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.

9.1 Variants of Assignment-based Clustering

In general we consider the family of *assignment-based clusterings*. 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 $\phi_C(x) = \arg\min_{c \in C} \mathbf{d}(x, c)$. That is cluster $S_j \subset X$ is defined $S_j = \{x \in X \mid \phi_C(x) = c_j\}$.

There are 4 popular (and commonly named) variants – although other versions could be considered as well.

1. 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 assign every point to the closest center, and want to minimize the sum of the squared distance of all such assignments.
The most common algorithm is called *Lloyd's algorithm* which only works when $\mathbf{d}$ is the Euclidean distance (or some very similar geodesic measures).

2. The *k-center clustering problem* finds the center set C of size k, to minimize the furthest assignment, specifically aiming for
$$\text{minimize} \max_{x \in X} \mathbf{d}(\phi_C(x), x).$$
The common algorithm is called *Gonzalez Algorithm* which provides a 2-approximation and applies generally to any metric space. This method is used in diversity maximization.

3. The *k-median clustering problem* just minimizes the sum of assignment costs.
$$\text{minimize} \sum_{x \in X} \mathbf{d}(\phi_C(x), x).$$
If it is probably the most obvious formulation, and is more robust to outliers than k-means or certainly k-center, but there is not a clever and simple iterative method to solve it – like Gonzalez or Lloyds.

4. The *k-medioid* variant is similar to k-median, but restricts that the centers C must be a subset of P. If one is going to restrict $C \subset X$, and wants something more robust to outliers than Gonzalez, then this is the popular formulation. Typically any heuristic will work in any metric space.

9.2 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 in a general metric space!

Luckily, there is an algorithm that achieves this factor 2 approximation, it is quite fast, applies to any metric distance, and it works very well in practice. It is usually attributed to Gonzalez (1985). The lesson is:

Be greedy, and avoid your neighbors!

Algorithm 9.2.1 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 9.2.1 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. And note again that this works for any distance metric $\mathbf{d}$.

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 $\mathbf{d}(x, c_i)$ 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).

Algorithm 9.2.2 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**
  $M = 0, \quad c_i = x_1$
  for $j = 1$ to n **do**
    if $\mathbf{d}(x_j, c_{\phi[j]}) > M$ **then**
      $M = \mathbf{d}(x_j, c_{\phi[j]}), \quad c_i = x_j$
  for $j = 1$ to n **do**
    if $\mathbf{d}(x_j, c_{\phi[j]}) > \mathbf{d}(x_j, c_i)$ **then**
      $\phi[j] = i$

However, it biases the choice of centers to be on the "edges" of the dataset.

9.3 Lloyd's Algorithm (for k-Means Clustering)

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].

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 (for both steps inside the for loop). There are a finite (precisely, $\binom{n}{k} = O(n^k)$) number of possible distinct cluster centers. If each assignment has a cost, it can only visit them in decreasing order – so at most once each. Thus it eventually stops. But it could be exponential in k and d (the dimension when Euclidean distance used).

Algorithm 9.3.1 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 $c_i = \mathsf{average}\{x \in X \mid \phi_C(x) = c_i\}$
until The set C is unchanged

- However, usually $R = 10$ is fine; maybe a bit more if $|X|$ and k are very large.

Lesson: there are intricately created special cases that can cause the algorithm to iterate a surprisingly long time, but usually it works well (but not always). 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.

9.3.1 Lloyd's Algorithm Can Get Stuck In Non-Optimal Solution

There is no guarantee that the iteration finds the optimal assignment when it stops iterating. The cost always decreases, but it might be in a *local optimum*.

This means Lloyd's algorithm does not "solve" the k-means problem.

For this reason, we usually initialize the clusters C in Lloyd's in multiple times (with randomness). The method is re-run. And ultimately the lowest cost solution is returned.

9.3.2 Initializing C

The goal is to get one point from each final cluster. Then it will converge quickly. Here are initial (not totally recommended) options:

- Random set of k points. By coupon collectors, we know that we need about $k \log k$ to get one in each cluster. We could thus oversample $k \log k$ clusters, run Lloyd's, and then merge nearby clusters (like in HAC) until we get k.
  But if the cluster sizes are imbalanced, we would need more than $k \log k$ at initialization.
- 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 (e.g., via Central Limit Theorem).
- Gonzalez algorithm [6] (for k-center). This may bias too much to outlier points. Moreover, it is deterministic, so (given a first center choice) we only get one output – if its not good, we are out of luck.

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 9.3.2 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 9.3.2 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.

One can prove that with enough randomized k-means++ initializations that for any data set, one will find an approximately optimal solution, within a $O(\log n)$ factor.

It works very well in practice, and is now part of the default setting for most libraries that run a method for k-means.

If one wants a stronger $(1+\varepsilon)$-approximation (for any $\varepsilon > 0$), then something like $nk \cdot (\log n)^{(d/\varepsilon)^{O(d)}}$ time is the best known, and is unlikely to be improved (is APX-HARD when $d = \Omega(\log n)$). These approaches can start with the set of k-means++ initializations, and then try to locally refine them by more explicitly considering more complex changes.

9.4 Problems with k-Means

- The key step that makes Lloyd's algorithm so natural and clean 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 $\mathtt{d}(x, c) = \|x - c\|_2$.

  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 this is slower, since it requires checking many choices. And typically, one would then consider the k-medioid formulation if we restrict $C \subset X$.

- It is effected by outliers more than k-median clustering. Can adapt Lloyd's algorithm, but then step two (recentering) is harder: The k-median's associated center step (minimize sum of distances) is called the *Fermet-Weber problem* [10, 5] and is surprisingly complicated – but for the most part, can be approximated with gradient descent.

- k-means tends to enforces equal-sized clusters, when all else equal. It is based on the 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.

Yet, this is still probably the most widely used clustering formulation.

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

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}.$$

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

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

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.

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)

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] 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.

[2] 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.

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

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

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