10 k-Means Clustering

Probably the most famous clustering formulation is k-means. This is the focus today. Note: k-means is not an algorithm, it is a problem formulation.

k-Means is in the family of *assignment based clustering*. Each cluster is represented by a single point, to which all other points in the cluster are "assigned." Consider a set X, and distance $\mathbf{d} : X \times X \to \mathbb{R}_+$, and the output is a set $C = \{c_1, c_2, \ldots, c_k\}$. This implicitly defines a set of clusters where $\phi_C(x) = \arg\min_{c \in C} \mathbf{d}(x, c)$. Then the *$k$-means clustering problem* is to find the set C of k clusters (often, but not always as a subset of X) to

$$\text{minimize} \sum_{x \in X} \mathbf{d}(\phi_C(x), x)^2.$$

So we want every point assigned to the closest center, and want to minimize the sum of the squared distance of all such assignments.

Recall, there are other variants:

- the *k-center clustering problem*: minimize $\max_{x \in X} \mathbf{d}(\phi_C(x), x)$
  This was covered in **L9.4**
- the *k-median clustering problem*: minimize $\sum_{x \in X} \mathbf{d}(\phi_C(x), x)$

10.1 Lloyd's Algorithm

When people think of k-means, they usually think of the following algorithm. It is usually attributed to Lloyd from a document in 1957, although it was not published until 1982 [9].

Algorithm 10.1.1 Lloyd's Algorithm for k-Means Clustering

```
Choose $k$ points $C \subset X$          [...arbitrarily?]
repeat
  For all $x \in X$, find $\phi_C(x)$ (closest center $c \in C$ to $x$)
  For all $i \in [k]$ let $c_i = \text{average}\{x \in X \mid \phi_C(x) = c_i\}$
until The set $C$ is unchanged
```

If the main loop has R rounds, then this take roughly Rnk steps (and can be made closer to $Rn \log k$ with faster nearest neighbor search in some cases).

But what is R?

- It is finite. The cost $(\sum_{x \in X}(\mathbf{d}(x, \phi_C(x))^2))$ is always decreasing, and there are a finite (precisely, $\binom{n}{k} = O(n^k)$) number of possible distinct cluster centers. But it could be exponential in k and d (the dimension when Euclidean distance used).
- However, usually $R = 10$ is fine.
- Smoothed analysis: if data perturbed randomly slightly, then $R = O(n^{35}k^{34}d^8)$ [2]. This is "polynomial," but still ridiculous.
- If all points are on a grid of length M, then $R = O(dn^4M^2)$. But thats still way too big.

Lesson: there are crazy special cases that can take a long time, but usually it works. Recall:

When data is easily cluster-able, most clustering algorithms work quickly and well.
When is not easily cluster-able, then no algorithm will find good clusters.

Sometimes there is a good k-means clustering, but it is not found by Lloyd's algorithm. Then we can choose new centers again (with randomness), and try again.

How do we initialize C? The goal is to get one point from each final cluster. Then it will converge quickly.

- Random set of k points. By coupon collectors, we know that we need about $k \log k$ to get one in each cluster.
- Randomly partition $X = \{X_1, X_2, \ldots, X_k\}$ and take $c_i = \textsf{average}(X_i)$. This biases towards "center" of the full set X (by Chernoff-Hoeffding).
- Gonzalez algorithm [6] (for k-center). This may bias too much to outlier points.

Recent algorithm by Arthur and Vassilvitskii [3] called k-means++.

Algorithm 10.1.2 k-Means++ Algorithm

```
Choose $c_1 \in X$ arbitrarily. Let $C_1 = \{c_1\}$.
(In general let $C_i = \{c_1, \ldots, c_i\}$.)
for $i = 2$ to $k$ do
    Choose $c_i$ from $X$ with probability proportional to $\mathbf{d}(x, \phi_{C_{i-1}}(x))^2$.
```

As Algorithm 10.1.2 describes, the algorithm is like Gonzalez algorithm, but is not completely greedy.

How accurate is Lloyd's algorithm for k-means? It can be arbitrarily bad.
Theory algorithm: Gets $(1+\varepsilon)$-approximation for k-means in $2^{(k/\varepsilon)^{O(1)}} nd$ time [8].
But k-means++ is $O(\log n)$-approximate (or 8-approximate if data is well-spaced) [3]. Can then be refined with k-means, if desired.

10.2 Problems with k-Means

- The key step that makes Lloyd's algorithm so cool is $\textsf{average}\{x \in X\} = \arg\min_{c \in \mathbb{R}^d} \sum_{x \in X} \|c - x\|^2$. But this only works with $\mathbf{d}(x, c) = \|x - c\|_2$.

  As an alternative, can enforce that $C \subset X$. Then choose each c_i from $\{x \in X \mid \phi_C(x) = c_i\}$ that minimizes distance. But slower.

- Is effected by outliers more than k-median clustering. Can adapt Lloyd's algorithm, but then step two (recentering) is harder: Called "Fermet-Weber problem,"[10, 5] and can be approximated with gradient descent.

- Enforces equal-sized clusters. Based on distance to cluster centers, not density.

  One adaptation that perhaps has better modeling is the EM formulation: Expectation-Maximization. It models each cluster as a Gaussian distribution G_i centered at c_i.

  - For each point $x \in X$, find cluster c_i with largest probability of containing that point.
  - For each cluster, find best fit Gaussian G_i with $c_i = \textsf{average}\{x \in X \mid \phi_C(x) = c_i\}$, but estimated variance from data.

  This can also allow for non-uniform Gaussians, but first taking PCA of data in cluster, and then estimating variance along each PCA axis. Can be made more robust with regularization.

10.3 Speeding-Up k-Means

- First run Lloyds (or k-means++) on random sample of points (of size $n' \ll n$). Then given good estimate of centers, run on full set (will hopefully be close to converged).

- Run a one-pass algorithm (streaming, covered later) getting $O(k \log k)$ clusters. Reduce to k clusters at end, but merging extra clusters [1].

  Can use another streaming trick where there are a hierarchy of clusters of recent subsets representing geometrically increasing size [7].

- A recent algorithm combines these ideas to make k-means++ somewhat scalable with some added approximation error [4].

Bibliography

[1] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *Advances in Neural Information Processing Systems*, 2009.

[2] David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of ACM*, 58(5):19, 2011.

[3] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[4] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *VLDB Journal*, 2012.

[5] Prosenjit Bose, Anil Maheshwari, and Pat Morin. Fast approximations for sums of distances, clustering and the fermat–weber problem. *Comput. Geom. Theory Appl.*, 24(3):135–146, 2003.

[6] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[7] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[8] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimension. In *Proceedings 45th IEEE Symposium on Foundations of Computer Science*, 2004.

[9] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[10] Yehuda Vardi and Cun-Hui Zhang. A modified Weiszfeld algorithm for the Fermat-Weber location problem. *Mathematical Programming*, 90(3):559–566, 2001.