9 Assignment-based 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. We will also discuss other variants, noteably the k-center clustering algorithm.

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 $\max_{x \in X} \mathbf{d}(\phi_C(x), x)$
- the *k-median clustering problem*: minimize $\sum_{x \in X} \mathbf{d}(\phi_C(x), x)$
  The *k-mediod* variant is similar, but restricts that the centers C must be a subset of P.

9.1 Gonzalez Algorithm for k-Center Clustering

Here we want every point assigned to the closest center, and want to minimize the *longest* distance of any such assignment.

Unfortunately, the k-center clustering problem is NP-hard to solve exactly. In fact, it is NP-hard to find a clustering within a factor 2 of the optimal cost!

Luckily, there is an algorithm that achieves this factor 2 approximation, it is quite fast, and it works very well in practice. It is usually attributed to Gonzalez (1985), but it may likely be much older. The lesson is:

Be greedy, and avoid your neighbors!

Algorithm 9.1.1 Gonzalez Greedy Algorithm for k-Center Clustering

```
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
    Set $c_i = \arg\max_{x \in X} \mathbf{d}(x, \phi_{C_{i-1}}(x))$.
```

As Algorithm 9.1.1 describes, the algorithm is to always pick the point in x that is furthest from the current set of centers, and let it also be a center.

In the worst case, this is a 2-approximation to the optimal clustering for the k-center clustering problem. But is often much better in practice.

It only takes time about $kn = O(kn)$. There are k rounds, and each round can be done in about n time. We maintain the map $\phi_{C_i}(x)$ for each x. When a new c_i is found, and added to the set of centers, all n assignments $\phi_{C_i}(x)$ can be updated in linear $O(n)$ time, by checking each distance $\mathbf{d}(x, \phi_{C_{i-1}}(x))$ against

Algorithm 9.1.2 Detailed Gonzalez Greedy Algorithm for k-Center Clustering

```
Choose $c_1 \in X$ arbitrarily, and set $\phi[j] = 1$ for all $j \in [n]$
for $i = 2$ to $k$ do
  $M = 0, \qquad c_i = x_1$
  for $j = 1$ to $n$ do
    if $\mathsf{d}(x_j, c_{\phi[j]}) > M$ then
      $M = \mathsf{d}(x_j, c_{\phi[j]}), \qquad c_i = x_j$
  for $j = 1$ to $n$ do
    if $\mathsf{d}(x_j, c_{\phi[j]}) > \mathsf{d}(x_j, c_i)$ then
      $\phi[j] = i$
```

$\mathsf{d}(x, c_i)$ and switching the assignment if the later is smaller. Then the minimum can be found in the next round on a linear scan (or on the same linear scan).

This works for any metric. However, it biases the choice of centers to be on the "edges" of the dataset. There are heuristic to try to recenter afterwards, but usually not worth it—just use another algorithm instead.

9.2 Lloyd's Algorithm

When people think of the k-means problem, 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 9.2.1 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 $c_i = \textsf{average}\{x \in X \mid \phi_C(x) = c_i\}$
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 $(\sum_{x \in X}(\mathsf{d}(x, \phi_C(x))^2))$ 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 data 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 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.

9.2.1 Initializing 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 $c_i = \mathsf{average}(X_i)$. 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.

k-means++ Since the above approaches all have issues, the accepted best way to seed Lloyd's algorithm has become k-means++, an algorithm by Arthur and Vassilvitskii [3].

Algorithm 9.2.2 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 9.2.2 describes, the algorithm is like Gonzalez algorithm, but is not completely greedy. It iteratively chooses each next center randomly – the further the squared distances is from an existing center, the more likely it is chosen. For a large set of points (perhaps grouped together) which are far from an existing center, then it is very likely that *one (does not matter so much which one)* of them will be chosen as the next center. This makes it likely that any "true" cluster will find some point as a suitable representative.

9.3 Problems with k-Means

- The key step that makes Lloyd's algorithm so cool is $\mathsf{average}\{x \in X\} = \arg\min_{c \in \mathbb{R}^d} \sum_{x \in X} \|c - x\|^2$. 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.

- It 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.

- It tends to 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, see more details below.

9.4 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].

9.5 Mixture of Gaussians

The k-means formulation tends to define clusters of roughly equal size. The squared cost discourages points far from any center. It also, does not adapt much to the density of individual centers.

An extension is to fit each cluster X_i with a Gaussian distribution $\mathcal{G}(\mu_i, \Sigma_i)$, defined by a mean μ_i and a covariance matrix Σ_i. Recall that the pdf of a d-dimensional Gaussian distribution is defined

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{d/2}} \frac{1}{\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

where $|\Sigma|$ is the determinant of Σ. For instance, for $d = 2$, and the standard deviation in the x-direction of X is σ_x, and in the y-direction is σ_y, and their correlation is ρ, then

$$\Sigma = \left[\begin{array}{cc} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{array} \right].$$

Now the goal is, given a parameter k, find a set of k pdfs $F = \{f_1, f_2, \ldots, f_k\}$ where $f_i = f_{\mu_i, \Sigma_i}$ to maximize

$$\prod_{x \in X} \max_{f_i \in F} f_i(x).$$

For the special case where when we restrict that $\Sigma_i = I$ (the identity matrix) for each mixture, then this is equivalent to the k-means problem.

This hints that we can adapt Lloyds algorithm towards this problem as well. To replace the first step of the inner loop, we assign each $x \in X$ to the Gaussian which maximizes $f_i(x)$:

$$\text{for all } x \in X\text{: assign } x \text{ to } X_i \text{ so } i = \arg\max_{i \in 1 \ldots k} f_i(x).$$

But for the second step, we need to replace a simple average with an estimation of the best fitting Gaussian to a data set X_i. This is also simple. First, calculate the mean as $\mu_i = \frac{1}{|X_i|}\sum_{x \in X_i} x$. Then calculate the covariance matrix Σ_i of X_i as the sum of outer products

$$\Sigma_i = \sum_{x \in X_i} (x - \mu_i)(x - \mu_i)^T.$$

Algorithm 9.5.1 EM Algorithm for Mixture of Gaussians

```
Choose $k$ points $S \subset X$                                   *arbitrarily?*
for all $x \in X$: set $w_i(x)$ for $s_i = \phi_S(x)$, and $w_i(x) = 0$ otherwise
**repeat**
  **for** $i \in [1 \ldots k]$ **do**
    Calculate $W_i = \sum_{x \in X} w_i(x)$                        *the total weight for cluster i*
    Set $\mu_i = \frac{1}{W_i} \sum_{x \in X} w_i(x) x$                *the weighted average*
    Set $\Sigma_i = \frac{1}{W_i} \sum_{x \in X} w_i (x - \mu_i)(x - \mu_i)^T$     *the weighted covariance*
  **for** $x \in X$ **do**
    for all $i \in [1 \ldots k]$: set $w_i(x) = f_i(x) / \sum_i f_i(x)$     *partial assignments using $f_i = f_{\mu_i, \Sigma_i}$*
**until** ($\sum_{x \in X} \sum_{i=1}^{k} -\log(w_i(x) \cdot f_i(x))$ has small change)
```

9.5.1 Expectation-Maximization

The standard way to fit a mixture of Gaussians actually uses a soft-clustering.

Each point $x \in X$ is given a weight $w_i = f_i(x) / \sum_i f_i(x)$ for its assignment to each cluster. Then the mean and covariance matrix is estimated using weight averages.

This procedure is the classic example of a framework called *expectation-maximization*. This is an alternate optimization procedure, which alternates between maximizing the probability of some model (the partial assignment step) and calculating the most likely model using expectation (the average, covariance estimating step).

But this is a much more general framework. It is particularly useful in situations (like this one) where the true optimization criteria is messy and complex, often non-convex; but it can be broken into two or more steps where each step can be solved with a (near) closed form. Or if there is no closed form, but each part is individually convex, the gradient descent can be invoked.

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.