Unsupervised Learning: Clustering
K-Means Clustering

K-means algorithm with different initial *randomly chosen* centroids.

https://www naftali harris.com/blog/visualizing-k-means-clustering/
Results of k-means algorithm depends on the initialization step.

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
K-Means Clustering Algorithm

- Given $X = \{x_1, \cdots, x_n\}$ where $x_i \in \mathbb{R}^d$
- Goal: partition $X$ into $k$ clusters, $S = \{S_1, \cdots, S_k\}$, $k \leq n$
- While: minimizing intracluster variance

$$\arg \min_S \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2,$$

where $\sum_{x \in S_i} ||x - \mu_i||^2 = |S_i| \text{Var}(S_i)$ and $\mu_i$ is mean of points in $S_i$.

K-Means Clustering Algorithm

1. **Initialization**: Choose a set of initial centroids $c_1^{(1)}, \ldots, c_k^{(1)}$
   - Chose by users, randomly or based on farthest points.

2. **Assignment**: assign each $x \in X$ to its nearest centroid, at step $t \geq 1$:
   
   $$S_i^{(t)} := \{ x : \| x - c_i^{(t)} \|^2 \leq \| x - c_j^{(t)} \|^2 \}, \forall j \neq i \}.$$ 

3. **Update**: Calculate the new centroids of the new clusters,
   
   $$c_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x \in S_i^{(t)}} x.$$ 

4. Repeat steps 2 and 3 until it converges: assignments no longer change. Algorithm does not guarantee convergence to the global optimum.

Differ from K-Means by the *Initialization* step.

High-level idea: spreading out the $k$ initial cluster centroids.

1. Choose one centroid uniformly at random from $X$.
2. For each $x \in X$, compute its distance $d(x)$ to the nearest centroid that has already been chosen.
3. Choose one new centroid from $X$ at random as a new centroid, using a weighted probability distribution where a point $x$ is chosen with probability proportional to $d(x)^2$.
4. Repeat Steps 2 and 3 until $k$ centroids have been chosen.
5. Apply standard $k$-means clustering.

Density-based spatial clustering of applications with noise (DBSCAN)

Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu, 1996

KDD 2014: the test of time award

Density-based clustering algorithm

High-level: groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

High-level idea: spreading out the $k$ initial cluster centroids.

https://en.wikipedia.org/wiki/DBSCAN
• $X$: points to be clustered.
• $\epsilon$: the radius of a neighborhood.
• $\textit{minPts}$: threshold.

Core points, (density-)reachable points and outliers:
• A point $p$ is a \textit{core point} if at least $\textit{minPts}$ points are within distance $\epsilon$ of it (including $p$).
• A point $q$ is directly \textit{reachable} from $p$ if point $q$ is within distance $\epsilon$ from the core point $p$. Points are only said to be directly reachable from core points.
• A point $q$ is reachable from $p$ if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each $p_{i+1}$ is directly reachable from $p_i$. Note that this implies that all points on the path must be core points, with the possible exception of $q$.
• All points not reachable from any other point are \textit{outliers} or noise points.

https://en.wikipedia.org/wiki/DBSCAN
If $p$ is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it.

Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.

https://en.wikipedia.org/wiki/DBSCAN
DBSCAN(D, epsilon, min_points):
    C = 0
    for each unvisited point P in dataset
        mark P as visited
        sphere_points = regionQuery(P, epsilon)
        if sizeof(sphere_points) < min_points
            ignore P
        else
            C = next cluster
            expandCluster(P, sphere_points, C, epsilon, min_points)

expandCluster(P, sphere_points, C, epsilon, min_points):
    add P to cluster C
    for each point P' in sphere_points
        if P' is not visited
            mark P' as visited
            sphere_points' = regionQuery(P', epsilon)
            if sizeof(sphere_points') >= min_points
                sphere_points = sphere_points joined with sphere_points'
            if P' is not yet member of any cluster
                add P' to cluster C

regionQuery(P, epsilon):
    return all points within the n-dimensional sphere centered at P with radius epsilon (including P)
DBSCAN Algorithm

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/
Unsupervised Learning: Clustering: Mapper
High-Dimensional Data Analysis
WhiteBoard
Singh et al. (2007)