K-Means Clustering with Seeding and Constraints

Introduction

A very common approach to classify unsupervised data is to use the K-Means clustering algorithm. It is an easy algorithm, which makes it popular, but in its standard form it has its drawbacks. For example, the initialization can become an issue. If the cluster centroids are initialized in a bad (random) location, the result can become very wrong. Also it is an issue with the K-Means clustering algorithm to choose the number of clusters. There are more complex versions of K-means clustering to solve the issue of the amount of clusters, but that version is significantly more computationally heavy [1].

In the paper *Semi-supervised Clustering by Seeding* [2] the standard K-Means clustering algorithm is compared to a K-Means clustering algorithm with seeding and also with another K-Means clustering algorithm with seeding and constraints. This will be repeated here with some small differences as instead of a spherical approach of measuring distance/similarity, the standard Euclidean distance will be used.

K-Means Clustering

This method is an iterative process, where K clusters are defined/partitioned from a dataset. The K centroids of the clusters are relocated after each iteration, by locally minimizing the averaged squared distance between the data samples and the centroids.

For a set of data samples X={x_1, x_2, … , x_N}, $x_i \in \mathbb{R}^d$, the algorithm create K partitions $\{X_l\}^K_{l=1}$ of X so that if {u_1, u_2, … , u_K} represents the K cluster centroids, the following function is minimized:

$$\mathcal{J}_{Kmeans} = \sum_{l=1}^{K} \sum_{x_i \epsilon X_l} ||x_i - u_l||^2$$

Seeded K-Means

This method is used to better handle the issue of initializing the centroids and it is appropriate to use in the presence of noisy seeds (compared to Constrained K-Means).

Instead of initializing the centroids random one can use seeding. Then subsets, or seed-sets, are partitioned and used to calculate the initializing centroids.

From the earlier mentioned set of data samples X, the seeding-sets are generated by picking approximately every tenth sample. Then total set of initial seeds can be written as $S = \bigcup_{l=1}^{K} S_l$.

Then each centroid to be used as initializing centroids for the K-Means clustering is the averaged mean of each seed-set. The output is K disjoint partitioning $\{X_l\}^K_{l=1}$ of X, so that the K-Means objective function is optimized.

Constrained K-Means

This method is appropriate when the seeded labels are noise-free or when the user wants the labels of the seeded data to change.

The same initialization as in the Seeded K-Means clustering algorithm is used. The difference is that the cluster membership (the classification) of the data samples in the seed-sets are not re-computed. Only the labels of the non-seeded data samples are re-estimated and classified.

Implementation

The standard K-Means clustering algorithm consists of six functions. Adding to that is one function for plotting the classified samples and one function for evaluation. The other two algorithms have one and two more functions, respectively.

The implemented functions are the following:

- **kmeans**
- **getNumFeatures**
- **getRandomCentroids**
- **shouldStop**
- **getLabels**
- **getCentroids**
- **plotClusters**
- **evaluate**
- **seeding**
- **constraint**

When a dataset has been chosen, the program will (for the standard K-Means clustering algorithm) first randomly initialize K number of centroids. Then it enters a while-loop that only ends if the objective function is optimized, that is that the centroids stop moving, or if it takes too many iterations (1000 in this case).

For the Seeded K-Means clustering algorithm the dataset is divided into K subsets and the averaged mean of each subset is the initialized centroid. This is done recursively by calling on the function **kmeans** while in the function **seeding**. After that it is proceeded as the method described above.

For the Constrained K-Means clustering algorithm the function **seeding** is also used, and the output from that function that is interesting are the samples that belong to the seed-sets. Those samples (the labels of the samples) are kept untouched and not re-estimated in the function **constraint**, but only the samples that isn't in any seed-set. In this way the function **kmeans** is also used recursively here.

The most important parameters to tune for best result (which are highly case-dependent) are K (the number of clusters, a main issue with K-Means in general) and the parameter *sub_rate* in the function **seeding** (controls how large each seed-set should be).

Results

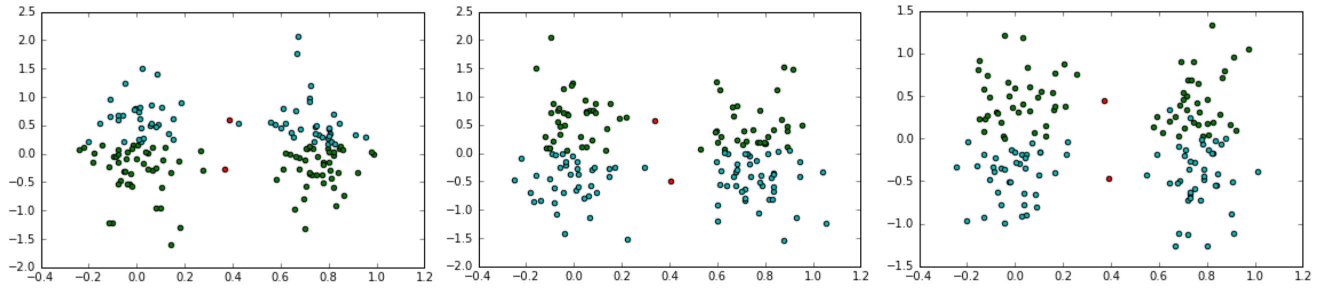

Figure 1 a) Standard K-Means bad results, 0.051 seconds. b) Seeded K-Means bad results, 0.048 seconds. c) Constrained K-Means bad results, 0. 056 seconds.

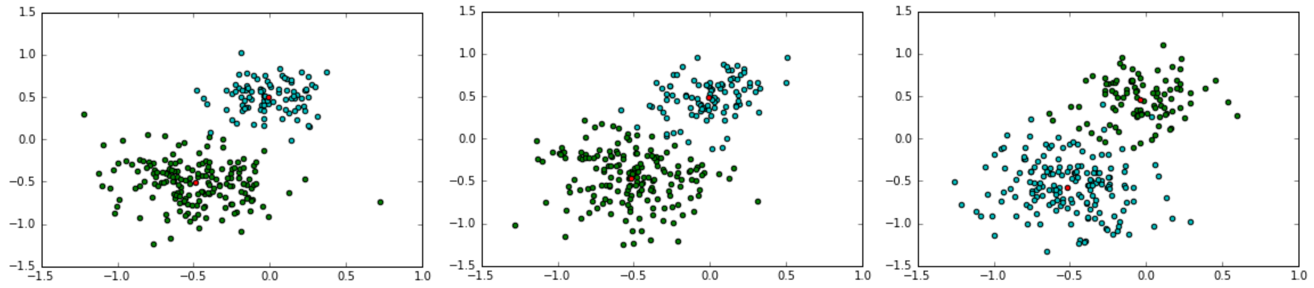

Figure 2 a) Standard K-Means good results, 0.060 seconds, 2 wrong classified labels. b) Seeded K-Means good results, 0.089 seconds, 7 wrong classified labels. c) Constrained K-Means good results, 0.088 seconds, 13 wrong classified labels.

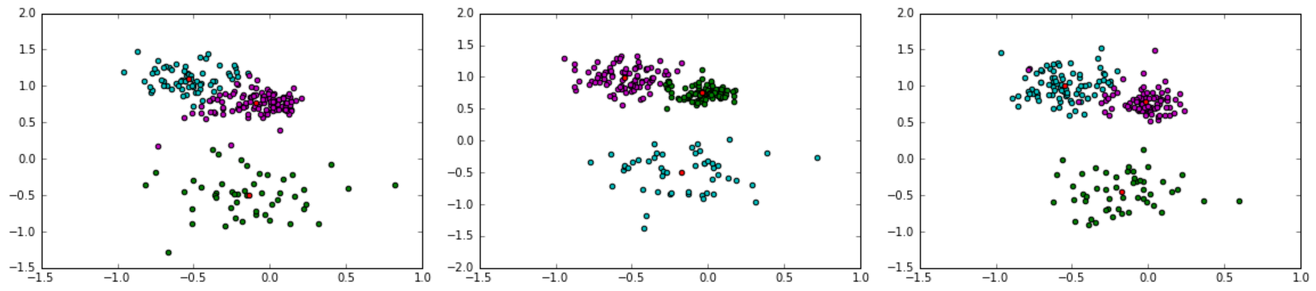

Figure 3 a) Standard K-Means OK results, 0.045 seconds, K=3 for 2 clusters. b) Seeded K-Means OK results, 0.074 seconds, K=3 for 2 clusters. c) Constrained K-Means OK results, 0.088 seconds, K=3 for 2 clusters.

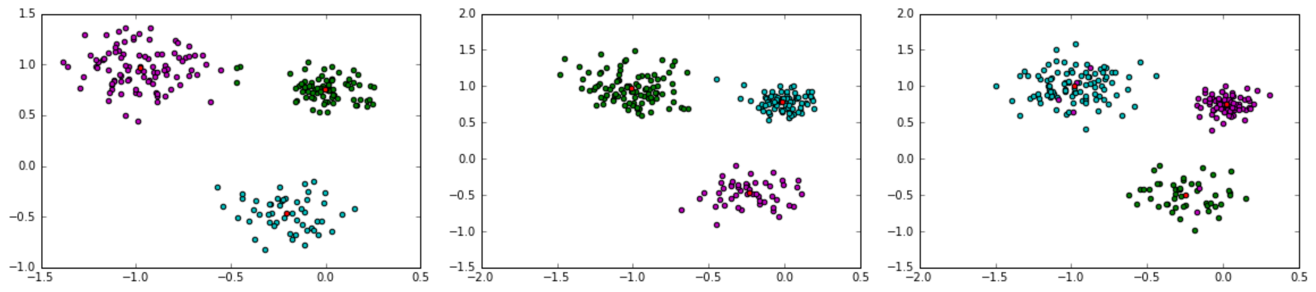

Figure 4 a) Standard K-Means good results, 0.057 seconds, 3 clusters, 3 wrong classified labels. b) Seeded K-Means good results, 0.048 seconds, 3 clusters, one wrong classified label. c) Constrained K-Means good results, 0.043 seconds, 3 clusters, six wrong classified labels.

Conclusion and Evaluation

The reason of investigating other variants of the standard K-Means clustering algorithm is because of the issue that can arise when initializing the centroids randomly. This problem can be seen in *Figure 1 a)*, though it can also be seen in *Figure 1 b)* and *Figure 1 c)* that the other variants here doesn't help. This is because the seeding-sets are also initialized randomly. By changing the fractions of the seeds, the result is changing, as can be seen in [2]. But to avoid the problem, the initializing centroids must be chosen in a cleverer way, e.g. as in *K-Means++* [3], or use another clustering method, e.g. *GMM* [4].

From the results of the 2D-datasets and some investigation it can be shown that the Constrained K-Means clustering algorithm perform worst, then the standard K-Means clustering algorithm and the best performance is achieved from the Seeded K-Means clustering algorithm. It is hard to show that just from a couple of test-runs, which is provided in this report, but after several runs the conclusion is that the Seeded K-Means clustering algorithm is the most stable method with best results. The standard K-Means clustering algorithm is the fastest one.

As has been shown in [2], the Constrained K-Means clustering algorithm performs well/better for higher dimensional data. The drawback from this algorithm here is that it will, because of the constraints, have wrong classified samples if the seeding-sets aren't chosen very luckily (as can be seen in the figures).

What is interesting in *Figure 4* is that in *Figure 4 c)* there are more wrong classified labels than in the other two algorithms, but the wrong labels are within the clusters and not as in the other algorithms, that the border of the clusters are miss-classified. This can be an advantage for the Constrained K-Means clustering algorithm, depending on the case.

Referenser

[1] J. Bejar, ”Clustering,” i *Unsupervised Machine Learning and Data Mining*, 2016, p. 36.

[2] A. B. a. R. M. Sugato Basu, "Semi-supervised Clustering by Seeding," *Prodeedings of the 19th International Conference on Machine Learning,* pp. 19-26, 2002.

[3] "Wikipedia," 21 December 2016. [Online]. Available: https://en.wikipedia.org/wiki/K-means%2B%2B. [Accessed 29 December 2016].

[4] ”Wikipedia,” 18 November 2016. [Online]. Available: https://en.wikipedia.org/wiki/Mixture_model. [Använd 29 December 2016].