P_{k'} \) \( \{k' = 1,2,\ldots,K\} \), will be the initial cluster center \( \hat{C}_{ij} \)

end

3 Results and Discussion

In order to demonstrate the effectiveness of the proposed algorithm for computing K-prototype points, we tested its performance on some real world data sets (fossil data, iris data, wine recognition data, british town data and satimage data)

Since different attributes are measured on different scales, when Euclidean distance formula is used directly, the effect of some attributes might be completely
dwarfed by others that have larger scales of measurement. Consequently it is usual to normalize all attribute values to lie between 0 and 1 [16], by calculating

\[ a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i} \]

where \( v_i \) is the actual value of attribute \( i \), and the maximum and minimum are taken over all attributes in the data set.

To measure the degree of closeness between the generated K-prototype and the desired K-cluster centers we calculated Closeness Index as

\[ \text{Closeness Index} = \frac{1}{K \times m} \sum_{s=1}^{K} \sum_{j=1}^{m} \left| \frac{f_{sj} - C_{sj}}{f_{sj}} \right| \]

where \( f_{sj} \) is the \( j^{th} \) attribute value of the desired \( s^{th} \) cluster center and \( C_{sj} \) is the \( j^{th} \) attribute value of the initial \( s^{th} \) cluster center.

1. **Fossil Data**: This data set is taken from Chernoff [17]. It consists of 87 nummulitidae specimens from the Eocene yellow limestone formation of north western Jamaica. Each specimen is characterized by six measurements. There are three cluster groups as identified by Chernoff, which contains 40, 34 and 13 patterns each.

2. **Iris Data**: This data set [17] has three classes that represents three different varieties of Iris flowers namely Iris setosa, Iris versicolor and Iris virginica. Fifty samples were obtained from each of three classes, thus a total of 150 samples is available. Every sample is described by a set of four attributes viz sepal length, sepal width, petal length and petal width.

3. **Wine recognition data**: This data set is taken from UCI repository website [18]. This data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There were overall 178 instances. There are 59, 71 and 48 instances in class I, class II and class III respectively. The classes are separable.
<table>
<thead>
<tr>
<th>Desired Centers</th>
<th>Generated Centers</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0, 3.4, 1.4, 0.2</td>
<td>5.0, 3.4, 2.0, 0.4</td>
</tr>
<tr>
<td>5.9, 2.7, 4.2, 1.3</td>
<td>5.8, 3.0, 3.7, 1.1</td>
</tr>
<tr>
<td>6.6, 2.9, 5.5, 2.0</td>
<td>6.6, 2.6, 5.5, 1.9</td>
</tr>
</tbody>
</table>

Table 1: Iris Data

4. **British Town data**: This data set [17] represents social-economic data collected from 155 British towns originally described by 57 variables. Later first four principal components from the correlation matrix of the original data was calculated. These 50 samples were identified to form four clusters groups having 10, 10, 16, 14 instances in each cluster.

5. **Satimage Data**: The database [18] is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighbourhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighbourhood and a number indicating the classification label of the central pixel. There are 2000 patterns in the testing set having 6 decision classes with 36 numerics attributes.

Table 1 suggests that for the Iris Data set the generated prototypes for 3 clusters are very close to the desired cluster centers. Same interpretation can be drawn for results depicted in Table 2 for Wine data. Similar results were obtained for other data sets, but due to lack of space we are not providing generated K-prototypes.

To generate random K-prototype points, we associate random class label with every pattern and cluster centers are taken as random K-prototype points. Then the Closeness Index was computed. This process was repeated 10 times and the averaged value of Closeness Index is reported in Table 3. Low value of Closeness Index reflects the closeness of generated prototype points to the
<table>
<thead>
<tr>
<th>Desired Centers</th>
<th>Generated Centers</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.74, 2.01, 2.45, 17.0, 106</td>
<td>13.81, 1.22, 2.64, 16.1, 113</td>
</tr>
<tr>
<td>2.8, 2.98, 0.29, 1.89, 5.52</td>
<td>2.9, 3.02, 0.23, 2.16, 5.05</td>
</tr>
<tr>
<td>1.06, 3.15, 1115</td>
<td>0.95, 3.31, 1060</td>
</tr>
<tr>
<td>12.27, 1.93, 2.24, 20.2, 94</td>
<td>12.19, 2.33, 2.09, 19.4, 85</td>
</tr>
<tr>
<td>2.2, 2.08, 0.36, 1.63, 3.08</td>
<td>2.2, 2.02, 0.36, 1.59, 2.74</td>
</tr>
<tr>
<td>1.05, 2.78, 519</td>
<td>1.18, 2.61, 432</td>
</tr>
<tr>
<td>13.15, 3.33, 2.43, 21.4, 99</td>
<td>13.00, 3.45, 2.36, 22.8, 99</td>
</tr>
<tr>
<td>1.6, 0.78, 0.44, 1.15, 7.39</td>
<td>1.6, 1.03, 0.48, 1.02, 7.36</td>
</tr>
<tr>
<td>0.68, 1.68, 629</td>
<td>0.72, 1.90, 746</td>
</tr>
</tbody>
</table>

Table 2: Wine Data

<table>
<thead>
<tr>
<th>Data set</th>
<th>Closeness Index</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed Algo.</td>
</tr>
<tr>
<td>Fossil Data</td>
<td>0.2286</td>
</tr>
<tr>
<td>Iris Data</td>
<td>0.1402</td>
</tr>
<tr>
<td>Wine Data</td>
<td>0.0833</td>
</tr>
<tr>
<td>British Town data</td>
<td>0.9009</td>
</tr>
<tr>
<td>Satimage Data</td>
<td>0.2730</td>
</tr>
</tbody>
</table>

Table 3: Comparing Closeness Index of Data sets
desired cluster centers. Table 3 suggests that we get smaller values using the proposed algorithm in comparison to randomly generated K-prototypes.

4 Conclusion

We have presented a new and efficient algorithm for computing generating K-prototype points for K-means clustering algorithm. This procedure is based on the assumption that attributes are normally distributed. For every attribute we computed different points that are far apart under the constraint that boundaries of the normal curve are avoided and points are symmetric with respect to mean of the distribution. We picked the best combinations of these points to get the initial K-prototypes. Experimental results show that generated K-prototypes points are very near to the desired cluster centers. These generated K-prototype points can be used as initial cluster centers which may avoid K-means to get trapped in one of the numerous local minima.

5 Acknowledgements

The authors are thankful to Prof. Vikram Kumar and Prof. C. E. Veni Madhavan for their support to carry out this work.

References


[3] Finance


