Software project: K-Means Improvements

Stefan Sebastian, 242

Contents

1 Specification

The purpose of this report is to document the 'K-Means Improvements' software library. The project presented contains implementations for the basic K-Means algorithm with random seeds and some improvements in terms of computation speed (Enhanced K-Means) and seed selection (K-d trees density based, Sort and split heuristic). The library offers some clear, easy to use methods for performing clustering and can be extended to incorporate new algorithms by following some simple conventions.

1.1 Motivation

The amount of data available is growing at an explosive rate[1] and most of it comes from people interacting online and does not have meaningful labels. There is a need for computationally efficient and reliable clustering methods and it does not make sense for every researcher to implement them in order to do an experiment. Furthermore, a good knowledge of algorithms might not be among the skills of a data scientist, meaning that they will look for a suitable tool.

Most frameworks that are popular in the python environment (sk-learn, tensorflow, etc) offer a wide selection of algorithms from different artificial intelligence subfields. Thus, it is not reasonable to expect that each topic will always be up to date with research results. This framework is narrow in scope, only considering K-Means clustering, meaning it can cover a wide range of improvements and variants from recent years.

1.2 Theoretical background

The first version of the library contains the following algorithms: basic K-Means, Enhanced K-Means, seed selection with K-d trees, seed selection using the Sort and Split heuristic. This section will give an overview of those algorithms and the context in which they are useful.

1.2.1 Enhanced K-Means

A simple improvement for the basic K-means has been proposed by Fahim et al.[2] under the name of "enhanced K-means" in 2006. The main idea is that some time can be saved during the computation of distances between points and cluster centers in the update phase of the original algorithm. This distance is calculated at every iteration of the algorithm but the information from previous iterations could be used, by employing an additional data structure that memorizes the previous distance between a point and its cluster's center. At every iteration the distance to the new cluster mean is calculated and if it's less or equal to the previous distance stored for that point then it is left in its cluster and there's no need to calculate the other k-1 distances.

The complexity of this algorithm can be approximated to O(nk) compared to the complexity of the classic algorithm which is O(nki), where n represents the number of points, k the number of clusters and i the maximum number of iterations. In brief, the time complexity when updating the position of a point is O(1) if it stays in its cluster and O(k) otherwise. Since the algorithm converges to a local minimum then the number of points updated in each iteration decreases, meaning the expected complexity is $nk\sum_{j=1}^{i} 1 \,/\, j$.

1.2.2 Seed selection using the Sort and Split heuristic

This conceptually simple heuristic for choosing the K-means seeds was proposed by Yedla et al.[3] in 2010. The main idea of the method is to sort all points in the initial dataset by distance from origin. In case that there are points with negative feature values then they are transformed, by subtracting the minimum attribute value out of the feature value for each point. After sorting, the data is partitioned in k equal sets, where k is the number of desired seeds. From each set, the middle point is chosen as a seed. This method was compared to the classic algorithm that uses the random seed initialization and has been shown to obtain better accuracy on all testing sets.

1.2.3 Seed selection with K-d trees

A k-d tree, introduced by Jon L. Bentley[4], meaning k-dimensional tree is a data structure used to organize points in k-dimensional space, which is useful for partitioning data and optimizing multidimensional queries.

An implementation of seed selection based on the k-d structure is proposed by Redmond and Heneghan[5] in 2007. The reason for using this data structure is to gain information about the density of the dataset. In order to suit the problem some decisions have been made while building the tree. At each step the dataset is split among the longest dimension of the bounding box. This box can be computed by taking the minimal and maximal values for each feature in the dataset, and building two new points. The longest dimension is simply the feature for which there is the greatest difference between the two points calculated in the previous step. The chosen condition for stopping and making a leaf node is when the size of the considered dataset is $\frac{n}{10K}$ or lower, which allows approximatively 10 leaf buckets for each cluster.

After the k-d tree is built some density estimations are made using the obtained leaf buckets. For each bucket the value of the density is calculated as N/V, where N is the number of points it contains and V is the volume of the bounding box. The volume is computed as the product of the differences between each feature of the two bounding points, or the geometric mean in case of two features being equal.

The heuristic for choosing the seeds is to get points that have a large density and are separated by a reasonable distance. Thus, the first seed is equal to the mean of the bucket with the highest density. The other k-1 seeds are chosen to be the ones that maximize the value g, where g is the product between the density of the bucket mean and the minimum distance to other bucket means. An optimization to avoid outliers is to compute the seeds, remove 20% of the least dense buckets, and then compute another pair of seeds.

2 Analysis

In order to evaluate the accuracy of the implemented algorithms some test data sets were generated and the results were displayed in a graphical way. Clustering is usually evaluated by an expert in the field for which it is applied, meaning that it is difficult to find a general method for evaluating clustering algorithms.

The method selected for generating test data was the *make_blobs* function from *sk_learn*[6]. For a quick evaluation of the algorithm a dataset of 1000 points was generated having the centers in $[1, 1, 1]$, $[4, 5, 6]$ and $[8, 9, 1]$ with a standard deviation of 1.

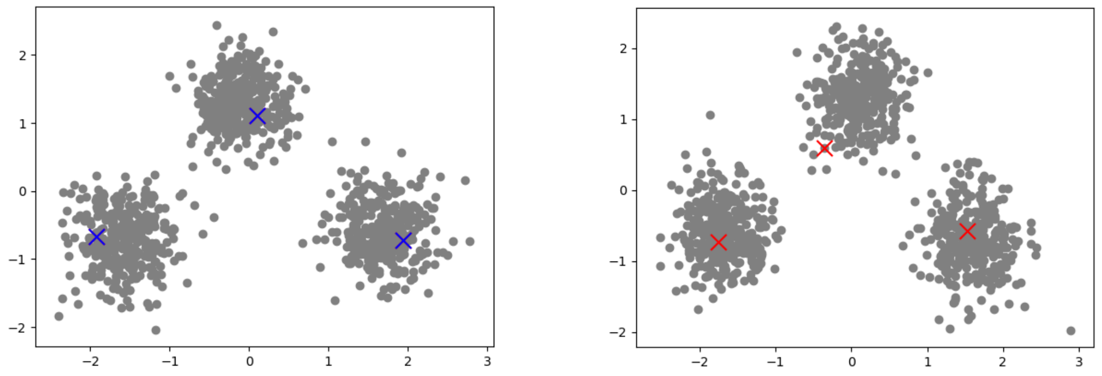
(a) K-d tree seeds (b) Sort and split heuristic

Figure 1: Comparison of seed selection algorithms

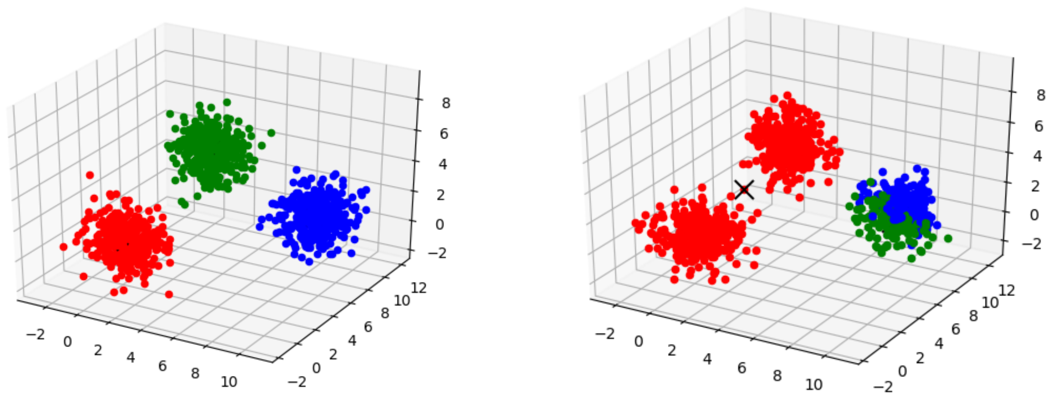
(a) Ideal clustering (b) Bad starting seeds

Figure 2: K-Means with random seeds

The points are then plotted using the matplotlib [7] library.

The seed selection algorithms show promising initial candidates, marked by X in figure 1. We can observe that both methods picked close to ideal initial seeds, each from a different cluster, which is a great improvement over random selection.

The main advantage of using these seed selection strategies is the ability to get constant results. For example, the basic K-means algorithm is run using randomly selected seeds for a number of times and some results are selected in figure 2. For most of the runs the result is the one expected at the beginning of the experiment (the one on the left), however, one in around 5 runs obtains an undesirable clustering, due to starting with poorly selected initial seeds.

A computational performance measurement is also done by logging the execution times. Using the previous dataset the difference between basic K-Means and Enhanced K-Means is barely noticeable, only a few milliseconds. However, on a dataset of 100.000 points with 9 features each the computational times are 14 seconds for the simple algorithm and 6 for the improved one, which is already a noticeable improvement.

3 Design

An overview of the main packages of the library can be seen in figure 3. All relevant classes are grouped under the 'algorithms' package. The two versions of the algorithm which are implemented in the project are the basic K-Means and the Enhanced K-Means. The methods for seed initialization are grouped inside the $'seed\_selection'$ package.

The utils package contains different distance computing functions such as manhattan and euclidean, serializing and deserializing models.

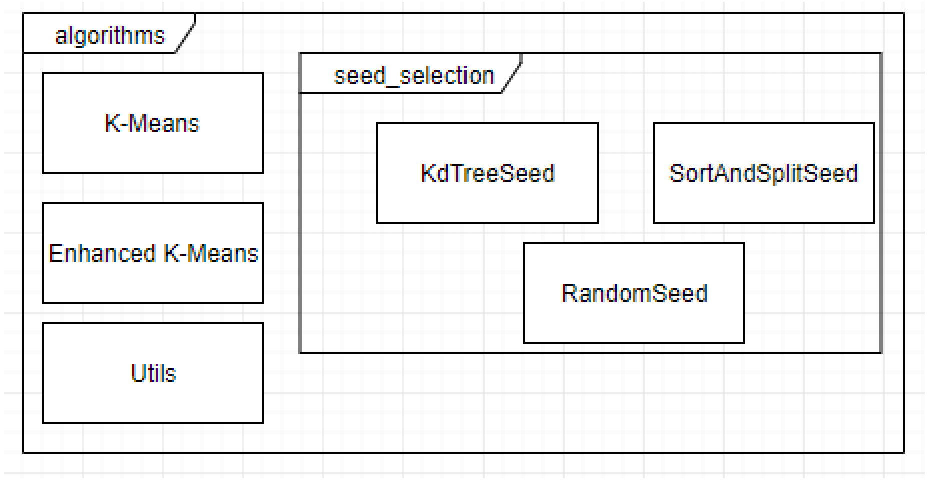

Figure 3: Main packages and modules

3.1 Basic K-Means

A pseudocode for the basic algorithm is given in figure 1. The algorithms iterates over two steps: mean assignment and mean update, which roughly correspond to the inner for loops.

```
select random seeds if not provided;
while $assignment\_change == true$ do
    assignment_change = false;
    for point p in dataset do
        find nearest mean;
        map to nearest mean;
        if nearest mean != previous nearest mean then
            assignment_change = true;
        end
    end
    for point p in dataset do
        update partial sum for each feature of the mean assigned to p;
        update count of points for each mean;
    end
    compute new means from partial sums;
end
```

Algorithm 1: Basic K-Means algorithm

3.2 Enhanced K-Means

The Enhanced K-Means is very similar to the basic implementation. The main difference is the use of a new data structure that maintains the previous distance from each point to its assigned mean.

```
for point p in dataset do
    assigned_mn = the mean assigned to p;
    dist_to_mean = distance(p, assigned_mn);
    if dist_to_mean >prev_dist[p] then
        compute distances to all other means;
        find the closest and update mapping if necessary;
    end
end
```

Algorithm 2: The updated mean assignment function

The updates in the basic algorithm can be observed in figure 2. The distance computation is skipped if the mean moved closer during the last update phase.

3.3 Sort and split

There are three main steps to this algorithm. The first is normalization and it consists of finding the minimum values for each feature, marking the features which need to be normalized (those that have a negative value for at least one of the points in the dataset) and the normalization (subtracting the minimum value from the marked features). The second and third steps are sorting the initial dataset by normalized distance to origin and partitioning the sorted set into k groups respectively. The seeds are chosen as the middle point from each of the discovered groups.

3.4 K-d trees

The K-d tree algorithm is split into three modules which can be reused or replaced depending on the needs of the programmer.

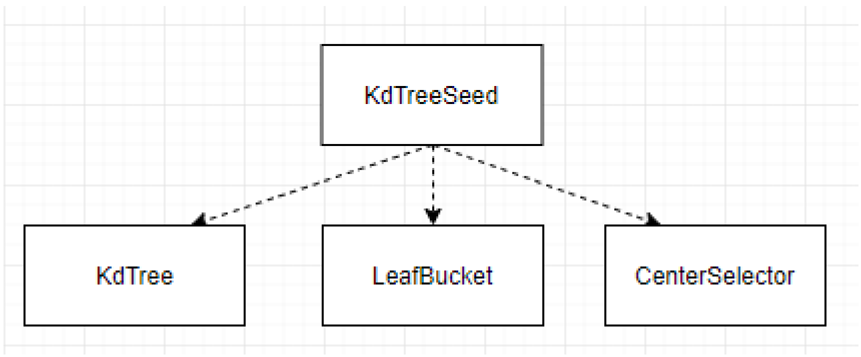

Figure 4: Overview of the Kd-tree seeding module

The KdTree class encapsulates the algorithm used for building a general K-d tree taking as input a set of n-dimensional points and the maximum size of a leaf (by default, 1).

A LeafBucket is initialized with the points that were assigned by the K-d tree algorithm to leaves and contains methods for computing statistics over them like volume and density.

The CenterSelector encapsulates the strategy used to determine the initial seeds using the statistics and information provided by each leaf bucket. The default strategy is to select the mean of the bucket with the highest density first and then to pick by a weight which is the product of density and distance to other bucket means.

References

[1] "How much data do we create every day? the mind-blowing stats everyone should read." `https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#7f96d4a060ba`. Accessed: 2018-11-11.

[2] A. Fahim, A.-B. M.Salem, F. A. Torkey, and M. Ramadan, "Efficient enhanced k-means clustering algorithm," *Journal of Zhejiang University*, vol. 7, no. 10, pp. 1626–1633, 2006.

[3] Y. Madhu, S. Rao Pathakota, and S. , "Enhancing k-means clustering algorithm with improved initial center," *International Journal of Computer Science and Information Technologies*, vol. 1, pp. 121 – 125, 01 2010.

[4] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, Sept. 1975.

[5] S. J. Redmond and C. Heneghan, "A method for initialising the k-means clustering algorithm using kd-trees," *Pattern Recognition Letters*, vol. 28, no. 8, pp. 965 – 973, 2007.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and ÃĽdouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, 2011.

[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science and Engineering*, vol. 9, pp. 90 – 95, 2007.