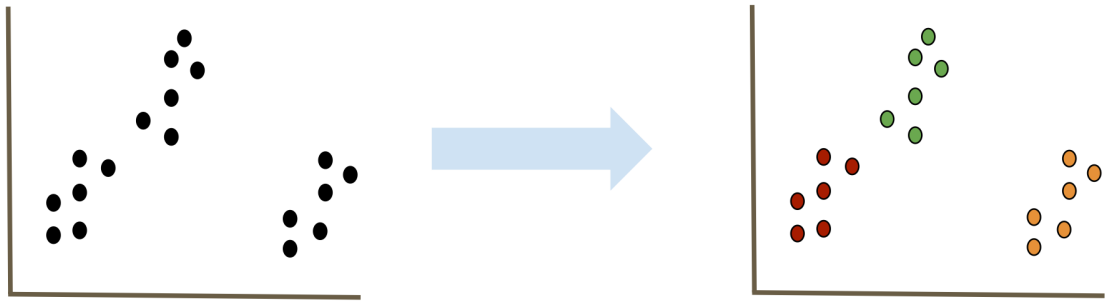
Clustering: K-means

What is a Clustering

- Clustering is a grouping/assignment of objects (data points) such that objects in the same group/cluster are:
- similar to one another
- dissimilar to objects in other groups



Applications

- **Outlier detection / anomaly detection**
  - Data Cleaning / Processing
  - Credit card fraud, spam filter, etc.
- **Filling Gaps in your data**

Questions:

- What does **similar mean?**
- how do we find clustering?
- how do we know if we have found a **good clustering**?

Types of Clusterings

Partitional

Each object belongs to exactly one cluster

Hierarchical

A set of nested clusters organized in a tree

Density-Based

Defined based on the local density of points

Soft Clustering

Each point is assigned to every cluster with a certain probability

Partitional Clustering

Given **n** data points and a number **k** of clusters: partition the **n** data points into **k** clusters.

Given a distance function d, we can find points (not necessarily part of our dataset) for each cluster called **centroids** that are at the center of each cluster.

When **D** is Euclidean, what is the centroid (also called the **center of mass**) of **m points**

{ $x_1, x_2, ..., x_m\}$

Answer: The mean of the points.

Turns out when **d** is Euclidean:

$$\sum_{k}^{K} \sum_{x_i, x_j \in C_k} d(x_i, x_j)^2 = \sum_{k}^{K} |C_k| \sum_{x_i \in C_k} d(x_i, \mu_k)^2$$

K-means

Given $X = \{x_1, ..., x_n\}$ our dataset and **k**

Find **k** points $\{\mu_1, ..., \mu_k\}$ that minimize the **cost function:**

$$\sum_{i}^{k} \sum_{x \in C_i} d(X, \mu_i)$$

K-means-Lloyd's Algorithm

Repeat two steps until convergence (convergent means the centers and clusters will not change anymore (maybe reach the local minimum)

The procedure alternates between two operations.

- Once a set of centroids is available, the clusters are updated to contain the points closest in distance to each centroid.
- Given a set of clusters, the centroids are recalculated as the means of all points belonging to a cluster.

The two-step procedure continues until the assignments of clusters and centroids no longer change.

$$C_k = \{\mathrm{x}_n : ||\mathrm{x}_n - \mu_k|| \leq \text{ all } ||\mathrm{x}_n - \mu_l||\} \qquad (1)$$

$$\mu_k = \frac{1}{C_k} \sum_{\mathrm{x}_n \in C_k} \mathrm{x}_n \qquad (2)$$

This is the code for Lloyd's Algorithm method:

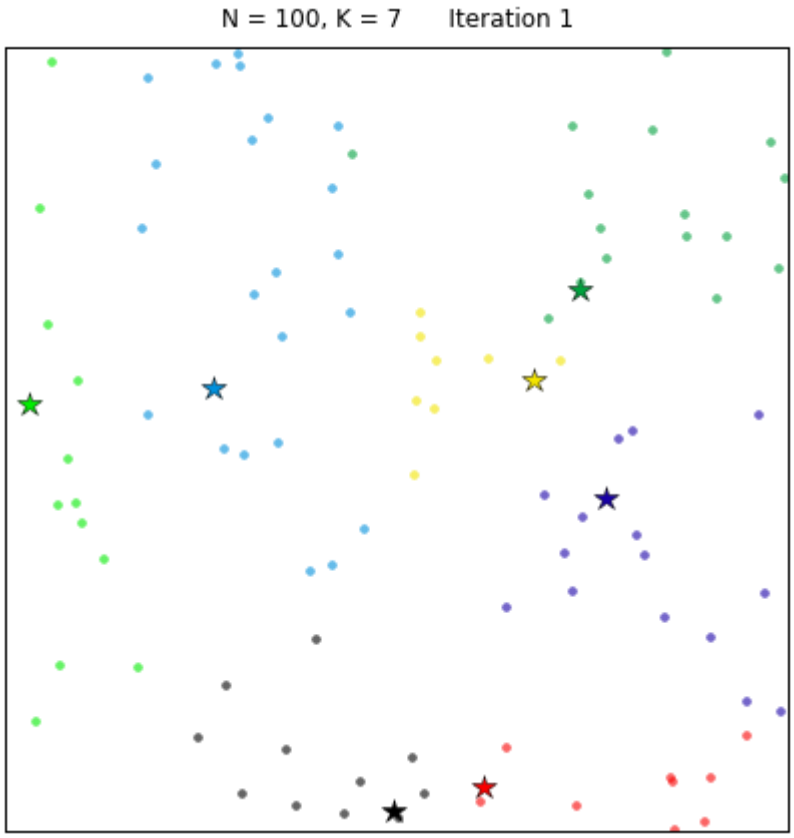
```
import NumPy as np

def cluster_points(X, mu):
    clusters  = {}
    for x in X:
        bestmukey = min([(i[0], np.linalg.norm(x-mu[i[0]])) \
                    for i in enumerate(mu)], key=lambda t:t[1])[0]
        try:
            clusters[bestmukey].append(x)
        except KeyError:
            clusters[bestmukey] = [x]
    return clusters
```

```
def reevaluate_centers(mu, clusters):
    newmu = []
    keys = sorted(clusters.keys())
    for k in keys:
        newmu.append(np.mean(clusters[k], axis = 0))
    return newmu

def has_converged(mu, oldmu):
    return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu])

def find_centers(X, K):
    # Initialize to K random centers
    oldmu = random.sample(X, K)
    mu = random.sample(X, K)
    while not has_converged(mu, oldmu):
        oldmu = mu
        # Assign all points in X to clusters
        clusters = cluster_points(X, mu)
        # Reevaluate centers
        mu = reevaluate_centers(oldmu, clusters)
    return(mu, clusters)
```



The code details: https://datasciencelab.wordpress.com/tag/lloyds-algorithm/

Will this always converge to the optimal solution?

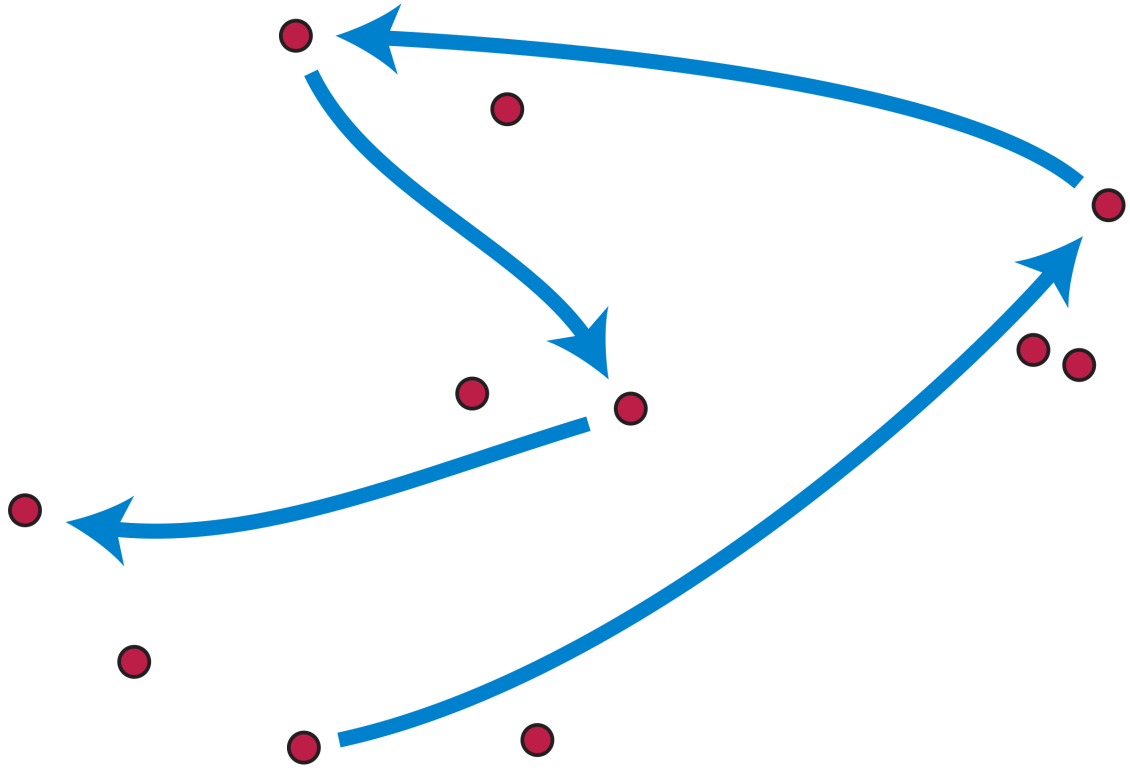
The choice of initial points has a large influence on the resulting clustering

one solution: Run Lloyd's algorithm multiple times and choose the result with the lowest cost.

another solution: Try different initialization methods.

Farthest First Traversal

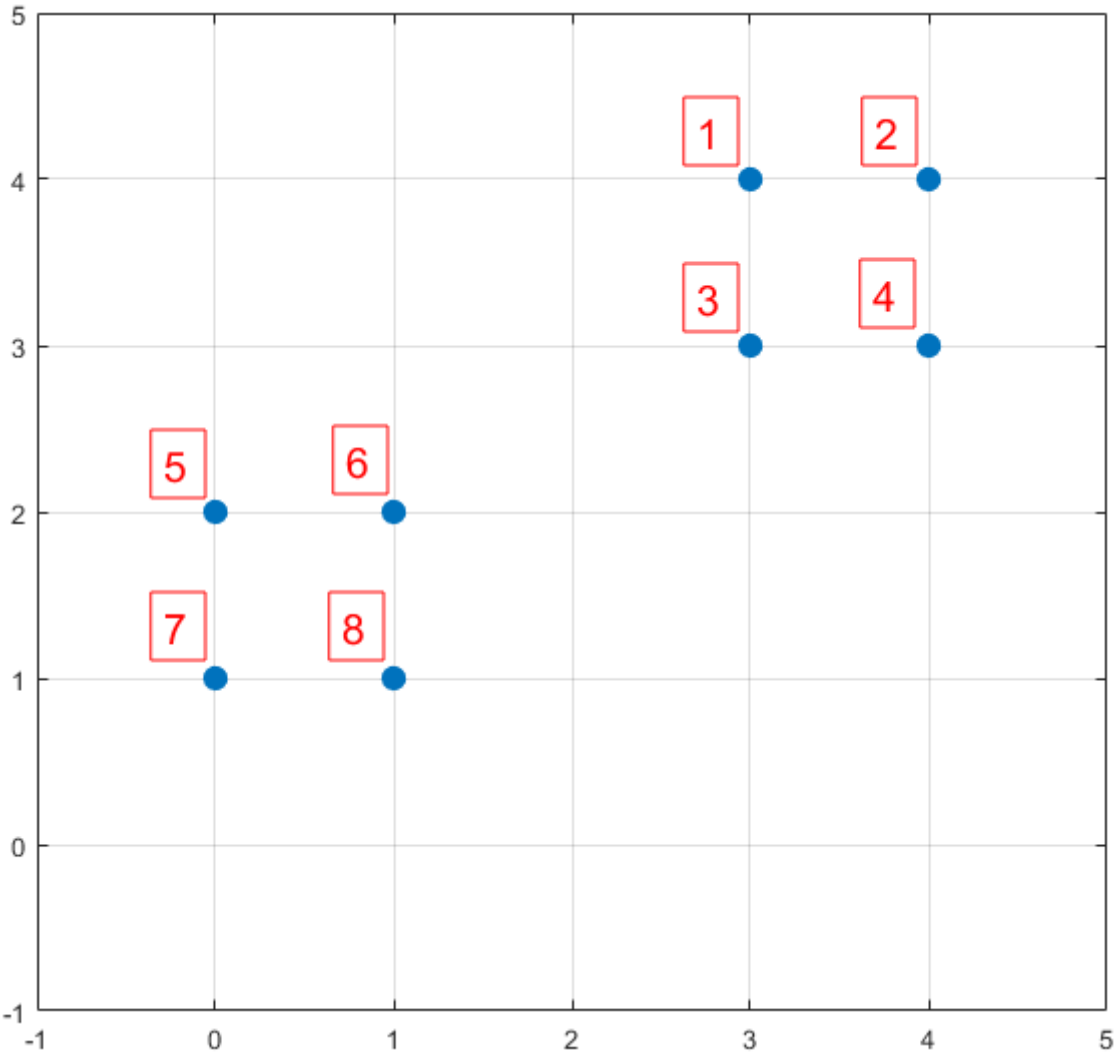
In computational geometry, the farthest-first traversal of a compact metric space is a sequence of points in the space, **where the first point is selected arbitrarily and each successive point is as far as possible from the set of previously-selected points**.



K-means++

https://www.cnblogs.com/yixuan-xu/p/6272208.html

K-means++算法
Step 1: 从数据集中随机选取一个样本作为初始聚类中心c_1；
Step 2: 首先计算每个样本与当前已有聚类中心之间的最短距离(即与最近的一个聚类中心的距离)，用$D(x)$表示；接着计算每个样本被选为下一个聚类中心的概率 $\frac{D(x)^2}{\sum_{x\in X} D(x)^2}$。最后，按照轮盘法选择出下一个聚类中心；
Step 3: 重复第 2 步直到选择出共K个聚类中心；
之后的过程与经典 K-means 算法中第 2 步至第 4 步相同



假设经过算法的步骤一后6号点被选择为第一个初始聚类中心，那在进行步骤二时每个样本的***D(x)***
和被选择为第二个聚类中心的概率如下表所示：

序号	①	②	③	④	⑤	⑥	⑦	⑧
$D(x)$	$2\sqrt{2}$	$\sqrt{13}$	$\sqrt{5}$	$\sqrt{10}$	1	0	$\sqrt{2}$	1
$D(x)^2$	8	13	5	10	1	0	2	1
$P(x)$	0.2	0.325	0.125	0.25	0.025	0	0.05	0.025
Sum	0.2	0.525	0.65	0.9	0.925	0.925	0.975	1

其中的***P(x)***就是每个样本被选为下一个聚类中心的概率。最后一行的***Sum***是概率***P(x)***的累加和，用于轮盘法选择出第二个聚类中心。方法是随机产生出一个0~1之间的随机数，判断它属于哪个区间，那么该区间对应的序号就是被选择出来的第二个聚类中心了。例如1号点的区间为[0,0.2)，2号点的区间为[0.2, 0.525)。

从上表可以直观的看到第二个初始聚类中心是1号，2号，3号，4号中的一个的概率为0.9。而这4个点正好是离第一个初始聚类中心6号点较远的四个点。这也验证了K-means的改进思想：即离当前已有聚类中心较远的点有更大的概率被选为下一个聚类中心。可以看到，该例的K值取2是比较合适的。当K值大于2时，每个样本会有多个距离，需要取最小的那个距离作为***D(x)***。

How to choose the right k?

1. Iterate through different values of k (elbow method)
2. Use empirical / domain-specific knowledge
3. Silhouette scores

K-means Variations

1. K-medians (uses the L1 norm / manhattan distance)
2. K-medoids (any distance function + the centers must be in the dataset)
3. Weighted K-means (each point has a different weight when computing the mean)