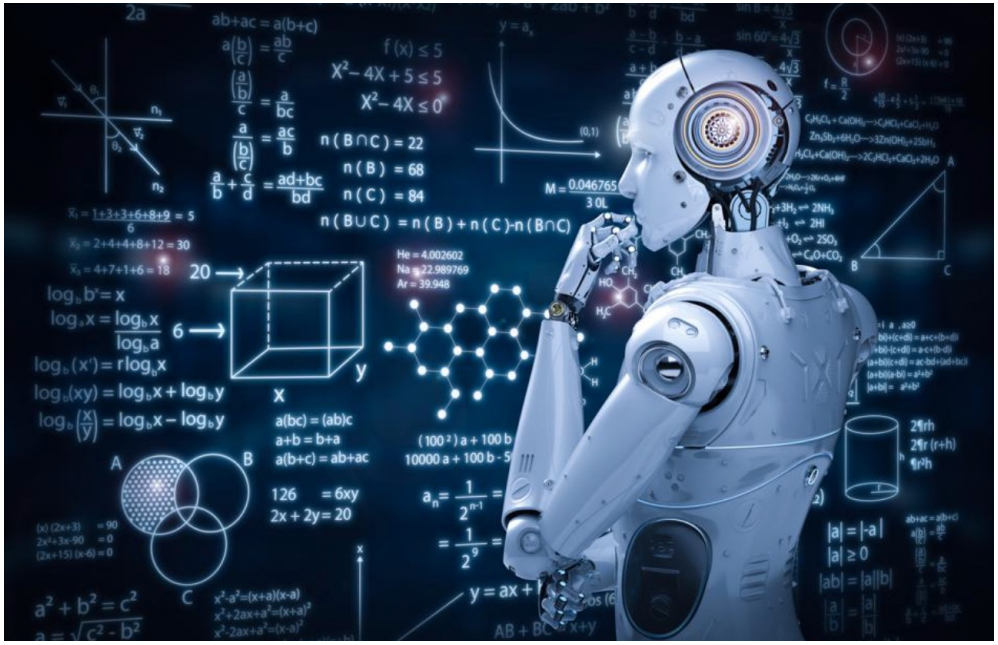
CS 4100: Introduction to AI

Wayne Snyder
Northeastern University

Lecture 14: K-Means concluded; Survey of Clustering; PCA



Unsupervised Learning: Clustering with K-Means

Lloyd's Algorithm for K-Means

Repeat until some termination criterion is met:

1. Randomly* choose the k centroids { c_1, ..., c_k };
2. For each $1 \leq j \leq k$ set the cluster X_j to be the set of points in X which are closest to the center c_j;
3. For each j, update the value of c_j to be the mean of the vectors in X_j.

* NOTE: This is a Hill-Climbing (search) algorithm, where the cost is the squared intra-cluster distances:

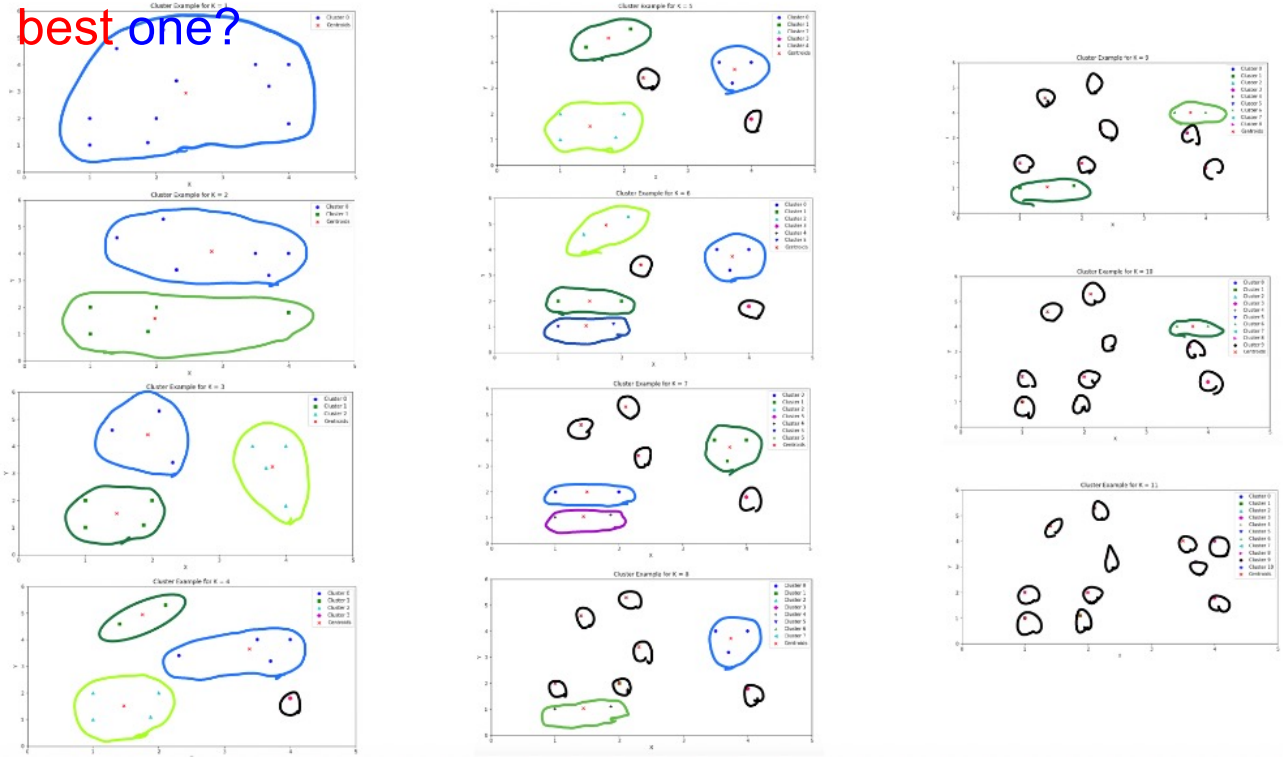
$$\sum_{j=1}^{k} \sum_{x \in X_j} ||x - c_j||_2^2$$

L2 norm: square of distance between points x and c_j.

Unsupervised Learning: Clustering with K-Means

Evaluating K-Means: What should K be????

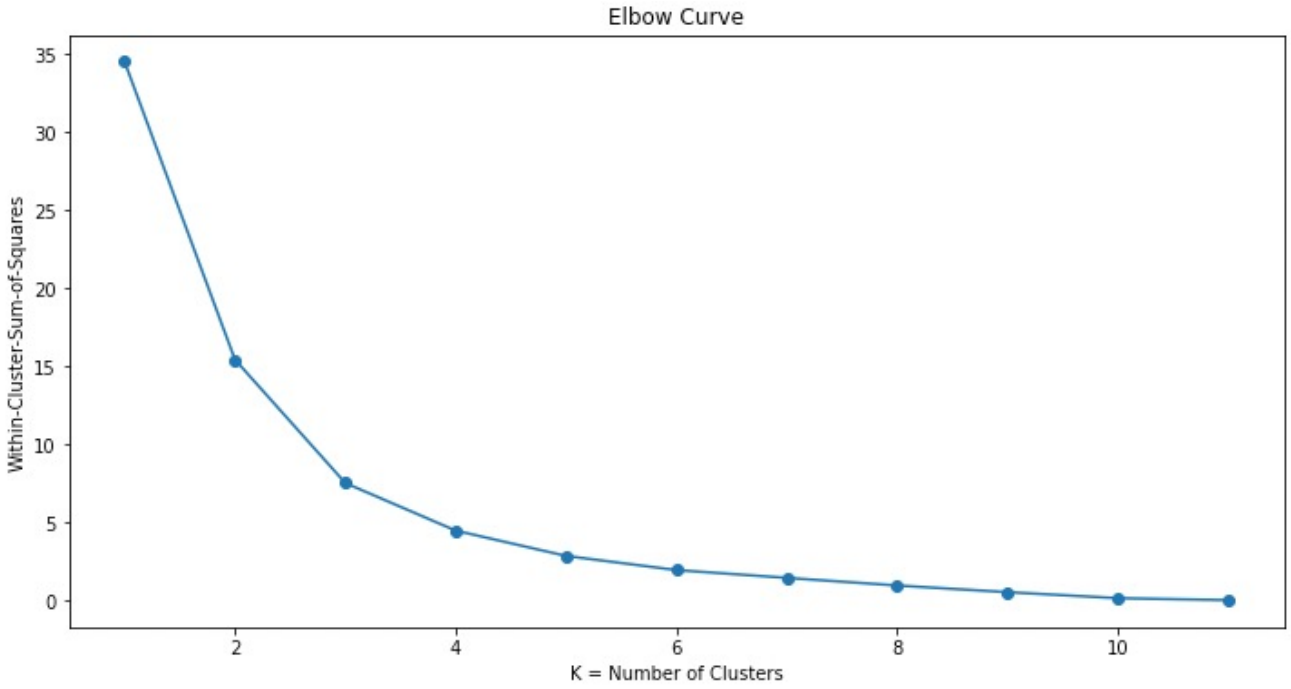
The main problem is that the cost (sum of squared intra-cluster distances) decreases as number of clusters gets smaller! Which is the best one?



Unsupervised Learning: Clustering with K-Means

How to choose the right K??

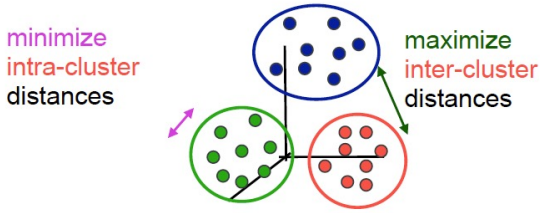
1. Iterate through different values of k (elbow method)
2. Use empirical / domain-specific knowledge: Example: Is there a known approximate distribution of the data? (K-means is good for spherical gaussians)
3. Metric for evaluating a clustering output...



Unsupervised Learning: Clustering with K-Means

Metrics for evaluating clustering results.

Recall our goal: Find a set of centroids such that
1. Similar data points are in the same cluster; and
2. Dissimilar data points are in differenet clusters.



K-Means does a good job on (1) because the cost function tells us that intra-cluster distances will be small overall.

$$\sum_{j=1}^{k} \sum_{x \in X_j} ||x - c_j||_2^2$$

L2 norm: square of distance between points x and c_j.

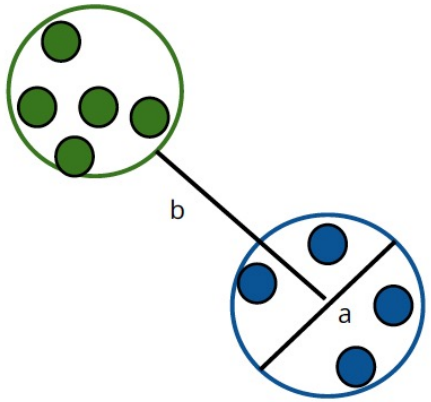
But what about the intra-cluster distance?

Are the cluster we created far away from each other? How far, and relative to what?

The Silhouette Score is a common way to analyse this....

Unsupervised Learning: Clustering with K-Means

Silhouette Score



a: average within-cluster distance
b: average intra-cluster distance

What does it mean for (b-a) to be 0? To be large? To be negative?

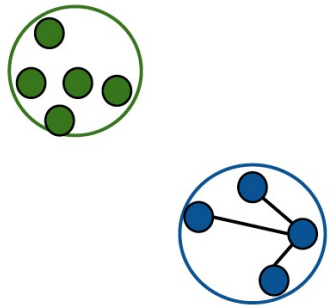
Should we compare (b-a) to be some other value, in order to get a sense of how representative that average value is overall?

The Silhouette Score uses (b-a) / max(a,b).

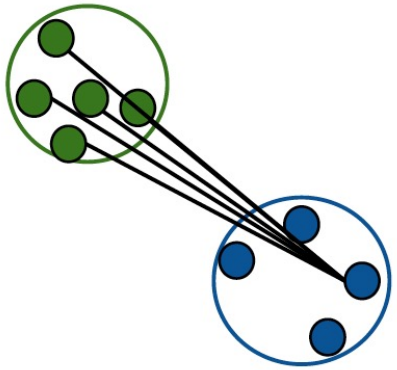
Unsupervised Learning: Clustering with K-Means

For each data point i:

a_i = the mean distance from point i to every other point in its cluster; and



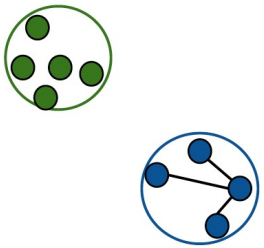
b_i = the smallest mean distance from point i to a point in a different cluster.



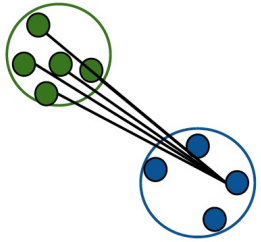
Unsupervised Learning: Clustering with K-Means

For each data point i:

a_i = the mean distance from point i to every other point in its cluster; and



b_i = the smallest mean distance from point i to a point in a different cluster.



The Silhouette Score for each point i is then:

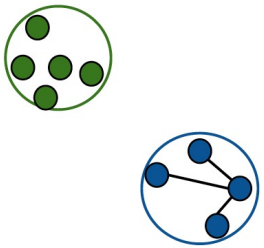
$$s_i = (b_i - a_i) / \max(a_i, b_i).$$

We return the mean of all s_i as a metric for goodness-of-fit for the clustering.

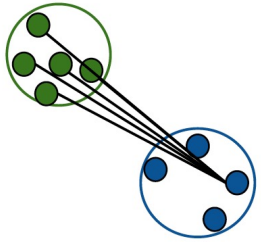
Unsupervised Learning: Clustering with K-Means

For each data point i:

a_i = the mean distance from point i to every other point in its cluster; and



b_i = the smallest mean distance from point i to a point in a different cluster.



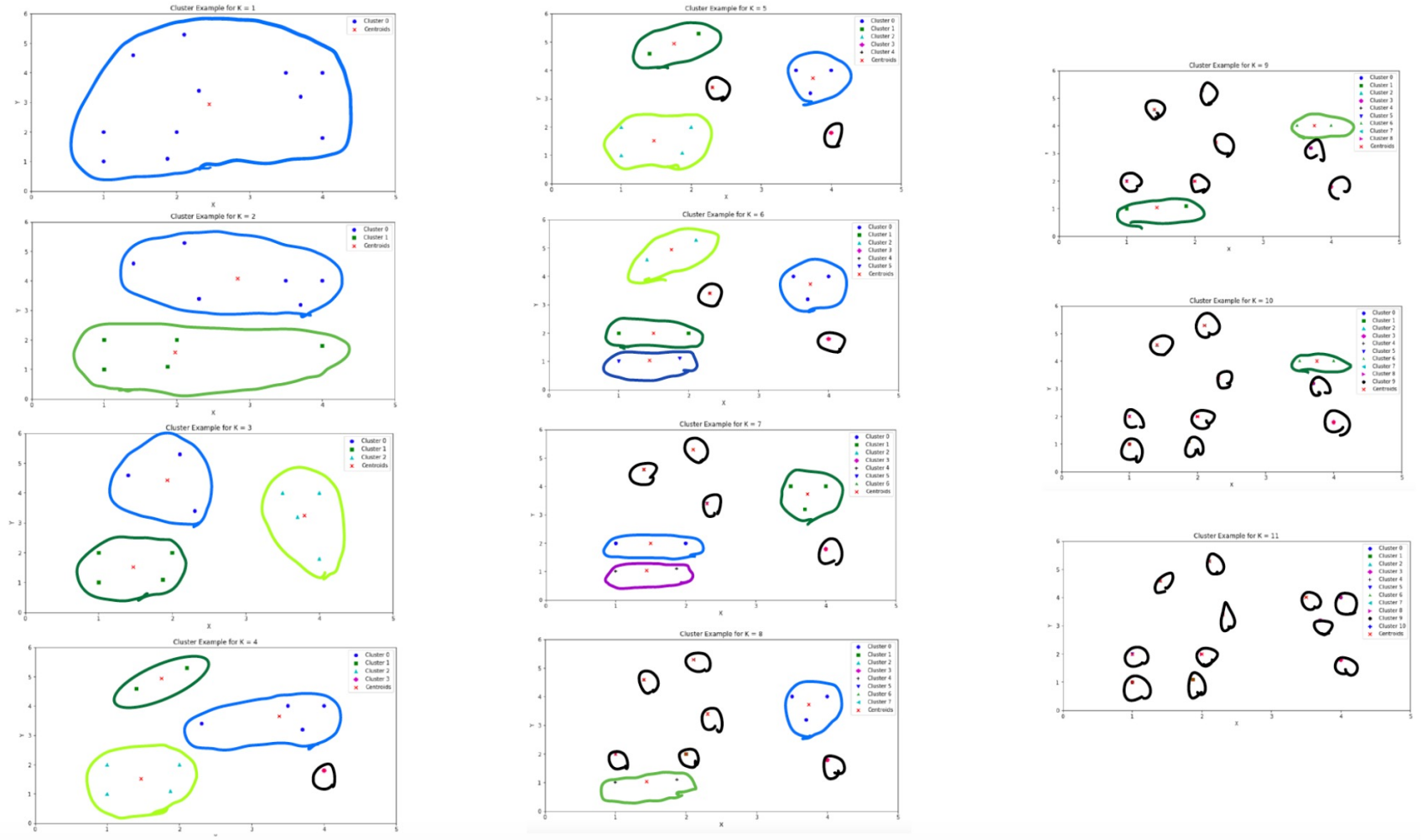
The Silhouette Score for each point i is then:

$$s_i = (b_i - a_i) / \max(a_i, b_i).$$

We return the mean of all s_i as a metric for goodness-of-fit for the clustering.

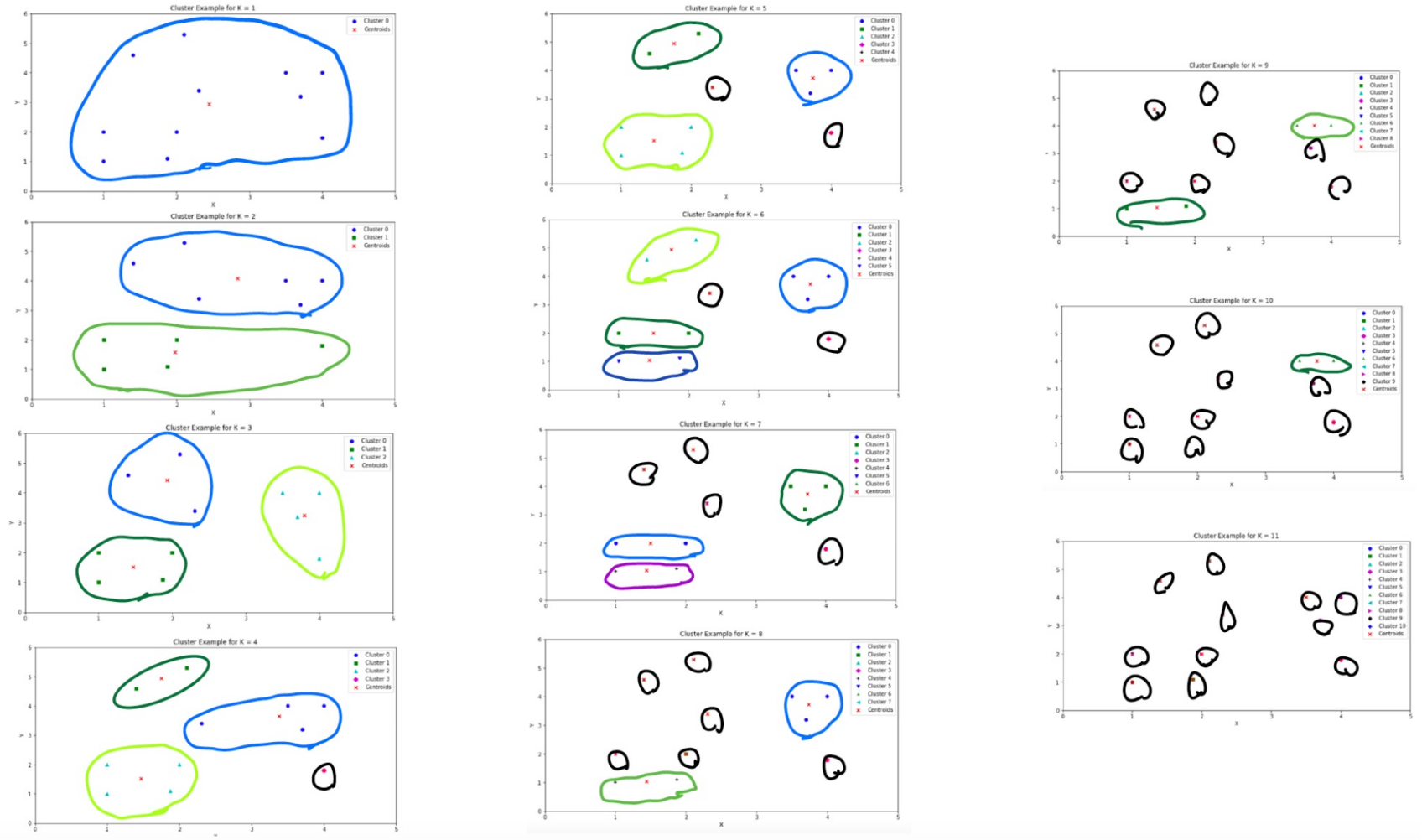
Unsupervised Learning: Clustering with K-Means

Comparing Within-Cluster-Sum-Of-Squares and Silhouette Scores on our example...



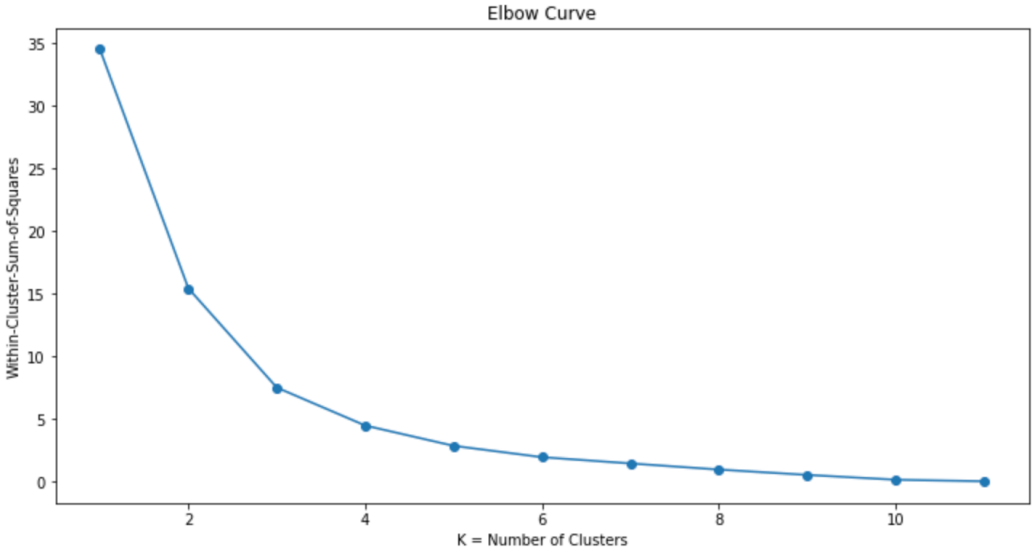
Unsupervised Learning: Clustering with K-Means

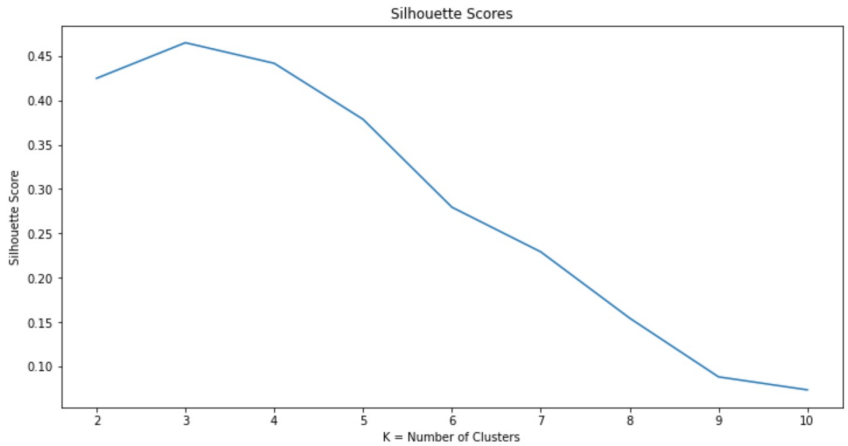
Comparing Within-Cluster-Sum-Of-Squares and Silhouette Scores on our example...

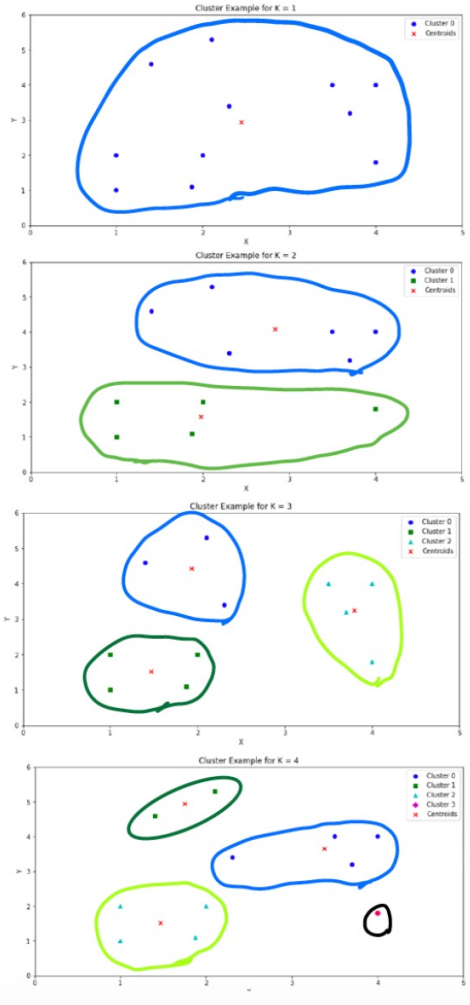


Unsupervised Learning: Clustering with K-Means

Comparing Within-Cluster-Sum-Of-Squares and Silhouette Scores on our example...

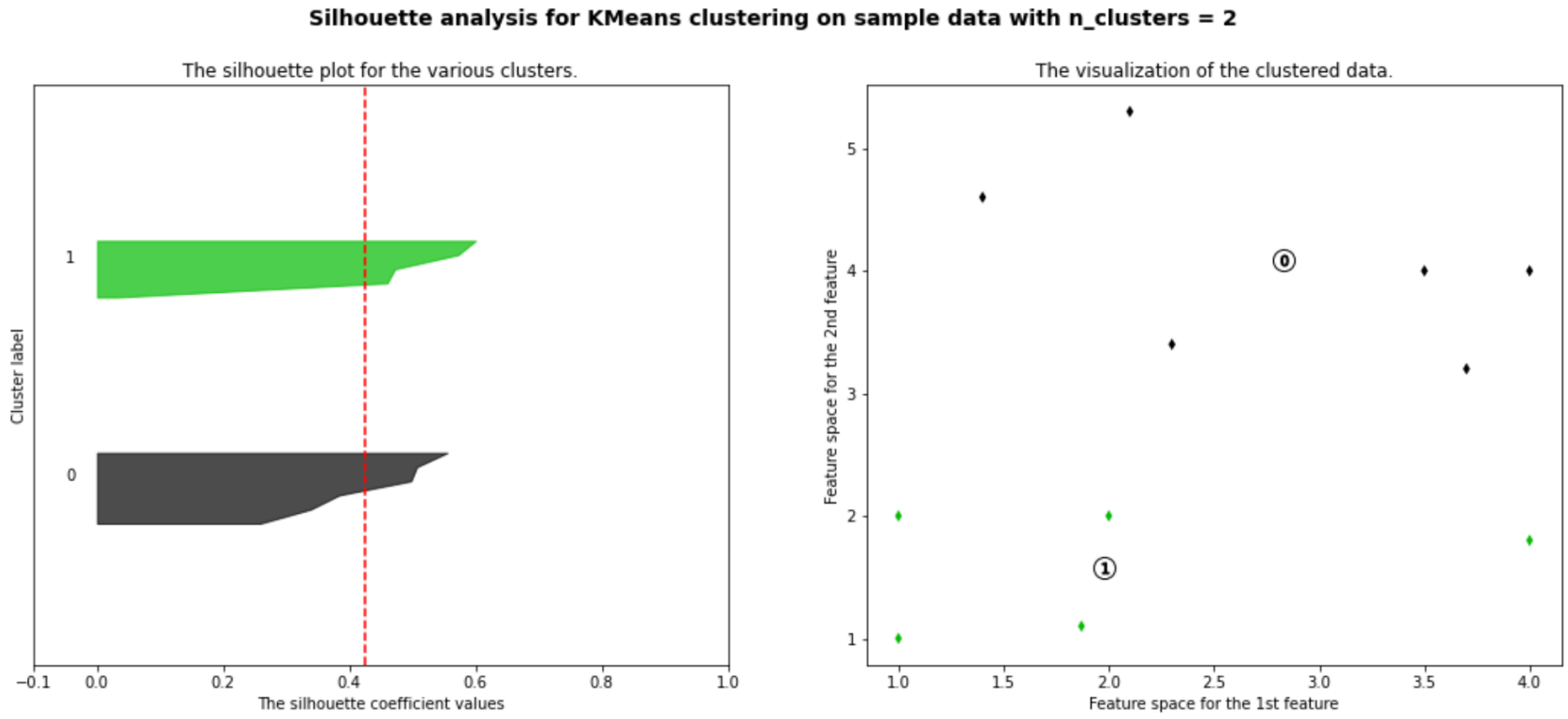






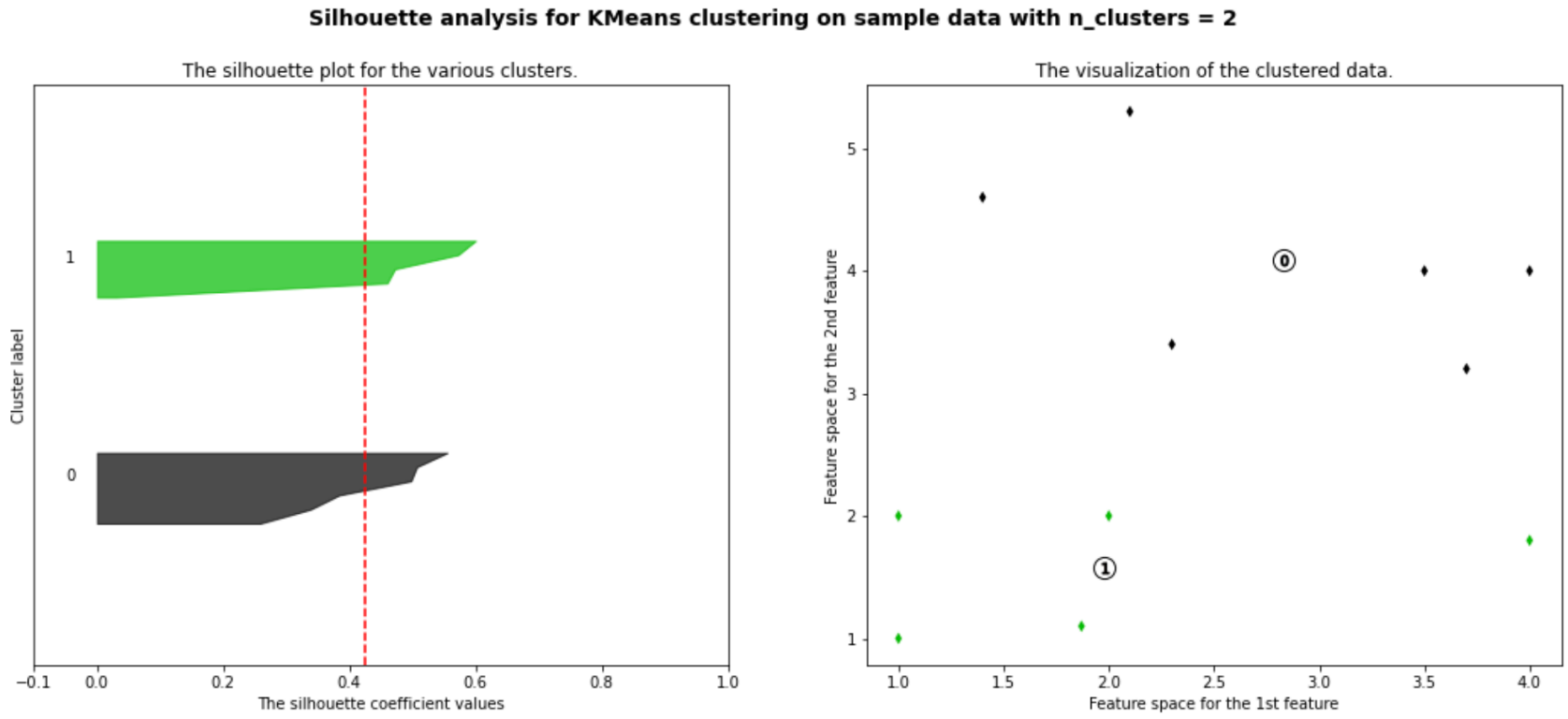
Unsupervised Learning: Clustering with K-Means

Silhouette Scores are often combined with a histogram of the scores of each point in each cluster:



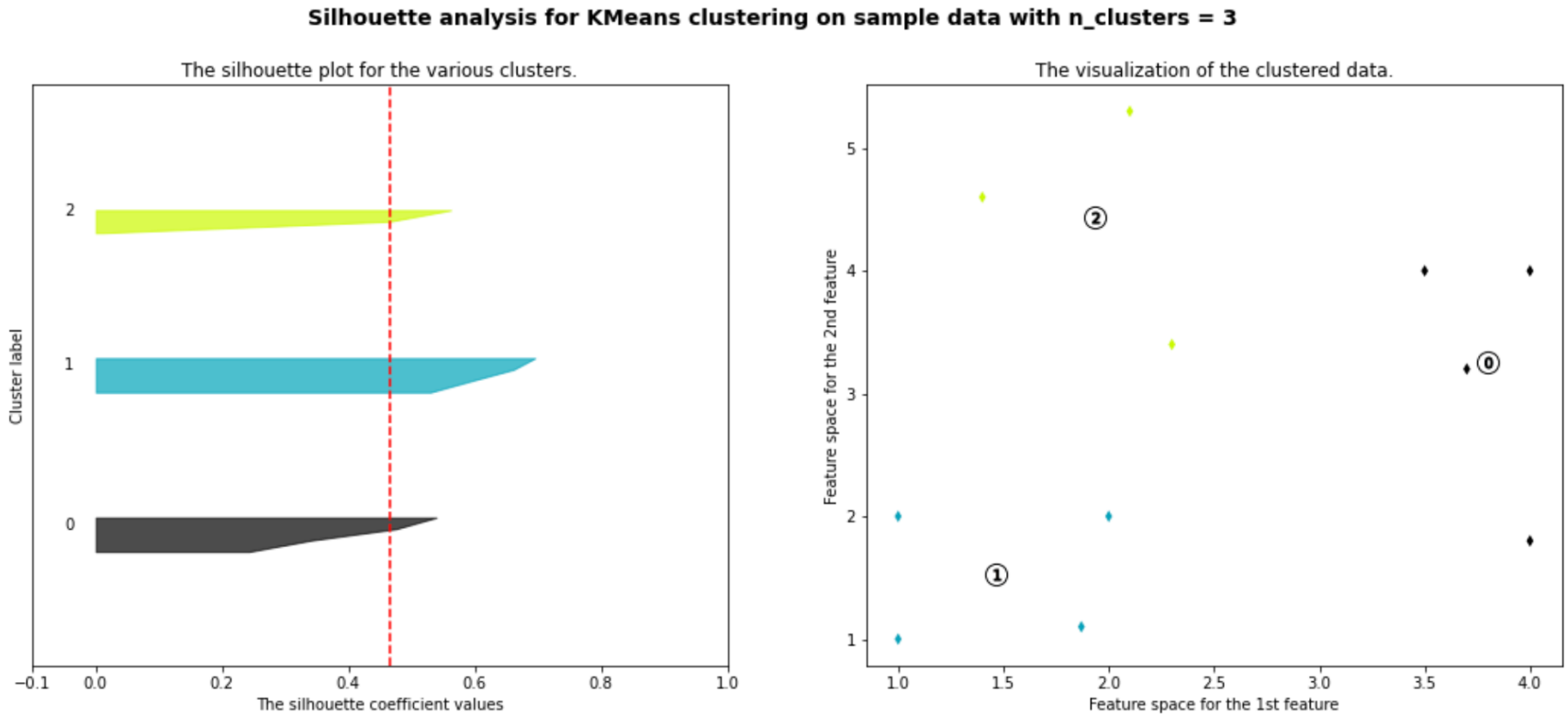
Unsupervised Learning: Clustering with K-Means

Silhouette Scores are often combined with a histogram of the scores of each point in each cluster:



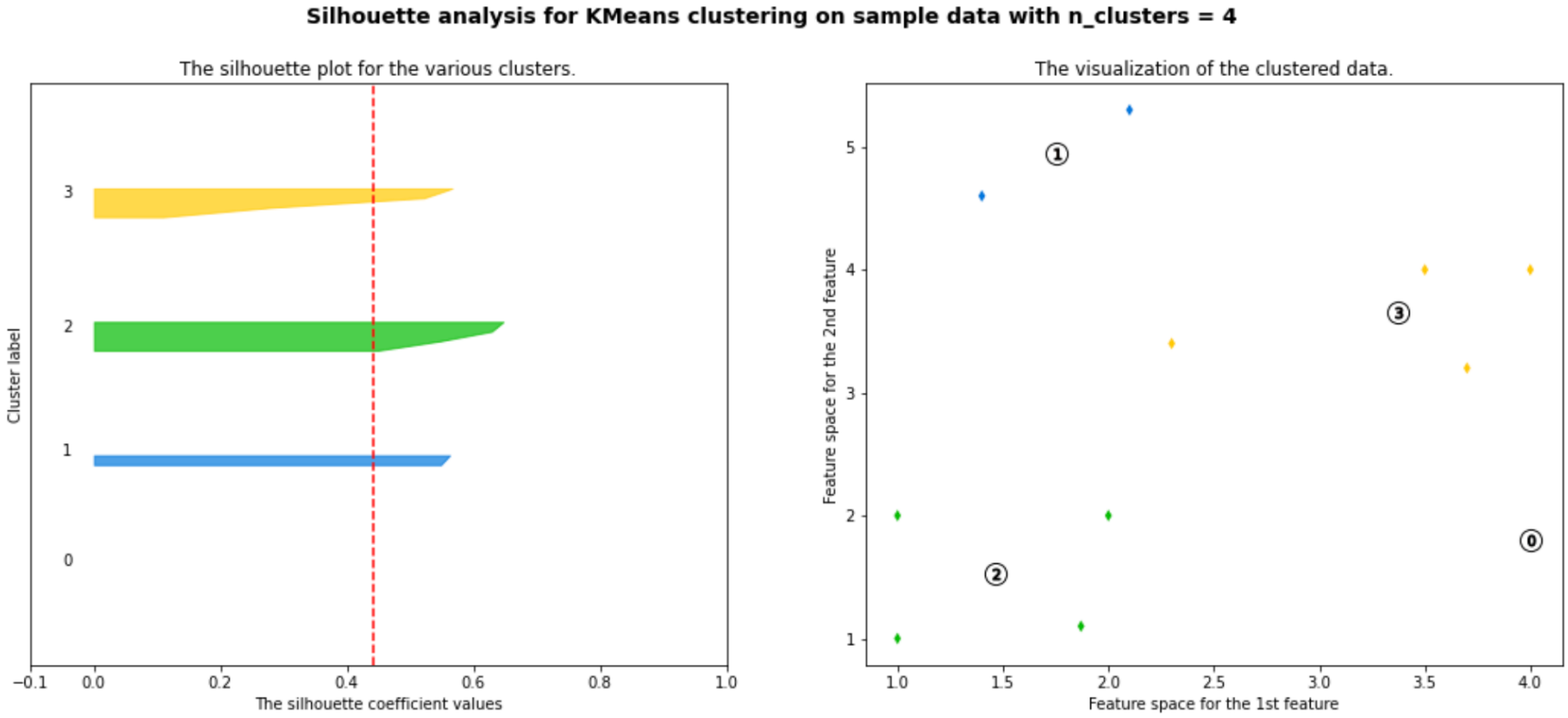
Unsupervised Learning: Clustering with K-Means

Silhouette Scores are often combined with a histogram of the scores of each point in each cluster:



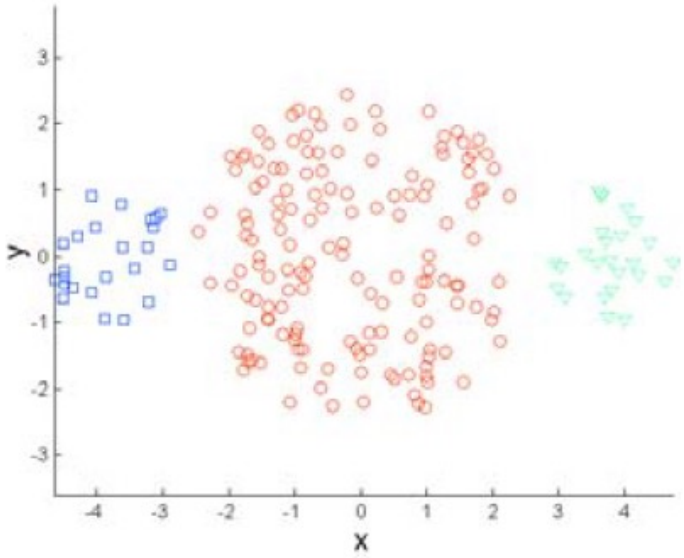
Unsupervised Learning: Clustering with K-Means

Silhouette Scores are often combined with a histogram of the scores of each point in each cluster:

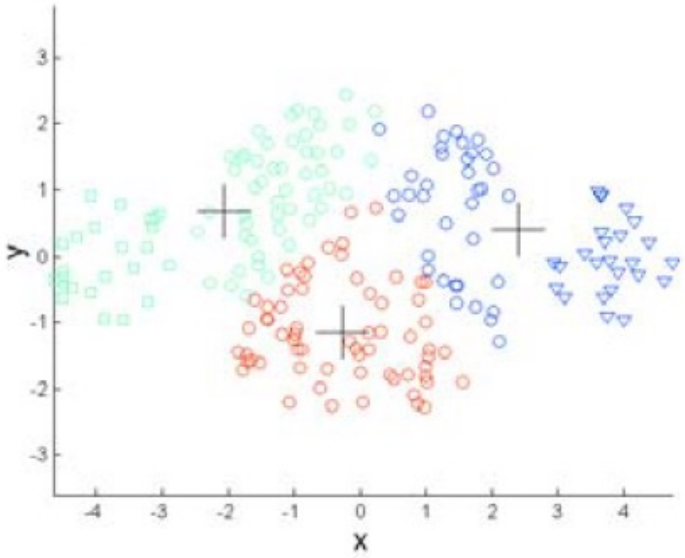


Unsupervised Learning: Clustering

Limitations of K-means: Clusters of different sizes



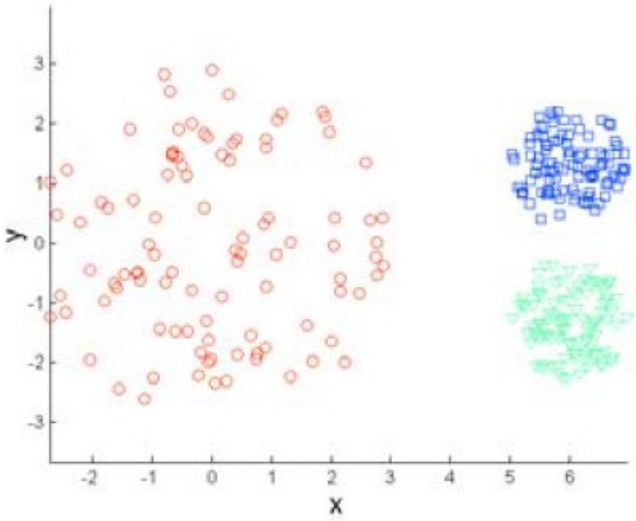
Original Points



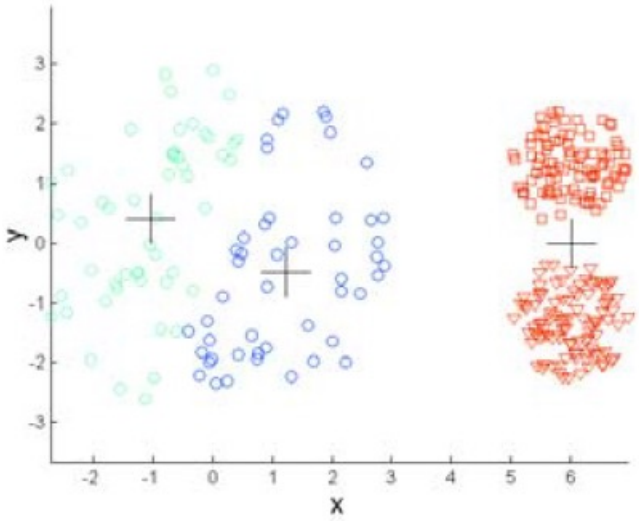
K-means (3 Clusters)

Unsupervised Learning: Clustering

Limitations of K-Means: Different Cluster Densities



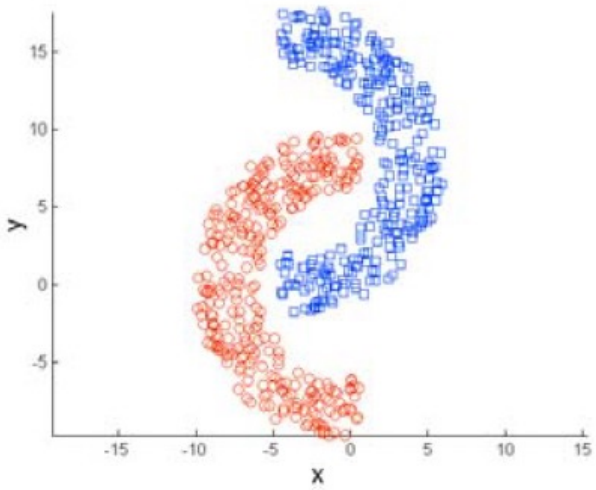
Original Points



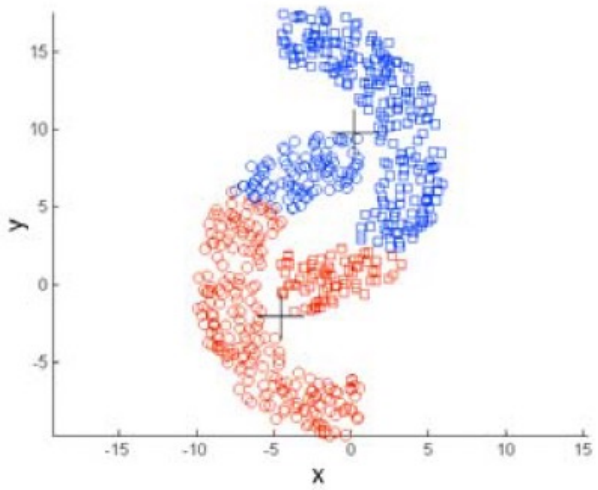
K-means (3 Clusters)

Unsupervised Learning: Clustering

Limitations of K-Means: non-spherical clusters



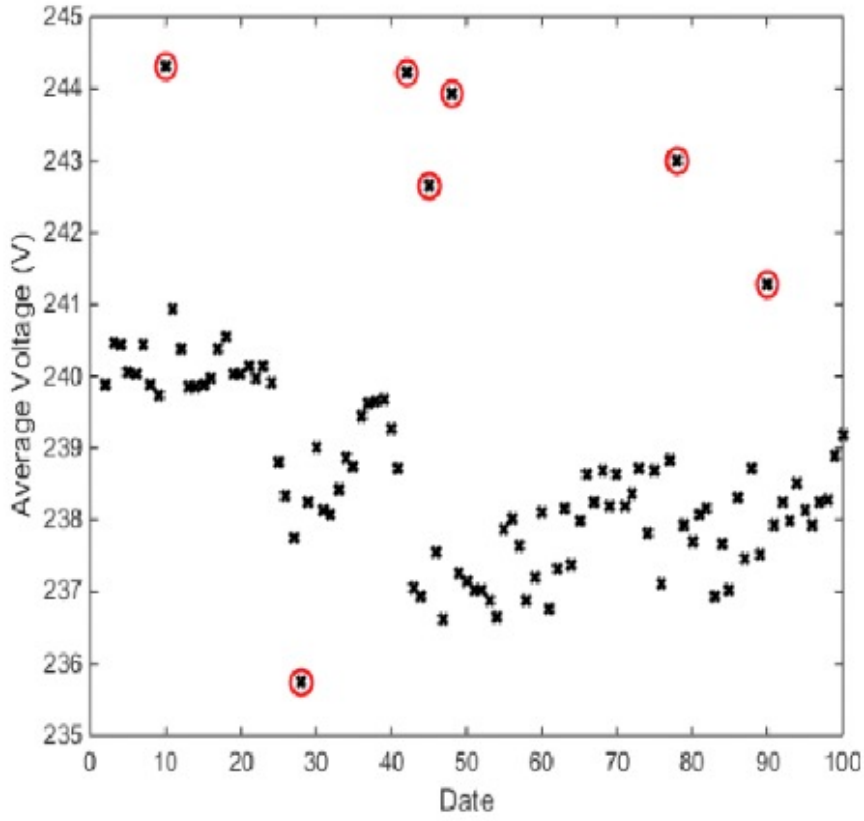
Original Points



K-means (2 Clusters)

Unsupervised Learning: Clustering

Limitations of K-Means: Outliers are a problem!



Survey of Clustering Algorithms

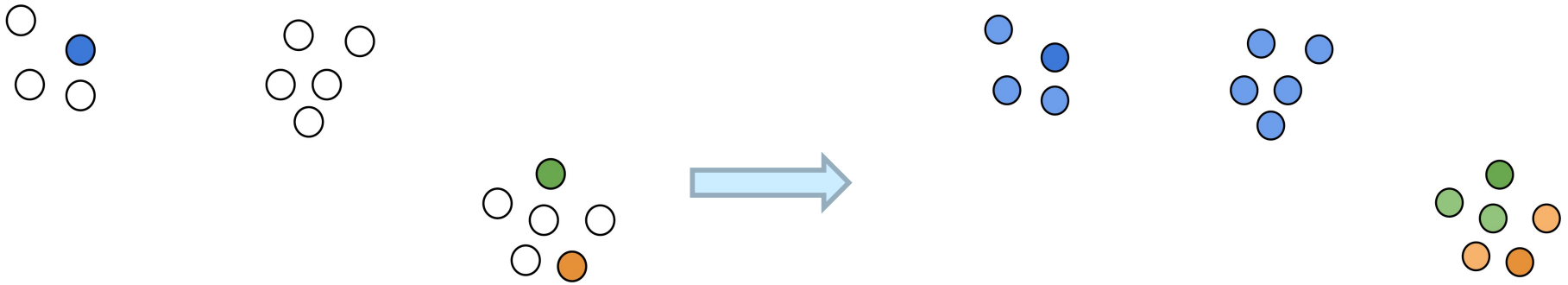
There are MANY different clustering algorithms, with different advantages and disadvantages... Here is a representative survey:

K-Means++

Improving K-Means Clustering: K-Means++

The problem with Lloyd's Algorithm is that choosing random starting points does not always lead to a good solution!

The naive solution is to run the algorithm many times with different random initializations, but the "fake it until you make it" approach is not optimal!

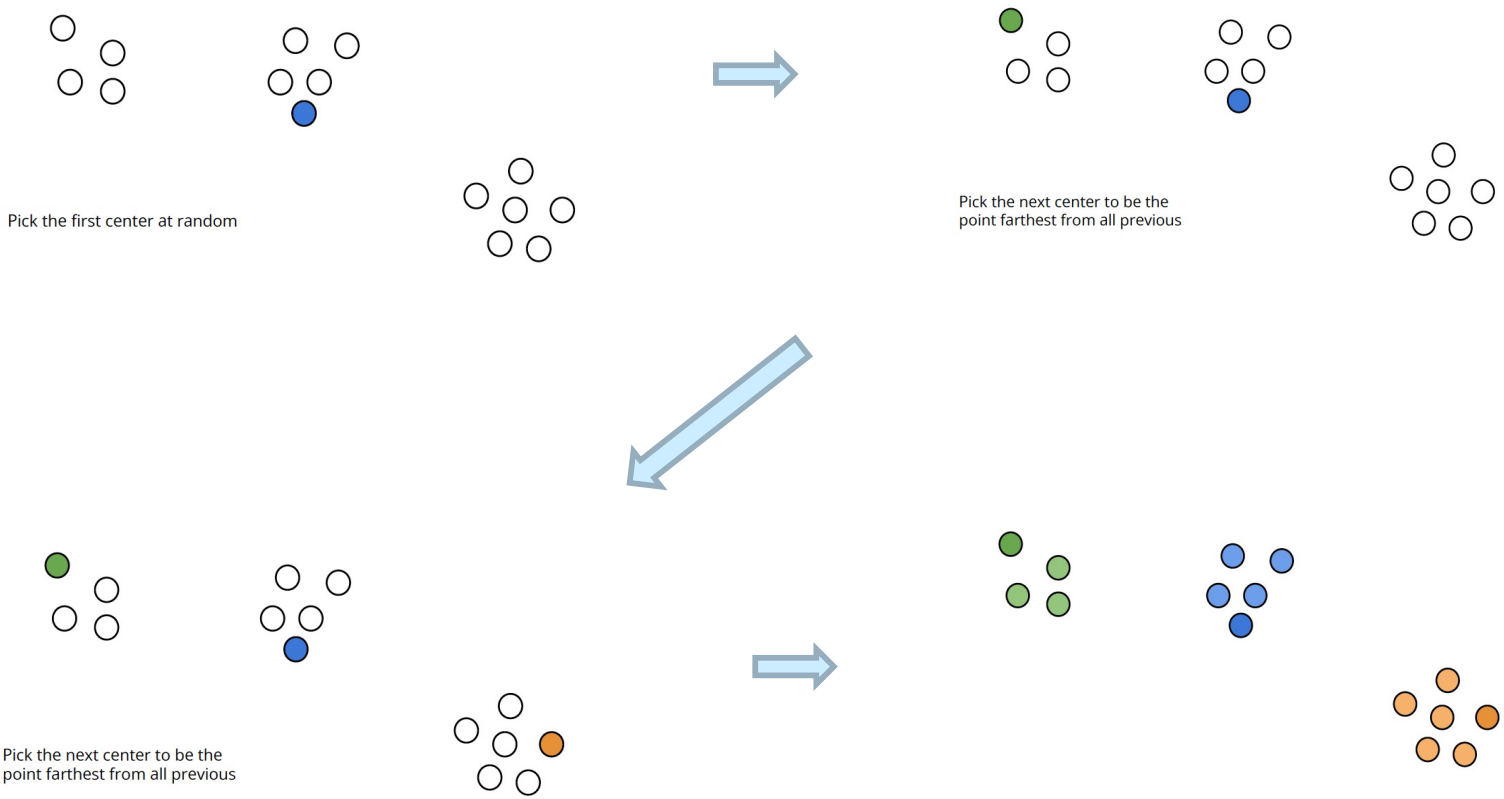


Typical problem with random initialization: Starting with initialization points too close to each other.

Improving K-Means Clustering: K-Means++

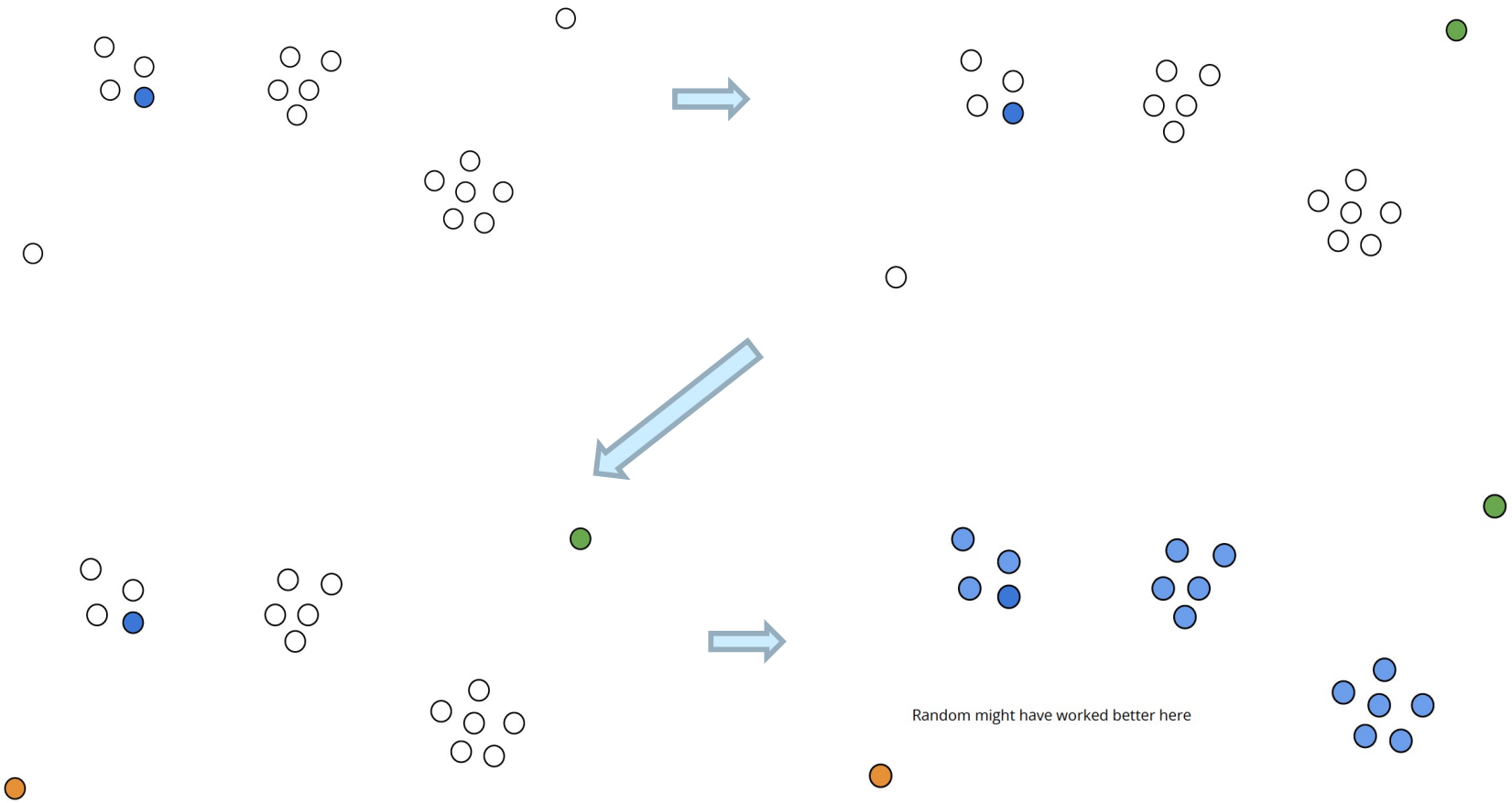
K-Means: Farthest-First Traversal (FFT) Initialization:

Pick the first center randomly; then choose the point farthest away from existing centers for the remainder.



Improving K-Means Clustering: K-Means++

K-Means: Farthest-First Traversal with Outliers



Improving K-Means Clustering: K-Means++

K-Means++ combines the two approaches (random vs FFT):

1. Start with a random center

2. Let D(x) be the distance between x and the centers selected so far.

Choose x to be the next center with probability proportional to $D(x)^a$.

where:

a = 0: Random initialization
a = 2: K-Means++
...
a = ∞ : FFT

Improving K-Means Clustering: K-Means++

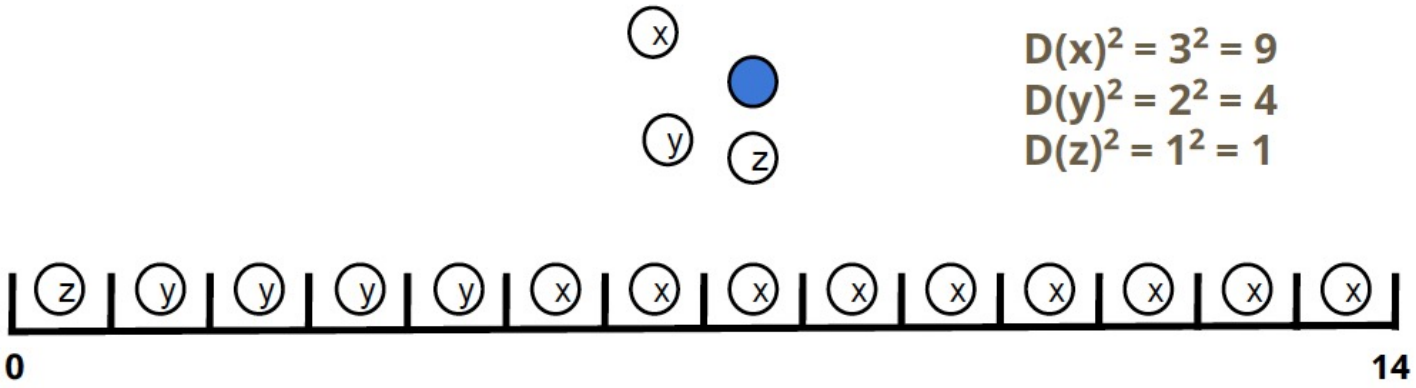
K-Means++ combines the two approaches (random vs FFT):

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

where:

a = 0: Random initialization
a = 2: K-Means++

How to select centroids using this method?

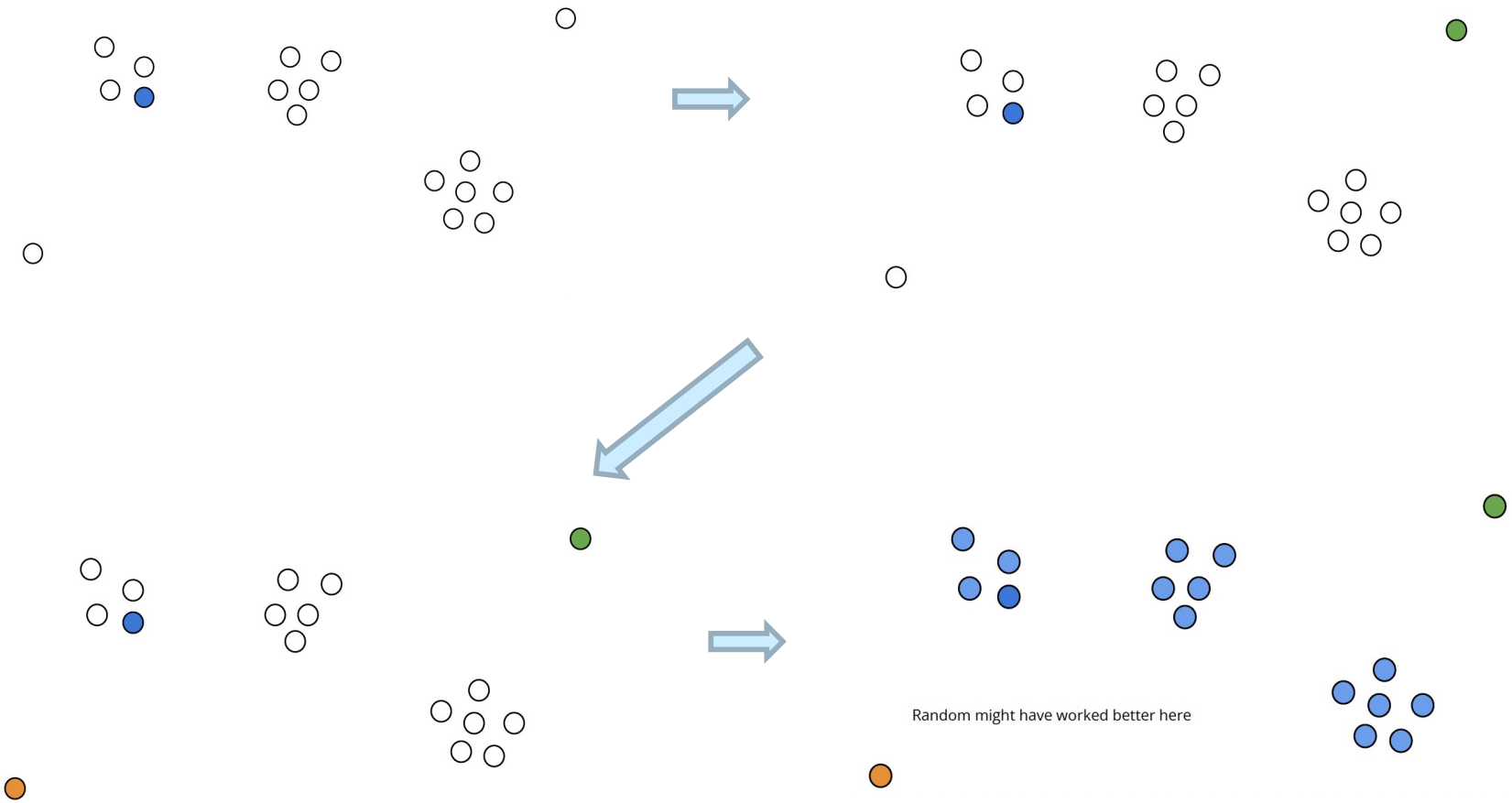



```
from numpy.random import randint

r = randint(15)
```

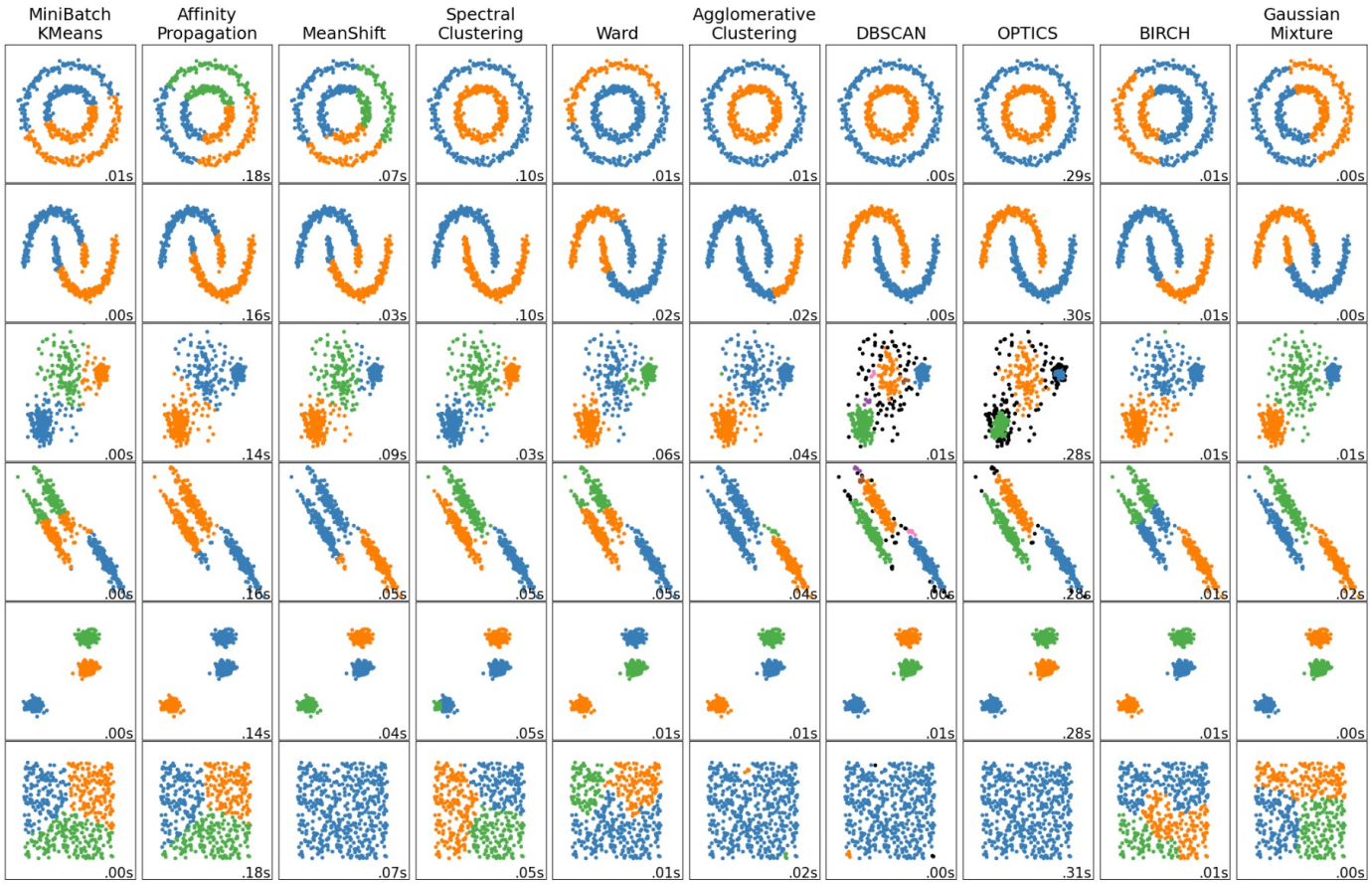
Improving K-Means Clustering: K-Means++

K-Means++



Clustering Algorithms

K-Means++ is the standard clustering algorithm, with many advantages, including speed of convergence. However, there are many others to choose from, and each has advantages and disadvantages. Choosing the appropriate method starts with exploratory data analysis.



A comparison of the clustering algorithms in scikit-learn

Hierarchical Clustering

Another framework for clustering, essentially different from K-Means and its variants, is Hierarchical Clustering, which has two flavors:

Divisive:

1. Start with all points in one cluster;
2. At each step, split until every point is in its own cluster

Agglomerative Clustering Algorithm:

1. Start with each point in its own cluster;
2. Compute the distance between all pairs of clusters;
3. Merge the two closest clusters;
4. Repeat 3 & 4 until only one cluster remains.

How do we define distance between clusters?

Each answer will give us a slightly different Hierarchical Clustering algorithm....

Hierarchical Clustering

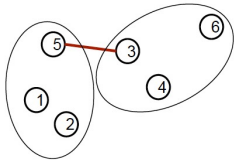
Agglomerative Clustering Algorithm:

1. Start with each point in its own cluster;
2. Compute the distance between all pairs of clusters;
3. Merge the two closest clusters;
4. Repeat 3 & 4 until only one cluster remains.

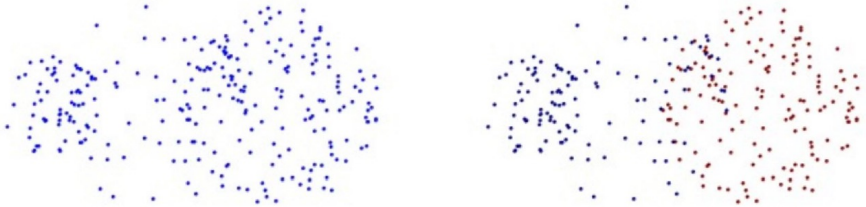
Distance calculation:

1. Single-Link Distance: Minimum distance between a point in one and a point in the other cluster:

$$D_{SL}(C_1, C_2) = \min \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

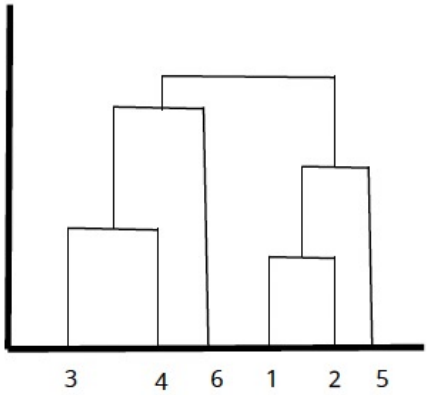


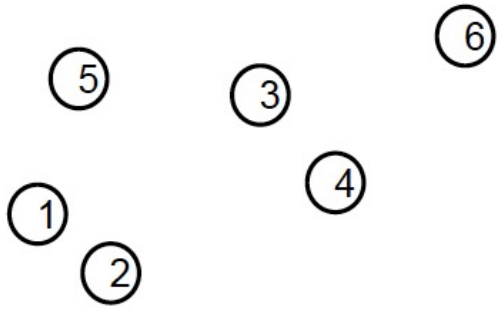
Can handle clusters of different sizes



But... Sensitive to noise points
Tends to create elongated clusters

Hierarchical Clustering





Hierarchical Clustering

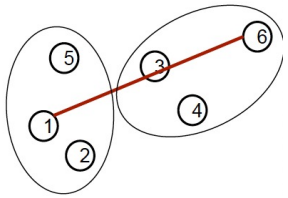
Agglomerative Clustering Algorithm:

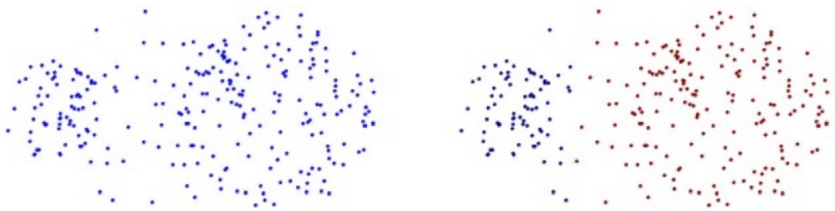
1. Start with each point in its own cluster;
2. Compute the distance between all pairs of clusters;
3. Merge the two closest clusters;
4. Repeat 3 & 4 until only one cluster remains.

Distance calculation:

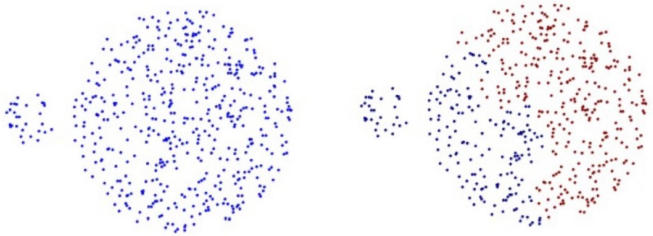
2. Complete-Link Distance: Maximum distance between a point in one and a point in the other cluster:

$$D_{CL}(C_1, C_2) = \max \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$





Less susceptible to noise
Creates more balanced (equal diameter) clusters



But... Tends to split up large clusters.
All clusters tend to have the same diameter

Hierarchical Clustering

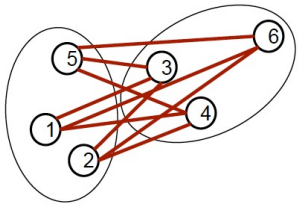
Agglomerative Clustering Algorithm:

1. Start with each point in its own cluster;
2. Compute the distance between all pairs of clusters;
3. Merge the two closest clusters;
4. Repeat 3 & 4 until only one cluster remains.

Distance calculation:

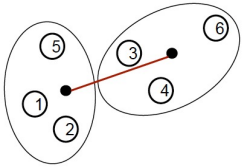
3. Average-Link Distance: Average distance between a point in one and a point in the other cluster:

$$D_{AL}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$$



Less susceptible to noise and outliers, but tends to be biased toward globular clusters....

Also: Centroid Distance, Ward's Distances, etc., etc.



Hierarchical Clustering

Agglomerative Clustering Algorithm:

1. Start with each point in its own cluster;
2. Compute the distance between all pairs of clusters;
3. Merge the two closest clusters;
4. Repeat 3 & 4 until only one cluster remains.

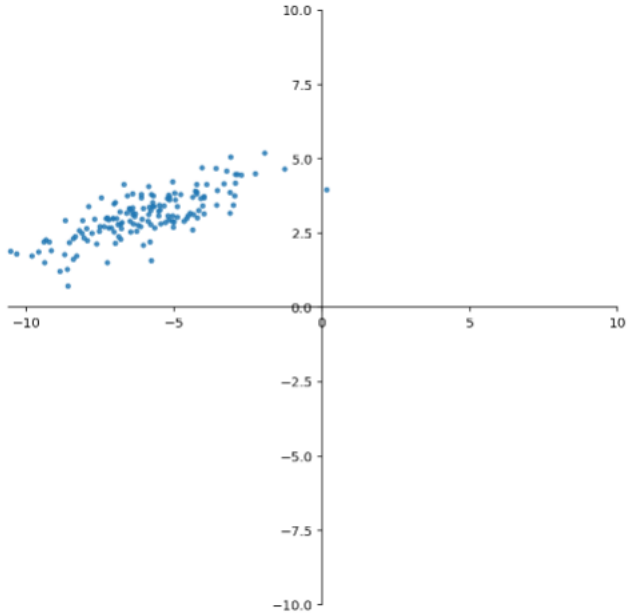
Finding the threshold with which to cut the dendrogram requires exploration and tuning. But in general hierarchical clustering is used to expose a hierarchy in the data (ex: finding/defining species via DNA similarity).

Dimensionality Reduction: PCA

A third kind of unsupervised learning is Principle Components Analysis: reducing the number of dimensions in a data set (typically down to 2 dimensions) whilst preserving the most important characteristics of the data.

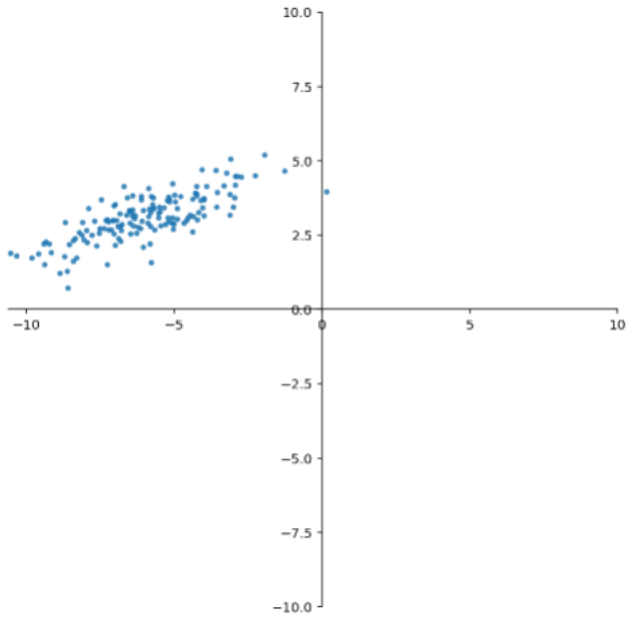
By "most important," we mean those parts of the data which show the most differences (e.g., variance).

Let's consider a small example to see the basic algorithm. Consider the points below, which live in R^2:



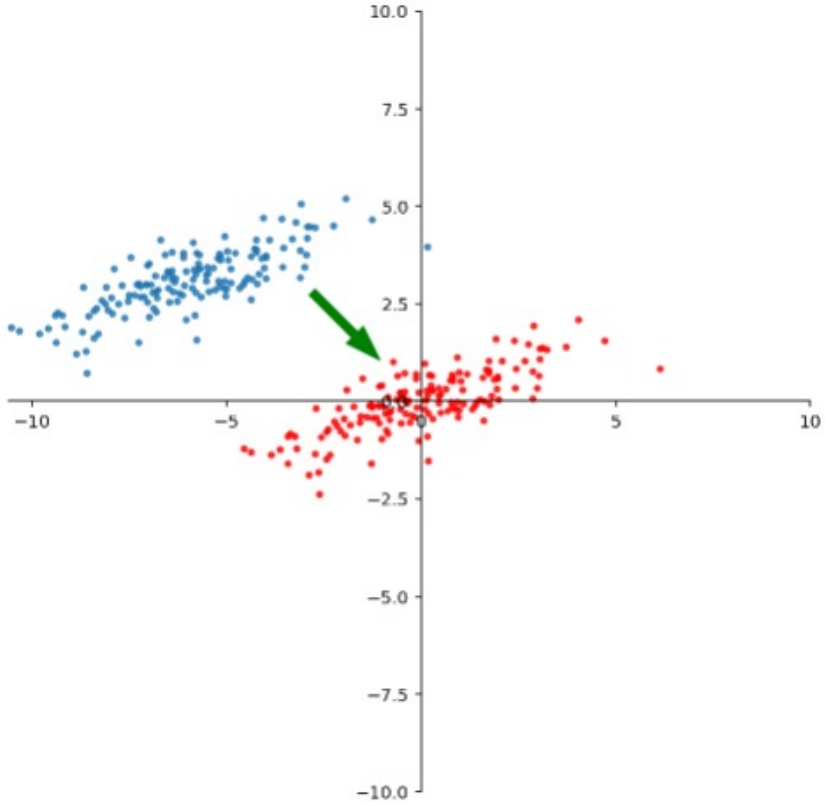
Dimensionality Reduction: PCA

These points are in two dimensions, but most of the variation (most of the meaning) in the data is in 1 dimension, i.e., along a line (not exactly a dimension):



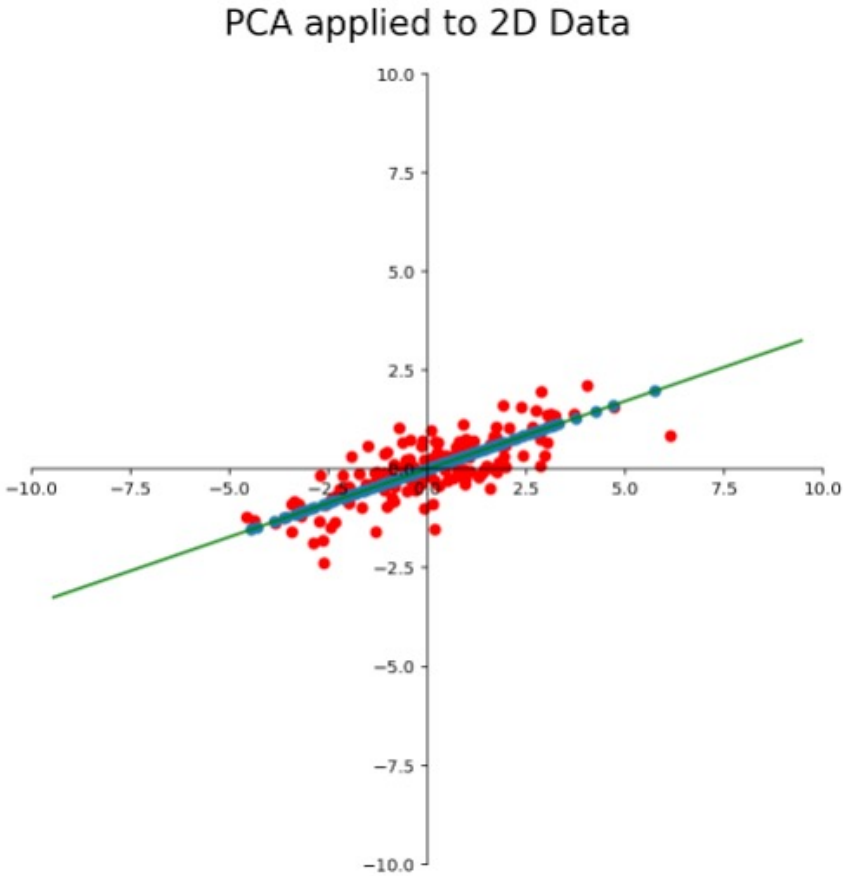
Dimensionality Reduction: PCA

The first step in PCA is to shift the data so its centroid (mean value point) is at the origin:



Dimensionality Reduction: PCA

Next, we analyze the data using eigenvectors (formally, Singular-Value Decomposition) to find the line through the data in the direction with the highest variance, and then project the data onto that line (all linear algebra stuff!):



Dimensionality Reduction: PCA

This can be done any time we want to visualize a complex set of data, say in 2 dimensions. Here is an example:

```
from sklearn.datasets import fetch_20newsgroups
categories = ['comp.os.ms-windows.misc', 'sci.space', 'rec.sport.baseball']
news_data = fetch_20newsgroups(subset='train', categories=categories)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english', min_df=4, max_df=0.8)
dtm = vectorizer.fit_transform(news_data.data).todense()
```

```
print('The size of our document-term matrix is {}'.format(dtm.shape))
```

```
The size of our document-term matrix is (1781, 9409)
```

This data set has 9409 dimensions! Here is the PCA reduction to two dimensions, with the three clusters shown: