K-Means++: The Advantages of Careful Seeding

PAPER BY DAVID ARTHUR & SERGEI VASSILVITSKII
PRESENTATION BY STEVEN ROHR
What is it and why do we need it?

“By Augmenting k-means with a very simple, randomized seeding technique, we obtain an algorithm that is $\Theta(\log k)$-competitive with the optimal clustering.”

Arthur & Vassilvitskii claim (often dramatic) improved accuracy and speed. Also focuses on improving cases where Lloyd’s method typically does poorly.
How are we defining “better”? 

As this is a derivative of Lloyd’s method, we are still using k-means cost as a measure of “good”.

- $k$-means cost $= \sum (\text{Dist}(\text{point, clusterCenter})^2)$

Finding the optimal clustering with respect to k-means cost is NP-hard, so we settle for good.

Typically several dozen (100 is common) runs of Lloyd’s method are done and the result with the lowest k-means cost is presented as output.

K-means++ promises equal or lower k-means cost in often far less time.

$O(\log k)$-competitive* (proven, but we won’t get into proofs here).

*Just for selecting the points(?)
How does it work?

Possible initial centers are chosen with probability according to the “contribution to the overall potential”, designated \( \phi \), and each following point \( c_i \) is chosen with probability \( \frac{D(x)^2}{\sum_{x \in X} D(x)^2} \) until we have the number of desired centers (this is called \( D^2 \) seeding).

- \( D(x) \) is the shortest distance from this point to the closest existing center.

Lloyd’s method then continues as normal.
What?

Basically, each new center is chosen probabilistically based off how close it is to existing centers.

Points further from existing centers have a higher chance of being chosen (which point chosen is still up to RNG, we’re just weighting the odds in our favor).

Think of it as a compromise between truly random centers and Furthest Centroids (where only the first point is random at all).
Experimental Results

### K-Means Cost

<table>
<thead>
<tr>
<th>k</th>
<th>Average $\phi$</th>
<th>Minimum $\phi$</th>
<th>Average $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k-means</td>
<td>k-means++</td>
<td>k-means</td>
</tr>
<tr>
<td>10</td>
<td>$1.365 \cdot 10^5$</td>
<td>$1.174 \cdot 10^5$</td>
<td>0.12</td>
</tr>
<tr>
<td>25</td>
<td>$4.233 \cdot 10^4$</td>
<td>$1.914 \cdot 10^4$</td>
<td>0.90</td>
</tr>
<tr>
<td>50</td>
<td>$7.750 \cdot 10^3$</td>
<td>$1.474 \cdot 10^1$</td>
<td>2.04</td>
</tr>
</tbody>
</table>

Table 1: Experimental results on the *Norm25* dataset ($n = 10000$, $d = 15$). For k-means, we list the actual potential and time in seconds. For k-means++, we list the percentage *improvement* over k-means: $100\% \cdot \left(1 - \frac{k\text{-means}^{++} \text{ value}}{k\text{-means} \text{ value}}\right)$.

### Time

<table>
<thead>
<tr>
<th>k</th>
<th>Average $\phi$</th>
<th>Minimum $\phi$</th>
<th>Average $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k-means</td>
<td>k-means++</td>
<td>k-means</td>
</tr>
<tr>
<td>10</td>
<td>$7.921 \cdot 10^3$</td>
<td>$6.284 \cdot 10^3$</td>
<td>0.08</td>
</tr>
<tr>
<td>25</td>
<td>$3.637 \cdot 10^3$</td>
<td>$2.550 \cdot 10^3$</td>
<td>0.11</td>
</tr>
<tr>
<td>50</td>
<td>$1.867 \cdot 10^3$</td>
<td>$1.407 \cdot 10^3$</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Table 2: Experimental results on the *Cloud* dataset ($n = 1024$, $d = 10$). For k-means, we list the actual potential and time in seconds. For k-means++, we list the percentage *improvement* over k-means.
Say we have a set of points which we want to cluster:
Pictures?: Lloyd’s method

Points are randomly selected as centers with equal probability \( \frac{1}{|\text{PointsNotChosen}|} \).

This is like a fair game of roulette.

Imagine there are numbers on the wheel corresponding to the data points. No betting on only Black or Red.
Pictures?: Lloyd’s method

Points are randomly selected as centers with equal probability \( \frac{1}{|\text{PointsNotChosen}|} \).

This is like a fair game of roulette.

Sadly the colors don’t always work out in this analogy.
Pictures?: Lloyd’s method

Points are randomly selected as centers with equal probability \( \frac{1}{|PointsNotChosen|} \).

This is like a fair game of roulette.
Pictures?: Lloyd’s method

Unfortunately “Fair” doesn’t mean “Good outcome”, as any gambler (or D&D player) will tell you.
“It explains the Phantom Menace.”
Pictures?: K-Means++

Points are still selected at random, but are now weighted according to proximity to other centroids.
Pictures?: K-Means++

Points are still selected at random, but are now weighted according to proximity to other centroids.

The result is not like a fair game of roulette.
Pictures?: K-Means++

Points are still selected at random, but are now weighted according to proximity to other centroids.

The result is not like a fair game of roulette.

A not-so-fair roulette wheel.
Now with proper color alteration.
Pictures?: K-Means++

This gives us a reasonably high chance of not choosing horrible initial centers, though the chance still exists (and thus multiple runs are still required).
K-Means++ vs. Other algorithms

Being a means algorithm, it has near linear runtime (much better than linkage algorithms which are near $O(n^3)$).

Vs. Furthest Centroids

- Furthest centroids always chooses the next centroid to be furthest from all current centroids.
- K-Means++ probabilistically chooses centers based off distance.
- End result: K-Means++ is faster, however both algorithms generally have very similar results.
Conclusions

Random center Lloyd’s methods are fast, but the traditional approach uses a brute-force approach to choosing points, and effectively leaves everything up to random chance.

By adding a concept of “weight” to the random selection we can dramatically improve performance by being smarter about how we choose initial centers.