Clustering - grouping

- Similar data points in same cluster
- Dissimilar data points in different clusters
- How do we know if we found a good clustering?

Type of clusterings

- Partitional
- Hierarchical
- Density-based
- Soft clustering

Partitional clustering
Partition n data points into k clusters
Inner sum: sum of all pairs of points in the cluster, take the sum of the distances,
Outer sum: distance between clusters

Given a distance function d, we can find points for each cluster called centroids that are at the center of each cluster.

When distance is euclidean, what is the centroid of m points??? - the average of the points

k-means:
Given x our dataset and k
Find k points that minimizes the cost function

When k=1 and k=n this is easy. Why? Every point in the same cluster, or every point in different clusters.

K-means - lloyd's algorithm

1. Randomly pick k centers
2. Assign each point in the dataset to its closest center
3. Compute the new centers as the means of each cluster
4. Repeat 2 & 3 until convergence

Will this algorithm always converge?
Proof by contradiction.
It always converges.

Will this always converge to the optimal solution?
Really depends on the k point that starts with.
No.

Farthest first traversal
If there's an outlier, then it will be problematic

K-means++
Initialize with a combination of the two methods:

1. Start with a random center
2. Let D(x) be the distance between x and the centers selected so far. Choose the next center with probability proportional to D(x)^a

No reason to use k-means over k-means++

Generate random integers and add D(x) tgt

K-mean: limitations
Tend to break large clusters
Does not handle points with density very well.
Does not handle special shape well

How to choose the right k?

1. Iterate through different values of k
2. Use empirical/domain-specific knowledge

```
|
|/
|//
|///
____________
```

K-means variation

- K-medians
- K-medoids + the center in the data set
- Weighted k means

Coding part:

Import random

```
Point_distances = {
        "X": 9,
        "Y": 4,
        "Z": 1,
}
```

N = sum(point_distances.values())

```
def get_point_from_rand(result):
    threshold = 0;
    for p in point_distances:
        threshold += point_distances[p]
        if result < threshold:
            return p
Count = {
    “X”: 0,
    “Y”: 0,
    “Z”: 0,
}
```

```
res = random.randint(1, N)
Point = get_point_from_rand(res)
Count[point] += 1
plt.bar(count.values(), , color=’green’)
plt.show()
```