k-means++ : The Advantages of Careful Seeding

David Arthur,Segei Vassilvitskii

SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete Algorithms, Pages 1027-1035, New Orleans, Louisiana-January 07-09,2007

Farhad Mohsin, Jun Wang, Shaunak Basu

6 December, 2018

Introduction

BACKGROUND:

- The k-means clustering problem: Given a set of n data points in R^d, choose k centers, with the objective of minimizing the distance between a data point and it's closest center.
- Having an exact solution to this problem is NP hard.

Lloyd's algorithm(k-means)

(i) Choose initital k-centers, $C = c_1, c_2, ..., c_k$ arbitrarily from n data points

(ii) For each center c_i where $i \in \{1, 2, ..., k\}$ set cluster C_i to be the set of data points, which are closer to c_i than c_j where $c_i \neq c_j$

(iii) For each C_i set the center of mass of all points as $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ and update c_i in C

(iv) Repeat (ii) and (iii) until C doesn't change

Lemma: Let S be a set of data points with center of mass $c(S)$ and let z be an arbitrary point. Then $\sum_{x \in S} ||x - z||^2 - \sum_{x \in S} ||x - c(S)||^2 = |S|.||c(S) - z||^2$

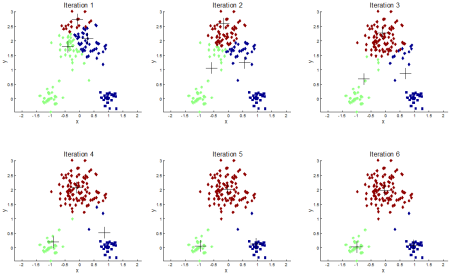

Figure 1: Lloyd's Algorithm at successive iterations(source:https://apandre.wordpress.com/visible-data/cluster-analysis/)

Lloyd's algorithm is bound to terminate as the number of different clusters possible is k^n

Disadvantage: The algorithm does not bound the total squared distance between each point and its closest cluster.

k-means++ Algorithm

Difference in approach from Lloyd's algorithm: Choose a starting center. Remaining $(k-1)$ centers are chosen probabilistically by the assignment of weights.
ALgorithm:

(i) Select a center c_1 uniformly from the data points

(ii) Choose point x_i as the next center with probability $\frac{D(x_i)^2}{\sum_{x_j \in n} D(x_i)^2}$ where $D(x_i)$ is the shortest distance of x_i from the centers already chosen.

(iii) Repeat step (ii) for $i = \{2, ...k\}$

(iv) Proceed as standard k-means

Randomization is in the Seeding - D^2 Distribution

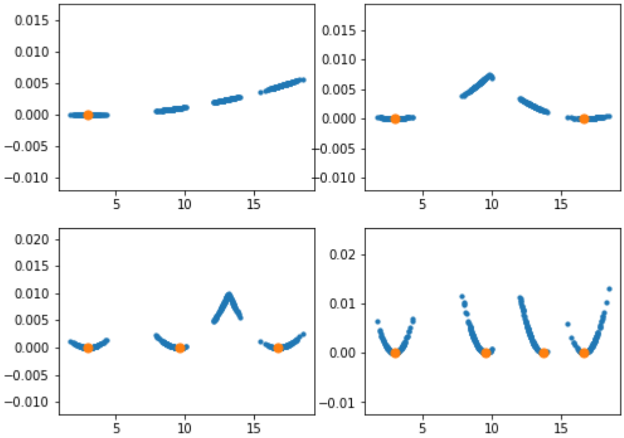

Figure 2: D^2 Distribution demonstrated for 1 dimension

Necessary definitions

- **Optimal Clustering**(C_{OPT}): The optimal clustering for a data set.
- **Potential Function**: $\phi = \sum_{x \in n} min_{c \in C} ||x - c||^2$

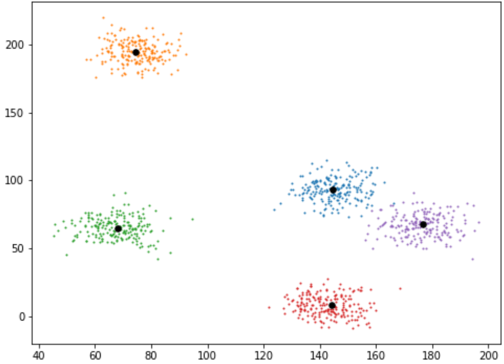

Figure 3: Optimal clustering for five clusters in 2D

Bound provided by k-means++

k-means++ is $O(log\ k)$-competitive.

Theorem: For any set of data points, $\mathbb{E}[\phi] \leq 8(ln\, k + 2)\phi_{OPT}$ where ϕ is the potential function for a k-means++ clustering and ϕ_{OPT} is for optimal clustering.

Making Sense of the Bound

- **Lemma**: Let A be an arbitrary cluster in C_{OPT}, and let C be the clustering with just one center, which is chosen uniformly at random from A. Then $\mathbb{E}[\phi(A)] = 2\phi_{OPT}(A)$ where ϕ_{OPT} is the minimum of the total distance squared between a point and its nearest center.

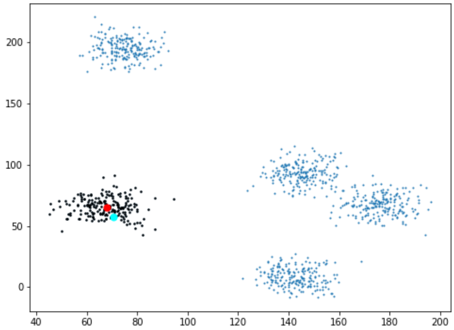

Figure 4: Choosing the first enter

Making Sense of the Bound, contd.

- **Lemma**: Let A be an arbitrary cluster in C_{OPT}, and let C be an arbitrary clustering. If we add a random center to C from A chosen with D^2 weighting, then $\mathbb{E}[\phi(A)] \leq 8\phi_{OPT}(A)$

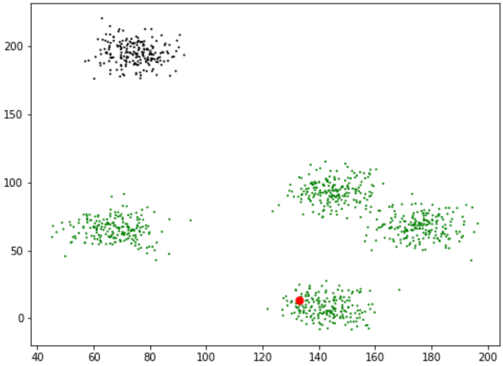

Figure 5: Adding an arbitrary center

Making Sense of the Bound, contd.

- **Lemma**: Let C be an arbitrary clustering. Choose $u > 0$ "uncovered" clusters from C_{OPT} and let X_u denote the set of points in these clusters. Also let $X_c = X - X_u$. Now suppose we add $t \leq u$ random centers to C, chosen with D^2 weighting. Let C' denote the resulting clustering, and let ϕ' denote the corresponding potential. Then,

$$\mathbb{E}[\phi'] = \Big(\phi(X_c) + 8\phi_{OPT}(X_u)\Big)(1 + H_t) + \frac{u-t}{u}\phi(X_u)$$

Making Sense of the Bound, contd.

- We first choose an initial center from cluster A in C_{OPT}.
- Then we choose $k-1$ new centers using D^2 weighting, which indicates $t = u = k-1$, which gives from the last lemma

$$\mathbb{E}[\phi'] = \Big(\phi(A) + 8\phi_{OPT} - 8\phi_{OPT}(A)\Big)(1 + H_{k-1})$$
$$\leq 8(\ln k + 2)\phi_{OPT}$$

De-randomized variant of k-means++

What if there was no randomness?

(i) Select a center c_1 uniformly from the data points.

(ii) Calculate $D(x_i)$ for each point x_i, the shortest distance of x_i from the centers already chosen. Choose the point with highest $D(x_i)$ as the next center.

(iii) Repeat step (ii) for $i = \{2, ...k\}$

(iv) Proceed as standard k-means

Experimental results

For most synthetic datasets, kmeans++ performs much better than kmeans in terms of ϕ, and our kmeans++ variant performs almost the same with kmeans++.

- Setting: $n = 10^4,\ d = 5,\ k = 10,\ \sigma = 10$
- kmeans ϕ=989419.045992, runtime=0.281306 s.
- kmeans++ ϕ=213627.757702, runtime=1.348053 s.
- kmeans++ variant ϕ=213627.757702, runtime=1.379518 s.

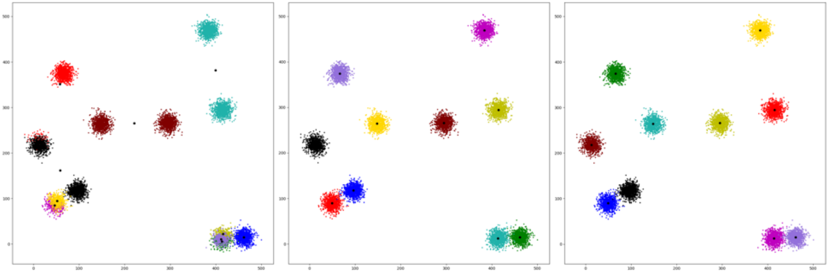

Figure 6: clusters of kmeans, kmeans++, and kmeans++ variant

Experimental results

For cases where most clusters have small variance except one, kmeans++ performs significantly better than our kmeans++ variant.

- Setting: $n = 10^4, \ d = 5, \ k = 10, \sigma_1 = 50, \sigma_{-1} = 1$
- kmeans ϕ=856045.859940, runtime=0.282743 s.
- kmeans++ ϕ=349221.124867, runtime=1.346147 s.
- kmeans++ variant ϕ=518611.857328, runtime=1.399079 s.

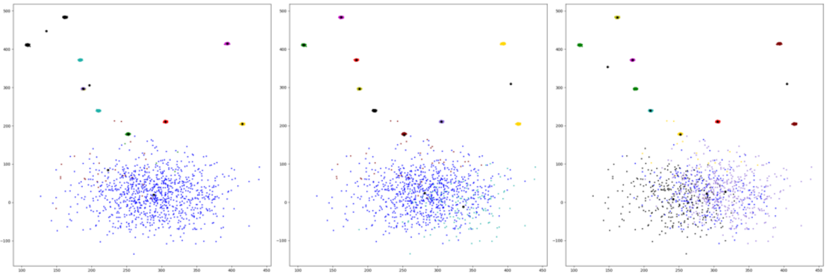

Figure 7: clusters of kmeans, kmeans++, and kmeans++ variant

Experimental results

But experiment shows kmeans++ does not always perform better than our kmeans++ variant.

- Setting: $n = 10^4,\ d = 5,\ k = 25,\ \sigma = 8$
- kmeans ϕ=561295.161033, runtime=0.277098 s.
- kmeans++ ϕ=205674.565783, runtime=2.965102 s.
- kmeans++ variant ϕ=169804.431694, runtime=2.919852 s.

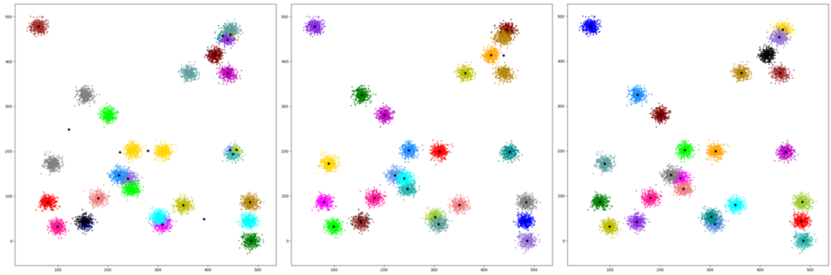

Figure 8: clusters of kmeans, kmeans++, and kmeans++ variant

Real-life Example - Cloud Dataset

- Setting: $n = 1024$
- kmeans ϕ=77252.384281, runtime=0.028508 s.
- kmeans++ ϕ=72780.661743, runtime=0.136611 s.
- kmeans++ variant ϕ=79133.180087, runtime=0.136551 s.

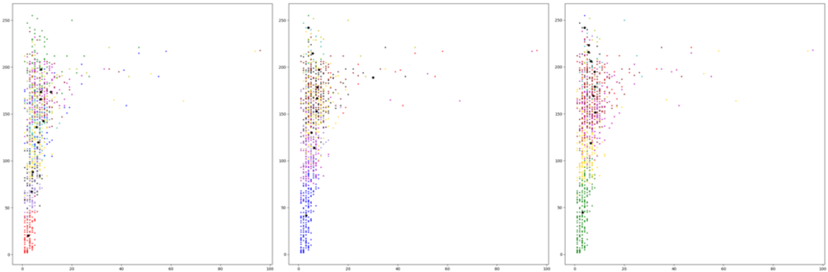

Figure 9: clusters of kmeans, kmeans++, and kmeans++ variant