Ensemble learning method for k-means clustering in Big Data environment

Automated k-means clustering with no prior knowledge

Dr. Yossi Kuttner (Ph.D.)
Algorithm Developer and Data Scientist
10/05/2018
Amdocs

✓ Amdocs is a leading software & services provider to the world’s most successful communications and media companies.

✓ Amdocs manufactures software for the management of subscribers of large telecommunications companies

  o Targeting subscribers
  o Predict churn
  o Understand churn
  o Recommendations
  o A subscription journey leads to churn or calling to call center
  o ...

© 2017 – Proprietary & Confidential Information of Amdocs
In Amdocs, we are building an Auto-ML tool that will sit on the clients side, will suggest what offering to give to which subscriber, with no prior knowledge – what are the offerings nor what are the features. The unique business need is to build a general model without being able to test it that will have good performance.
Our Goal

Apply fully automated clustering algorithm on huge amount of data with large number of features
Steps to achieve the goal

Pick and run cluster algorithm

Feature selection
Cluster Algorithm

Why K-means?

- simple
- fast
- very intuitive
- easy to distribute (Spark)
- scalable
- no parameters to set
K-means algorithm

requires that the user provides in advance the exact number of clusters (k)
Feature Selection

Which features should be selected in order to create the most distinct K clusters?
Feature Selection and K-means

Given a specific K, different set of features may be optimal

given set of features that define the problem features-space, a different optimal K may be appropriate
In Summary

The optimal K and set of features are Interdependent on one another.
Don’t be naïve!

The naïve and best way to find these two solutions is to try all possible combinations of k and different sets of features

finding the "winning" combination of K and features set, in terms of any chosen models' KPI, is very inefficient and very slow

The complexity of such an exhaustive search is $O((f!*n*k))$
Our Solution

like Bootstrap aggregating (bagging) meta-algorithm
Our solution- ensemble of mini-clusters

- The algorithm randomly divides the instances and features to create many subsets.

  1) For each subset we run the K-Means algorithm with different K’s

  2) Each K cluster independently and get in turn a performance score.

  3) For scoring, Shilhouette or Dunn were used.
Our solution - ensemble of mini-clusters

Step 1: Data for clustering

K – number of clusters
B – batch (bag) → subset
F – features

Step 2:
Randomly sample multiple subsets and cluster with different set of features and k

Step 3:
Our solution - “bagging” method, finding best $K$

**Step 3a:**
Writing the clustering score to $[K, B, F]$ 3D matrix. Finding the best $k$

$$\text{Best } k = \max \{k_1^*, k_2^*, k_T^*\}$$
Our solution- “bagging” method, features ranking

**Step 3b:**
Finding top T set of features for that k

feature i score = best_k{\(b_{1,f_i} + b_{2,f_i} + ... + b_{B,f_i}\)}
Our solution - “bagging” method

- The optimal K and feature-set combination is selected using voting over performance score.

  - The winning K ($K_{\text{winner}}$) is the one with the highest aggregated score amongst all mini-batches. The winning features subset is the one with the highest score for K-winner.
Results
Results – Iris flower data set

Samples (instances, observations)

<table>
<thead>
<tr>
<th></th>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Setosa</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Versicolor</td>
</tr>
<tr>
<td>50</td>
<td>6.4</td>
<td>3.5</td>
<td>4.5</td>
<td>1.2</td>
<td>Versicolor</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>5.9</td>
<td>3.0</td>
<td>5.0</td>
<td>1.8</td>
<td>Virginica</td>
</tr>
</tbody>
</table>

Features (attributes, measurements, dimensions)

Class labels (targets)
Results – Iris flower data set

- **Best k is found to be 3**

<table>
<thead>
<tr>
<th></th>
<th>Actual number in cluster</th>
<th>clustered correctly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setosa</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Versicolor</td>
<td>34</td>
<td>32</td>
</tr>
<tr>
<td>Virginicia</td>
<td>34</td>
<td>31</td>
</tr>
</tbody>
</table>

Total: 95% match
Results – Iris flower data set

- Feature ranking:
  1) Petal_Length
  2) Petal_Width
  3) Sepal_Length
  4) Sepal_Width
Results – synthetic clusters

We generate cluster – 3D (3 feature)s clusters of random blobs
Results – synthetic clusters

Messing up the 3rd feature data set
Results – synthetic clusters

Outcome

✓ The algorithm found 4 clusters

✓ Feature f1 gets a high score and features f2 and f3 are getting lower scores with small differences between them

✓ The algorithm reduced one dimension – which is what we expected 😊
# Results – synthetic clusters

<table>
<thead>
<tr>
<th>Blob</th>
<th>number in cluster</th>
<th>clustered correctly</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>3</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>125</td>
<td>125</td>
</tr>
</tbody>
</table>

Total: 100% match
Results – synthetic clusters

Outcome

Feature space of the 1st feature

Feature space of the 2nd feature
Summary

- We show here a simple method for clustering data without prior knowledge.

- This technique is appropriate for distributed systems.

- The algorithm finds a number of clusters that explain the data and ranks the features according to their ability to separate the clusters and describe the solution.
Future plan

1. Study more in depth the voting method. For example: adding a penalty where the $f_i$ was not included and if the score went up/down without it

2. Use several variants of k-means such as: k-modes and k-medoids

3. Testing the algorithm on different “real dataset” and examine the results and performance

4. Support mixture of numerical and categorical variables
Thank you