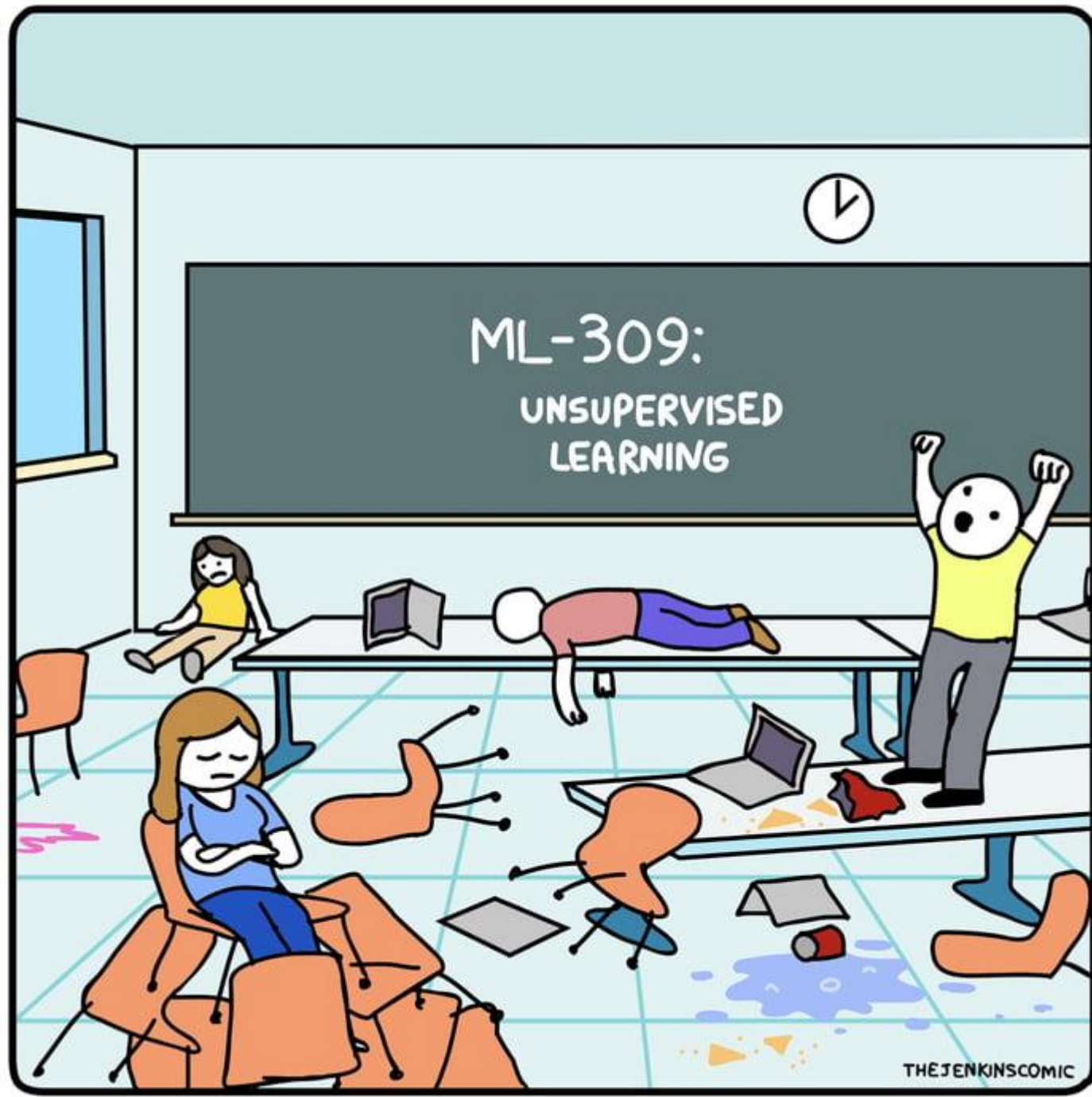


Unsupervised Learning

Vishal Patel

Spring 2024



SUPERVISED LEARNING

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1j} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2j} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nj} \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}$$

The goal is to fit **a model that relates the target to the features**, with the aim of accurately predicting the response for future observations.

UNSUPERVISED LEARNING

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1j} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2j} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nj} \end{pmatrix}$$

The goal is to understand the **relationships between the observations** or **between the features**.

Clustering

Partition observations[†] into groups
based on similarity.

E.g., Customer Segmentation.

[†]or sometimes, columns

A Few More Examples

1. A **cancer researcher** might assay gene expression levels in 100 patients with breast cancer. The researcher might then **look for subgroups** among the breast cancer samples, or among the genes, in order to obtain a better understanding of the disease.
2. An **online shopping site** might try to **identify groups of shoppers** with similar browsing and purchase histories, as well as items that are of particular interest to the shoppers within each group. Then an individual shopper can be preferentially shown the items in which the shopper is particularly likely to be interested, based on the purchase histories of similar shoppers.
3. A **search engine** might choose what search results to display to a particular individual based on the click histories of **other individuals with similar search patterns**.

Clustering Methods

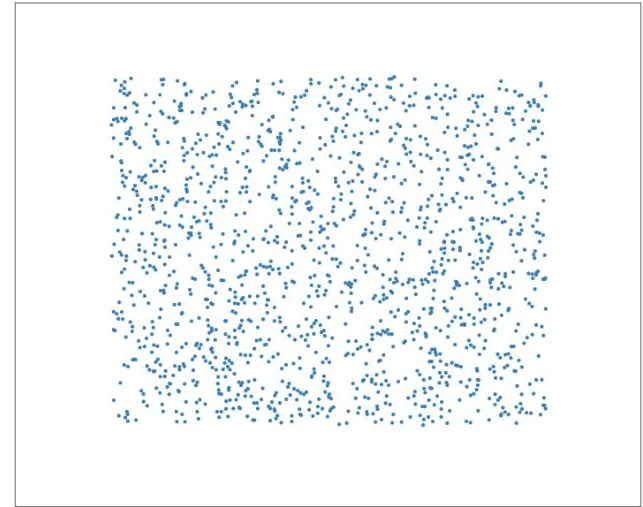
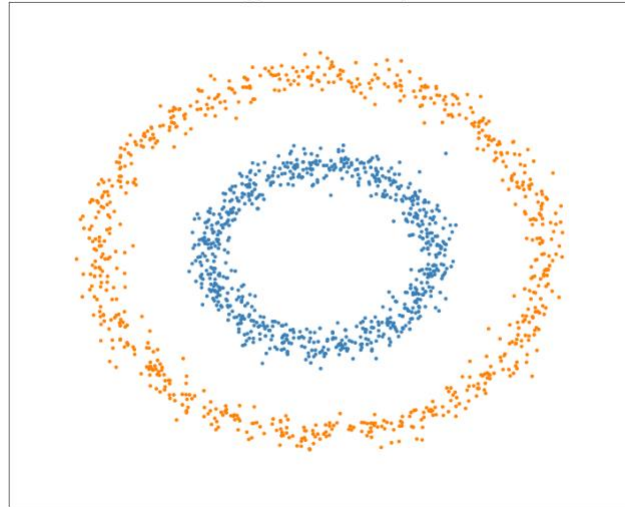
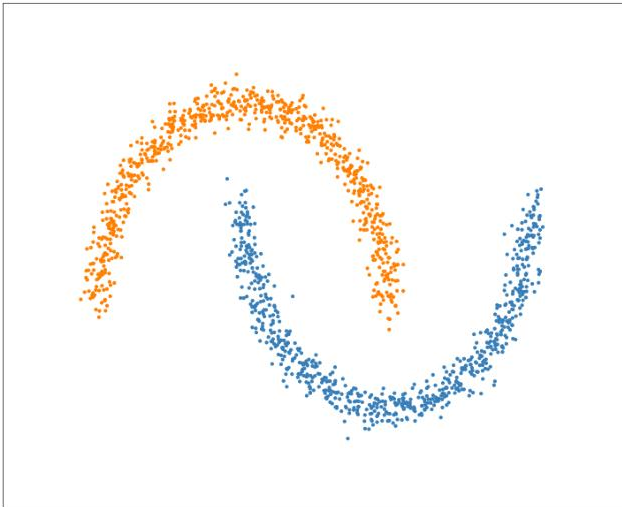
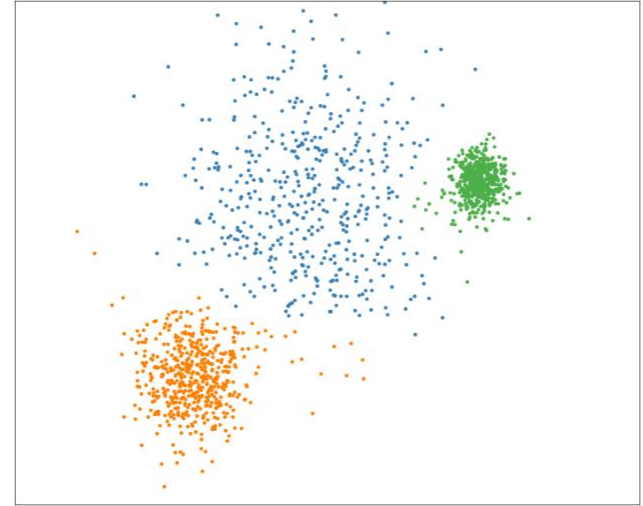
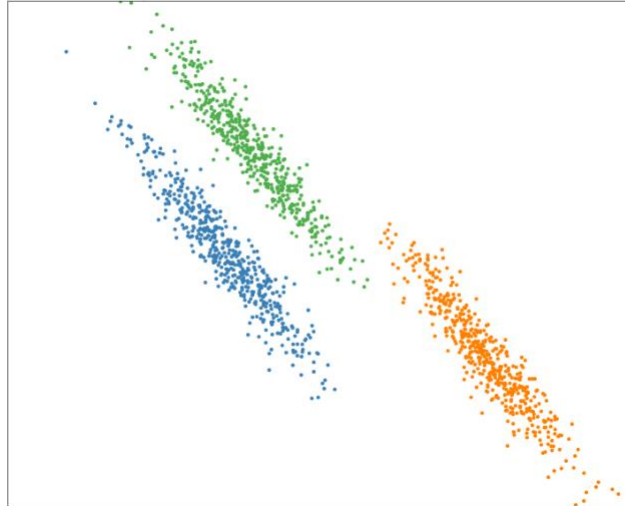
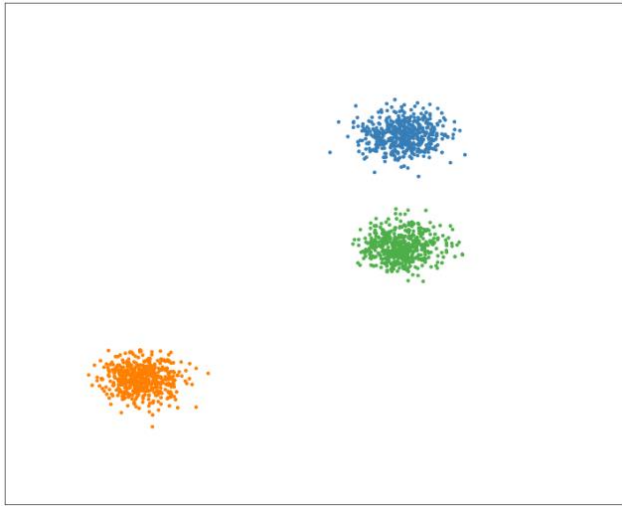
1

Distance-Based

2

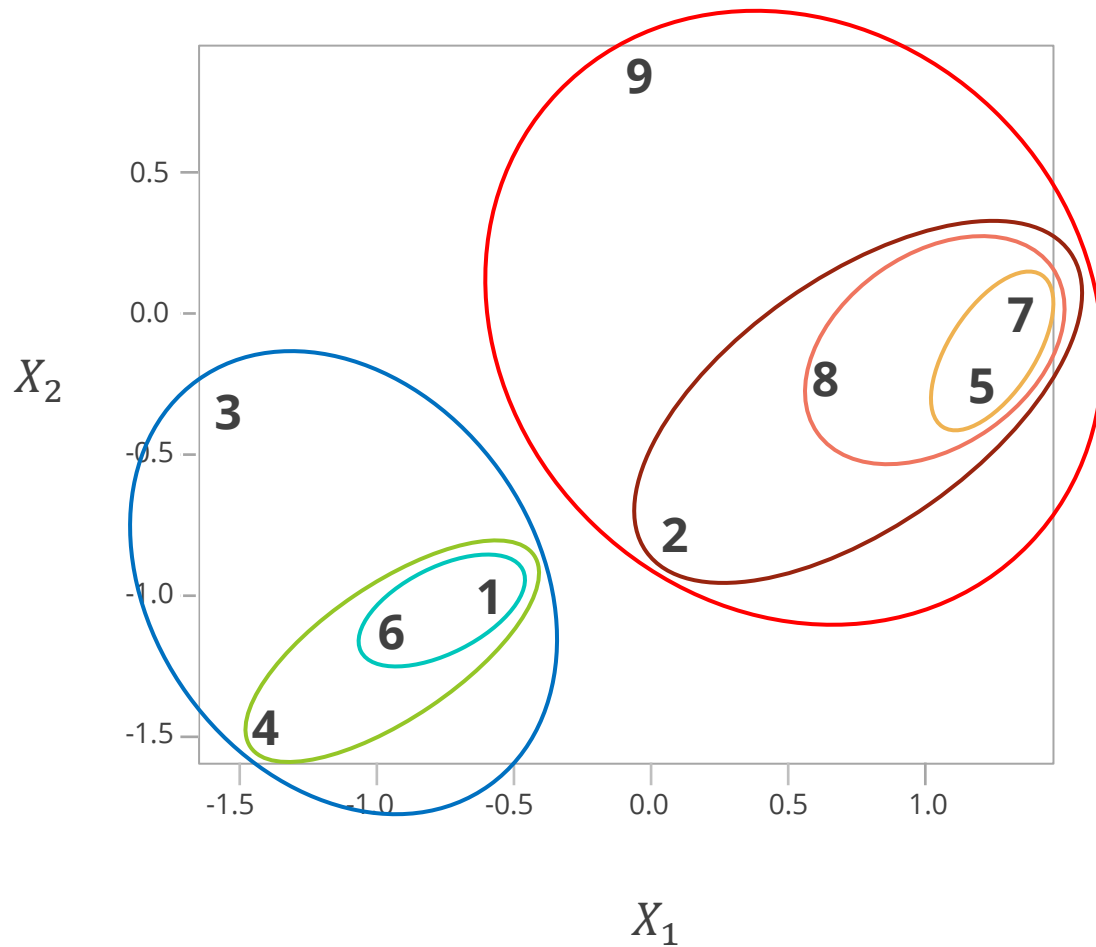
Density-Based

The Geometry of Clusters

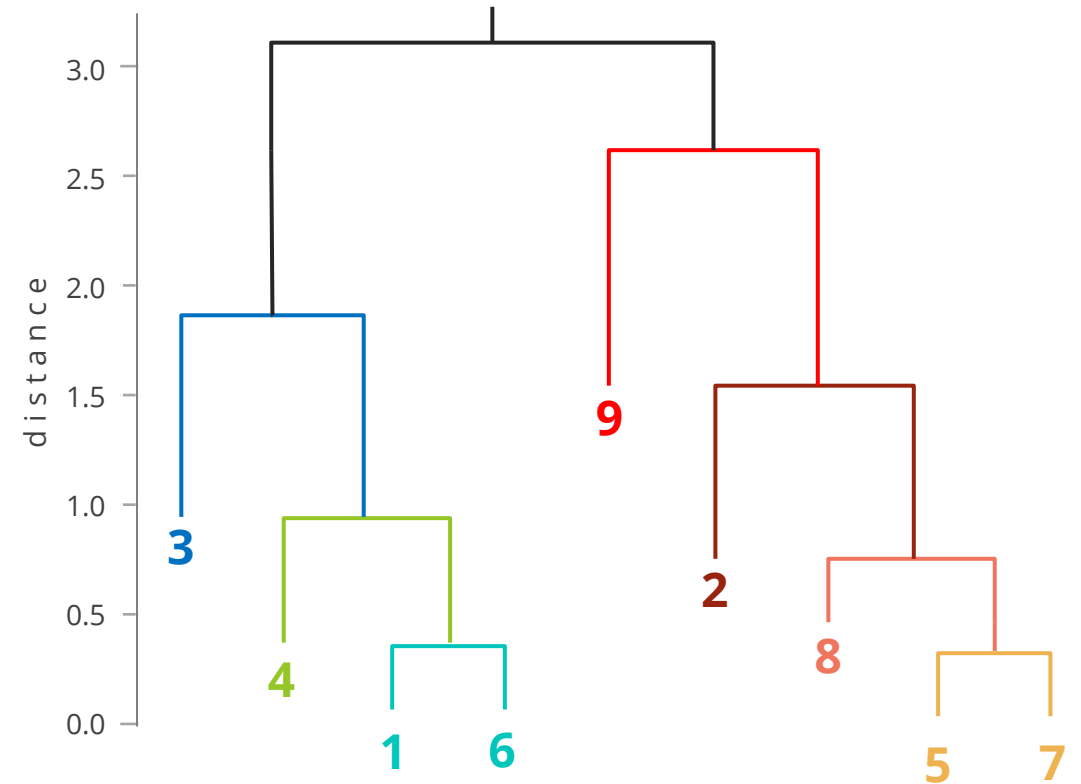


Hierarchical Clustering

Hierarchical (Agglomerative) Clustering



Nested Clusters

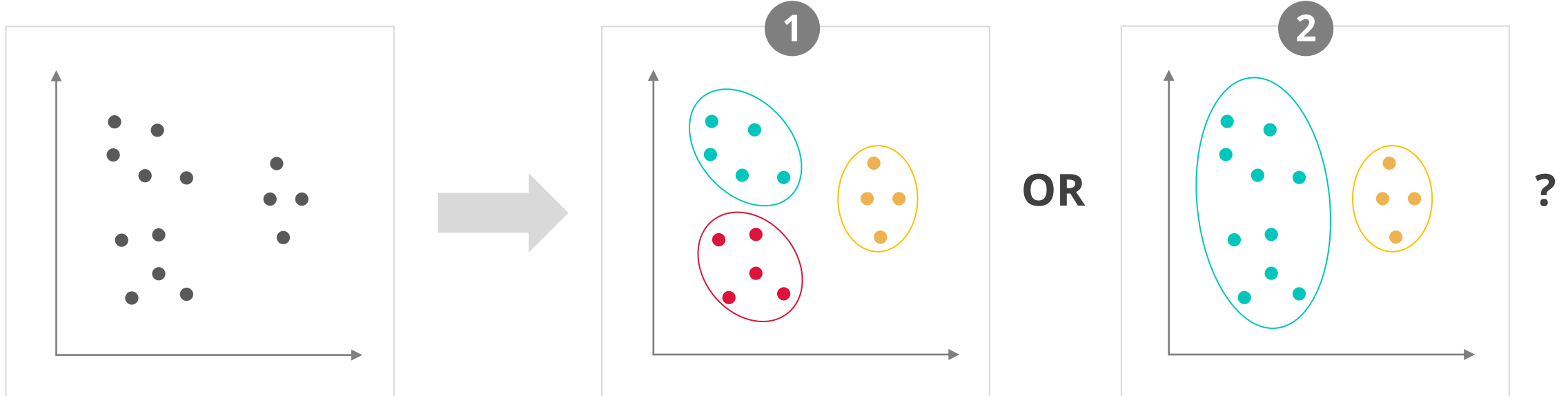


Dendrogram

ILLUSTRATIVE

Distance-Based Clustering

How to define **similarity**
between data points (and clusters)?



Similarity Measures

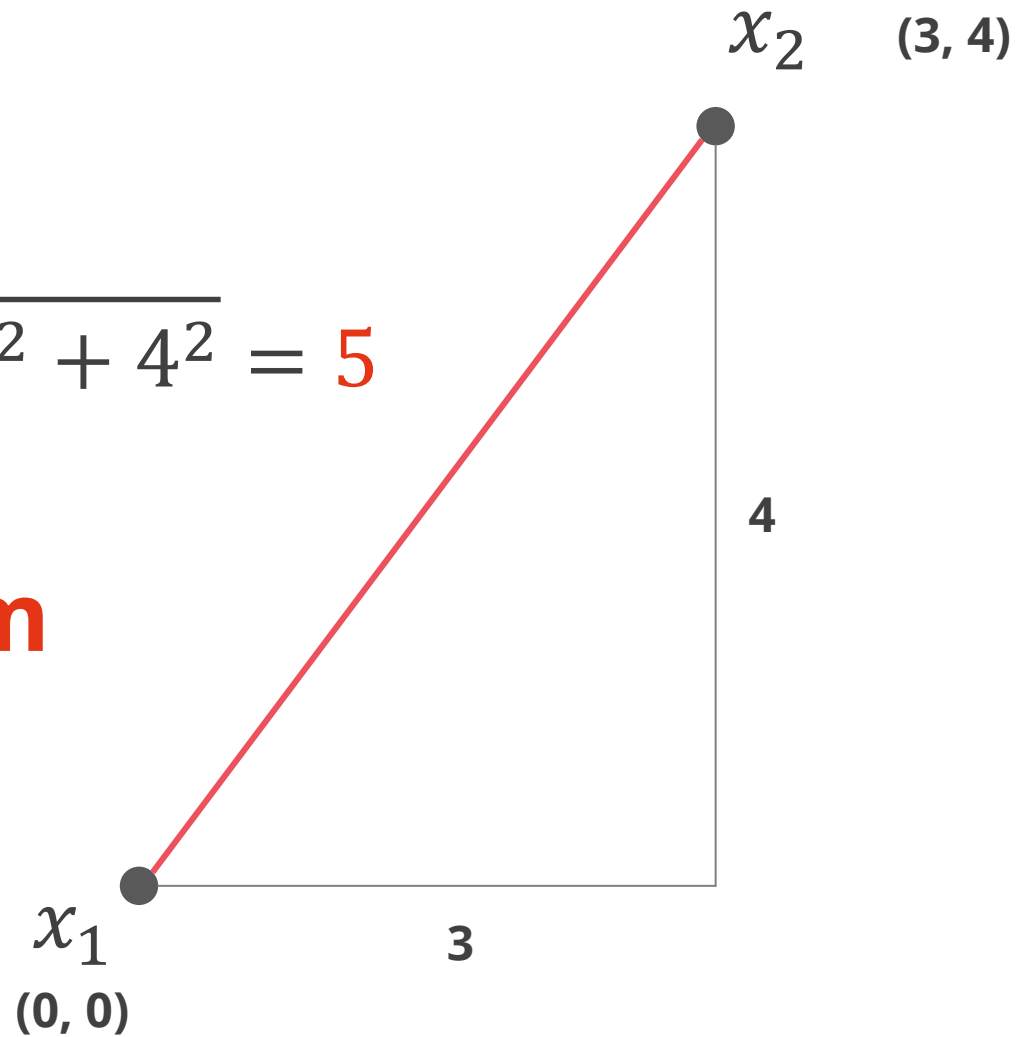
- 1 Similarity between **data points**.
- 2 Similarity between **clusters**.

Similarity Measures

- 1 Similarity between **data points**.
- 2 Similarity between **clusters**.

$$\text{dist}(x_1, x_2) = \sqrt{3^2 + 4^2} = 5$$

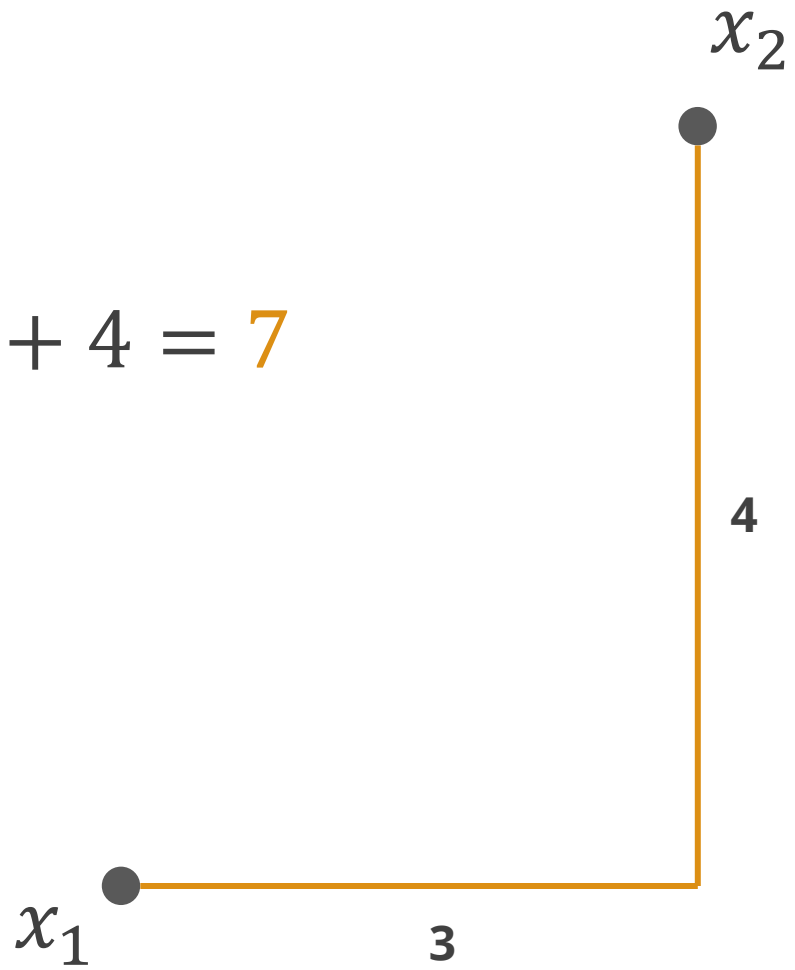
L_2 -norm



This is the most common notion of “distance”.

$$\text{dist}(x_1, x_2) = 3 + 4 = 7$$

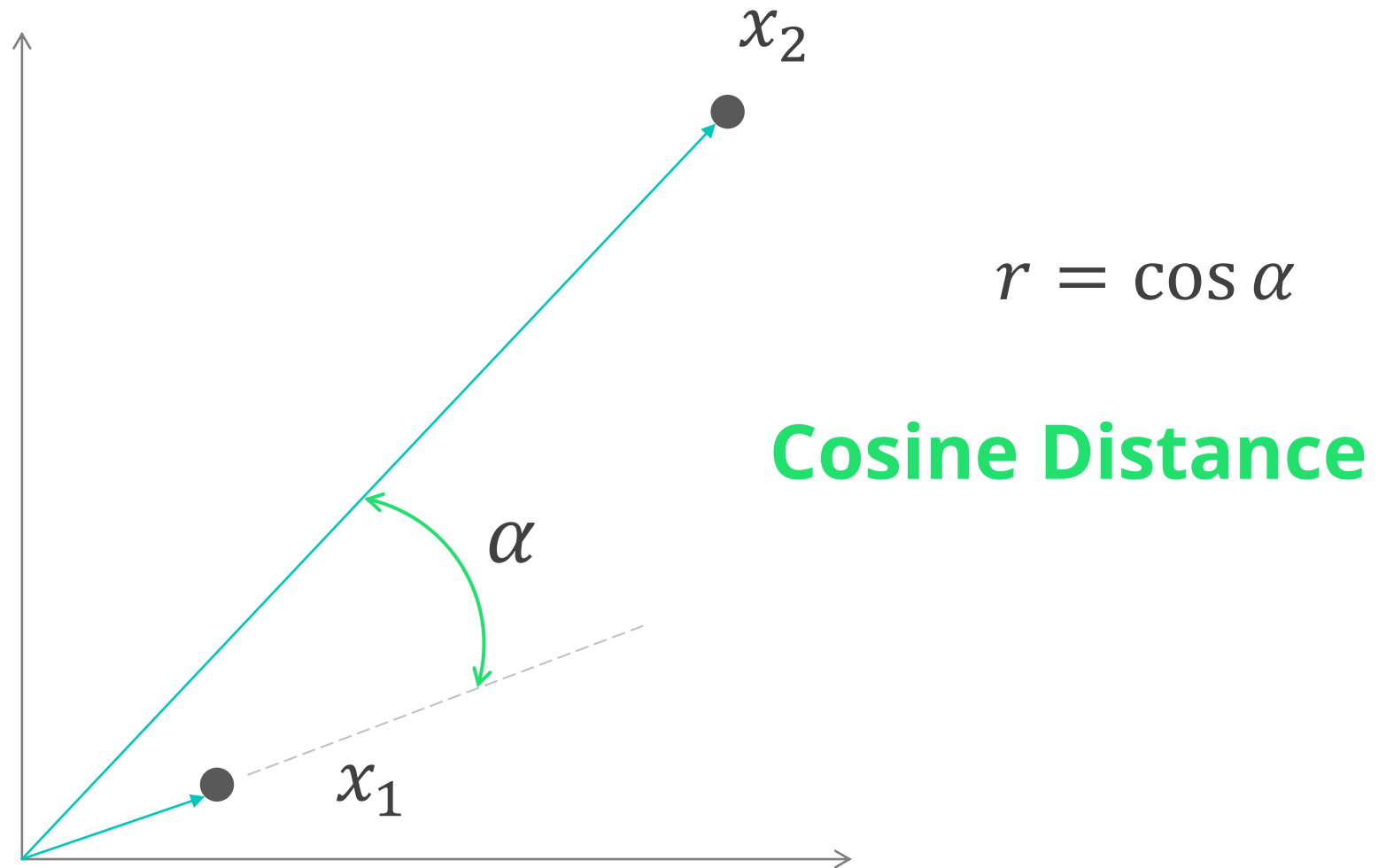
L_1 -norm



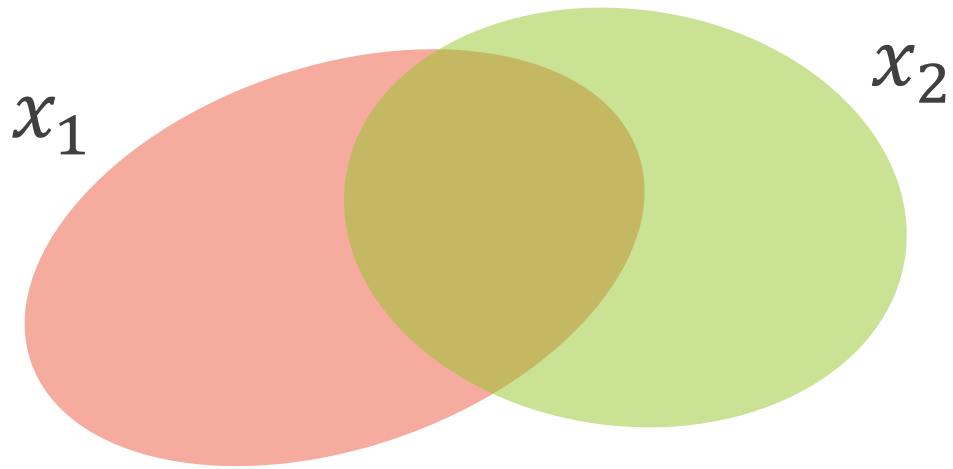
L_1 -norm



L_1 -norm is also known as the **Manhattan distance** – the distance if you had to travel along the coordinates only.



The correlation coefficient can be rescaled to distance measure of range 0 – 1 by $r_{distance} = (1 - r)/2$

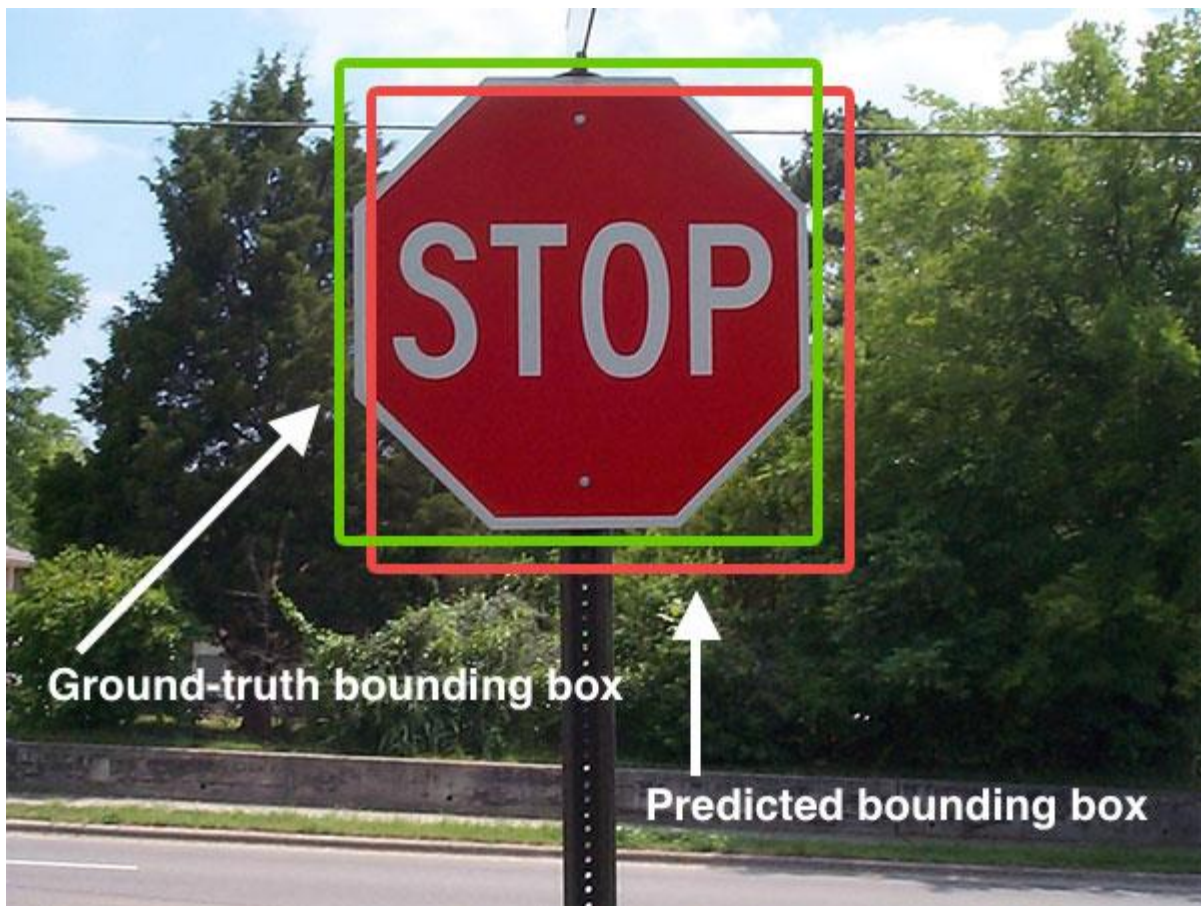


$$\text{Jaccard Similarity} = \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|}$$


$$\text{Jaccard Distance} = 1 - \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|}$$

Jaccard distance (dissimilarity) is the proportion of the combined abundance that is not shared.[†]

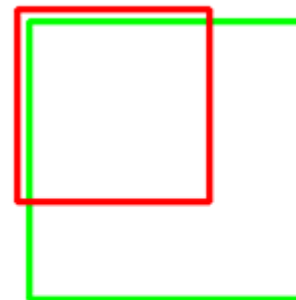
[†]This metric was originally conceived by Ecologists.



https://en.wikipedia.org/wiki/Jaccard_index

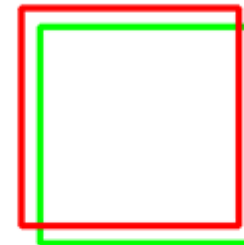
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


IoU: 0.4034



Poor

IoU: 0.7330

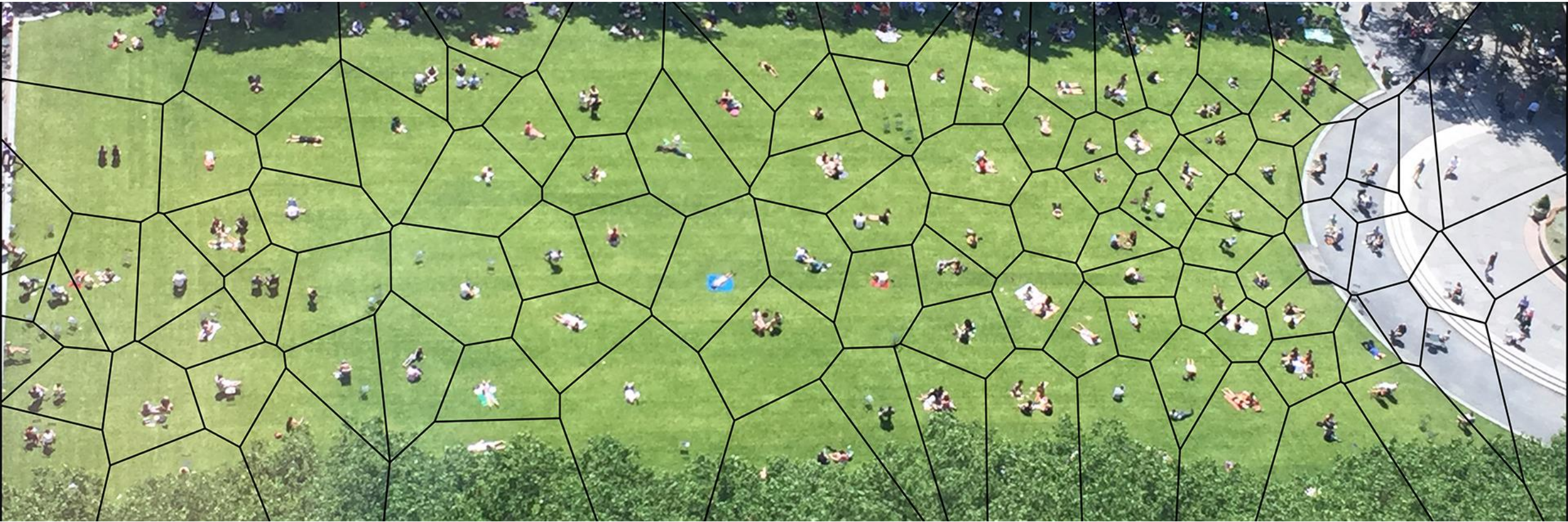


Good

IoU: 0.9264

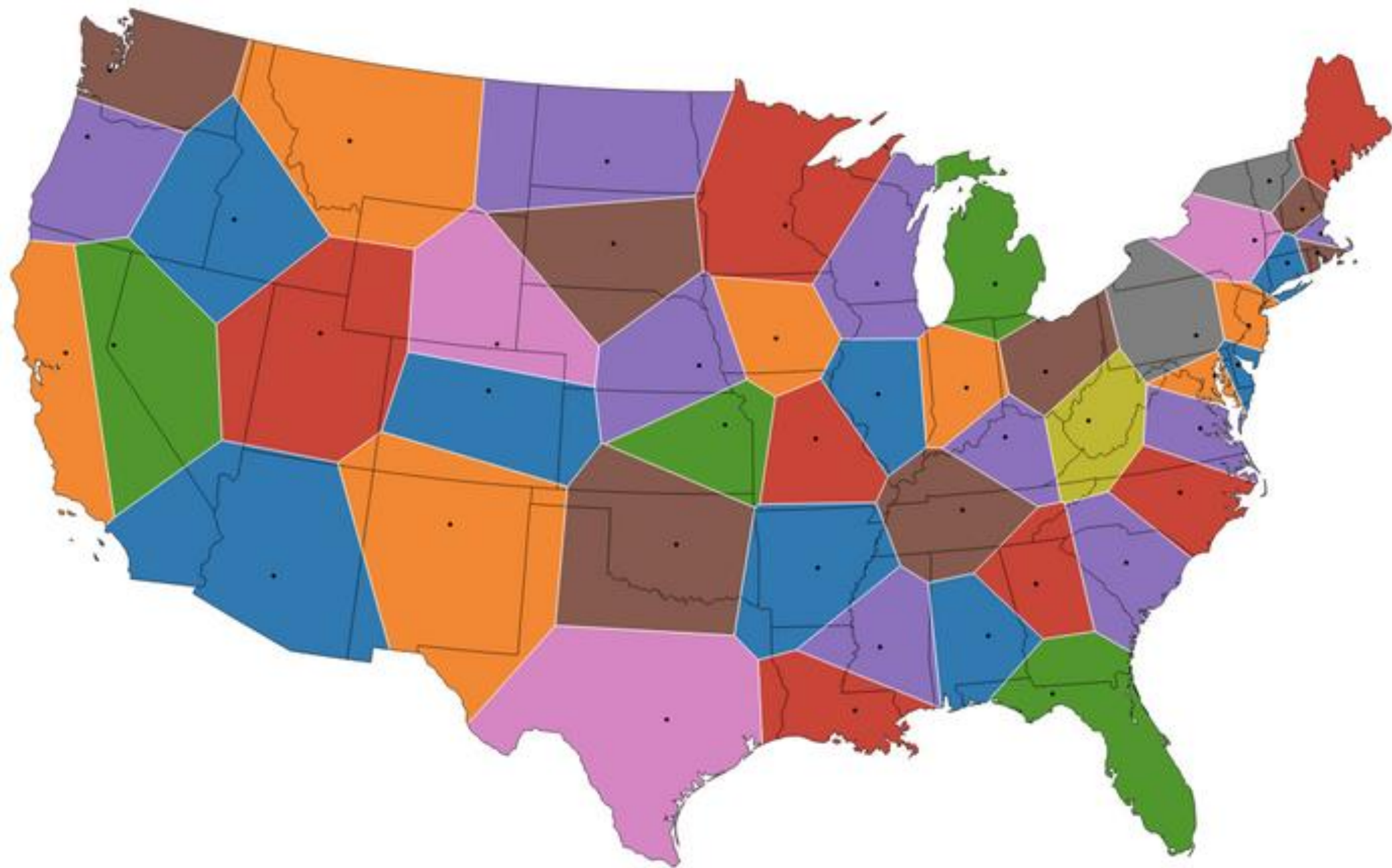


Excellent



A **Voronoi diagram** of people enjoying the sun in Bryant Park.

(by @RodBogart)



Similarity Measures

- 1 Similarity between **data points**.
- 2 Similarity between **clusters**.

Hierarchical Clustering

Distance Between Clusters:

1. **Single link** = Smallest dissimilarity
2. **Complete link** = Largest dissimilarity
3. **Average link** = Average dissimilarity
4. **Centroid method** = Distance between centroids
5. **Ward's method** = Sum of squared distances of points in clusters

Distance Between Clusters

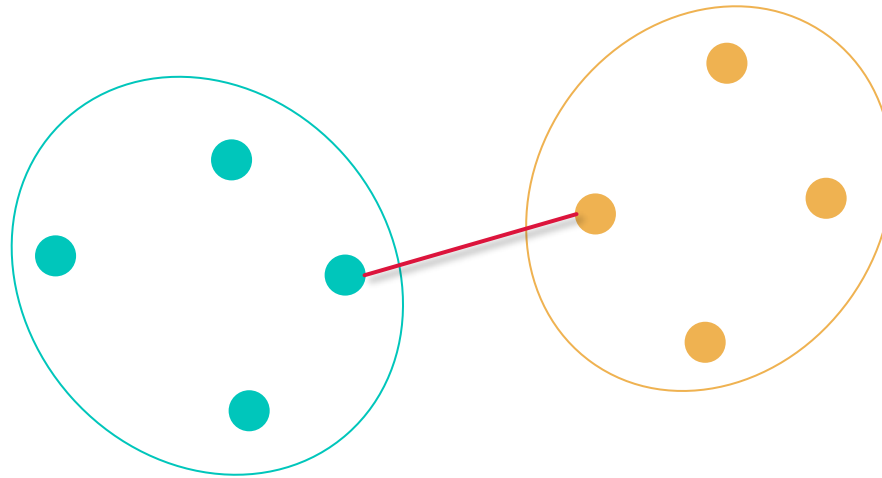
1. Single link

2. Complete link

3. Average link

4. Centroid method

5. Ward's method

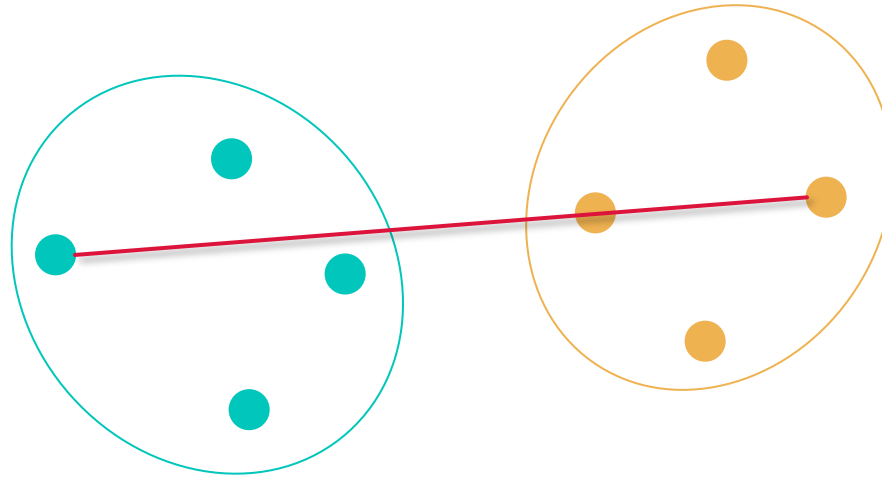


Minimal intercluster dissimilarity: Compute all pairwise dissimilarities between the observations in cluster **A** and the observations in cluster **B** and record the **smallest** of these dissimilarities.[†]

[†] From 'An Introduction to Statistical Learning' by James, Witten, Hastie, Tibshirani

Distance Between Clusters

1. Single link
- 2. Complete link**
3. Average link
4. Centroid method
5. Ward's method

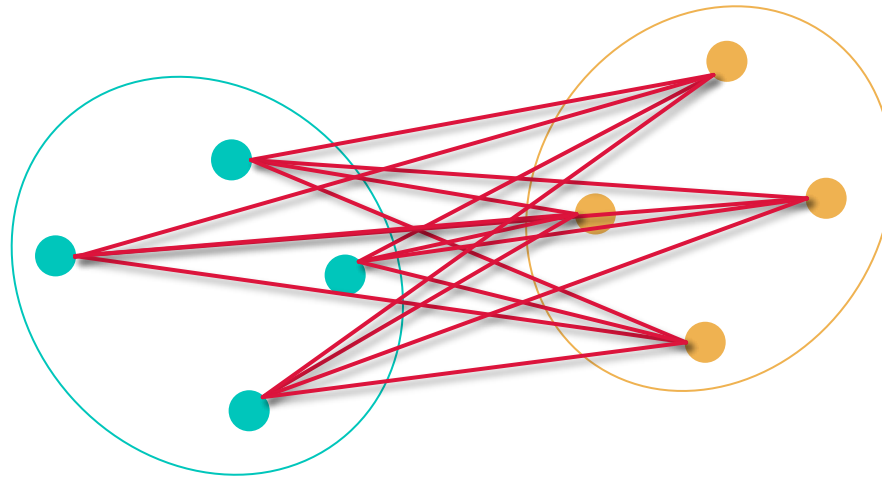


Maximal intercluster dissimilarity: Compute all pairwise dissimilarities between the observations in cluster **A** and the observations in cluster **B** and record the **largest** of these dissimilarities.[†]

[†] From 'An Introduction to Statistical Learning' by James, Witten, Hastie, Tibshirani

Distance Between Clusters

1. Single link
2. Complete link
- 3. Average link**
4. Centroid method
5. Ward's method

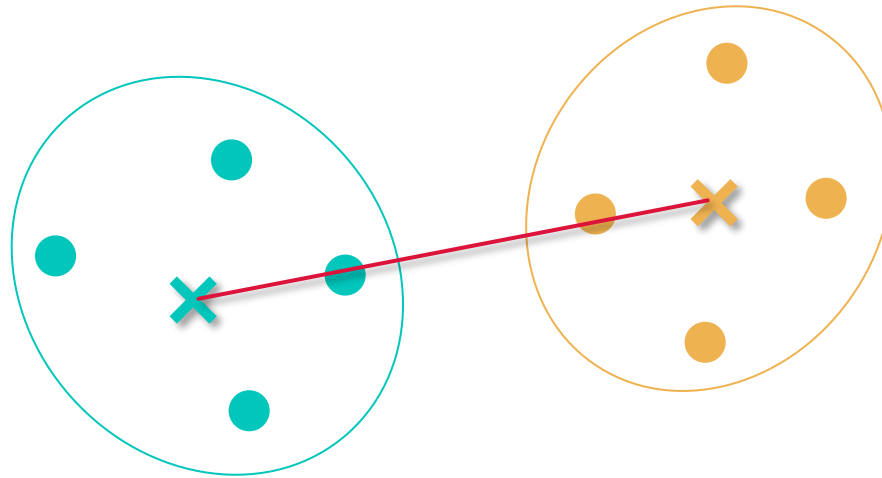


Mean intercluster dissimilarity: Compute all pairwise dissimilarities between the observations in cluster **A** and the observations in cluster **B** and record the **average** of these dissimilarities.[†]

[†] From 'An Introduction to Statistical Learning' by James, Witten, Hastie, Tibshirani

Distance Between Clusters

1. Single link
2. Complete link
3. Average link
- 4. Centroid method**
5. Ward's method

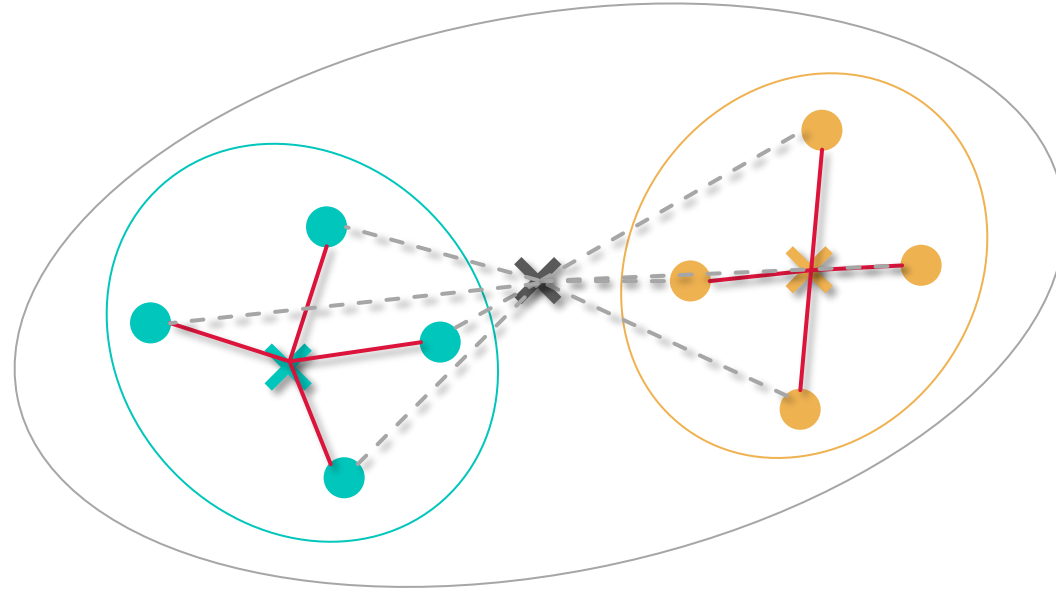


Dissimilarity between the centroid for cluster **A** and the centroid for cluster **B**.[†]

[†] From 'An Introduction to Statistical Learning' by James, Witten, Hastie, Tibshirani

Distance Between Clusters

1. Single link
2. Complete link
3. Average link
4. Centroid method
5. **Ward's method**



Minimum variance: The dissimilarity between two clusters, **A** and **B**, is how much **the sum of squares** will increase when we merge them.

$$\Delta(A, B) = \sum_{i \in A \cup B} \|x_i - \mu_{A \cup B}\|^2 - \sum_{i \in A} \|x_i - \mu_A\|^2 - \sum_{i \in B} \|x_i - \mu_B\|^2$$

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

?

{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}
-------	------	-------	-------	------	-------

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

?

{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

?

{BOS, NY, CHI}		{DEN}	{SF}	{SEA}	
{BOS, NY}	{CHI}	{DEN}	{SF}	{SEA}	
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

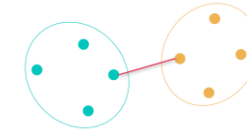
?

{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

Single Linkage



{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

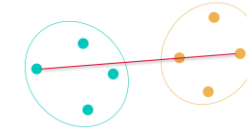
Single Linkage

{BOS, NY, CHI, DEN}				{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

Complete Linkage



{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Example of Hierarchical Clustering

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1,949	3,095	2,979
NY		0	802	1,771	2,934	2,815
CHI			0	966	1,242	2,103
DEN				0	1,235	1,307
SF					0	808
SEA						0

Complete Linkage

{BOS, NY, CHI}			{DEN, SF, SEA}		
{BOS, NY, CHI}			{DEN}	{SF, SEA}	
{BOS, NY, CHI}			{DEN}	{SF}	{SEA}
{BOS, NY}		{CHI}	{DEN}	{SF}	{SEA}
{BOS}	{NY}	{CHI}	{DEN}	{SF}	{SEA}

Hierarchical Clustering



Pros

1. Intuitive
2. More informative than “flat clustering”
3. Deterministic
4. Does not require k to be pre-specified



Cons

1. Computationally expensive

```
class sklearn.cluster.AgglomerativeClustering (  
    n_clusters=2,  
    affinity='euclidean',  
    memory=None,  
    connectivity=None,  
    compute_full_tree='auto',  
    linkage='ward')
```

Agglomerative Clustering

Recursively merges the pair of clusters
that minimally increases
a given linkage distance.

```
class sklearn.cluster.AgglomerativeClustering (
```

```
    n_clusters=2,
```

```
    affinity='euclidean',
```

```
    memory=None,
```

```
    connectivity=None,
```

```
    compute_full_tree='auto',
```

```
    linkage='ward')
```

The number of clusters to find.

Similarity Measures

① Similarity between **data points**.

AFFINITY

② Similarity between **clusters**.

LINKAGE


```
class sklearn.cluster.AgglomerativeClustering (  
    n_clusters=2,  
    affinity='euclidean',  
    memory=None,  
    connectivity=None,  
    compute_full_tree='auto',  
    linkage='ward')
```

Metric used to compute the linkage.

Can be "euclidean", "l1", "l2", "manhattan",
"cosine", or "precomputed".

```
class sklearn.cluster.AgglomerativeClustering (  
    n_clusters=2,  
    affinity='euclidean',  
    memory=None,  
    connectivity=None,  
    compute_full_tree='auto',  
    linkage='ward')
```

**The linkage criterion determines
which distance to use
between sets of observation.**

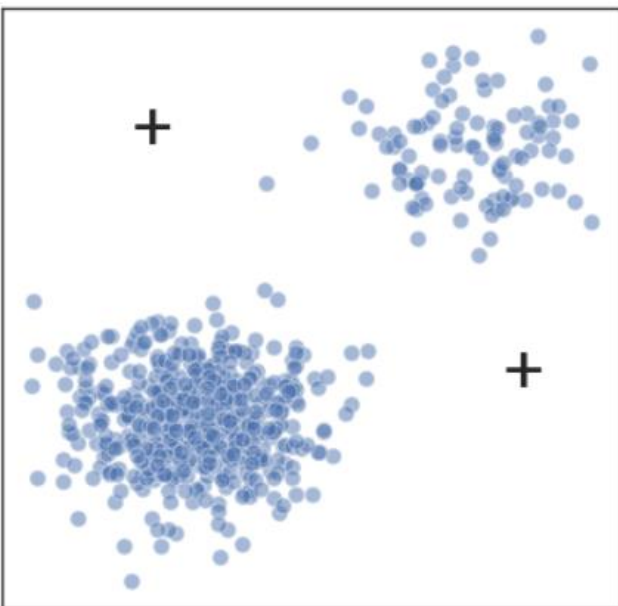
Supported criteria are “ward”, “complete”,
“average”, and “single”.

→ → → **Agglomerative Clustering Tutorial**

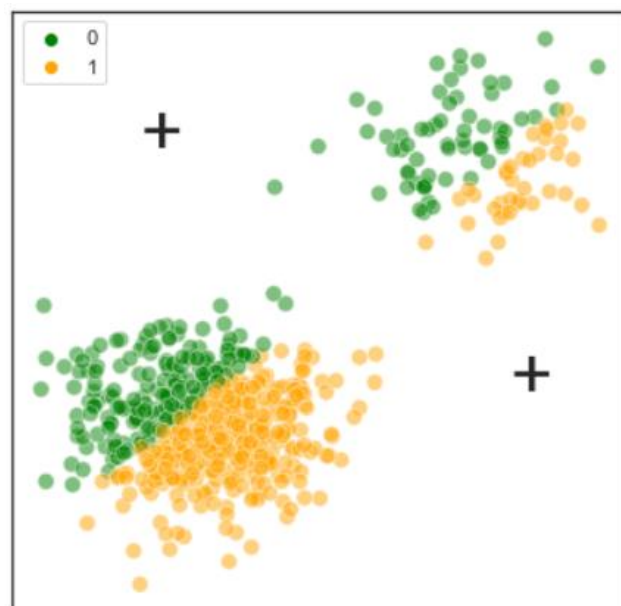
13_clustering_intro.ipynb

k-means Clustering

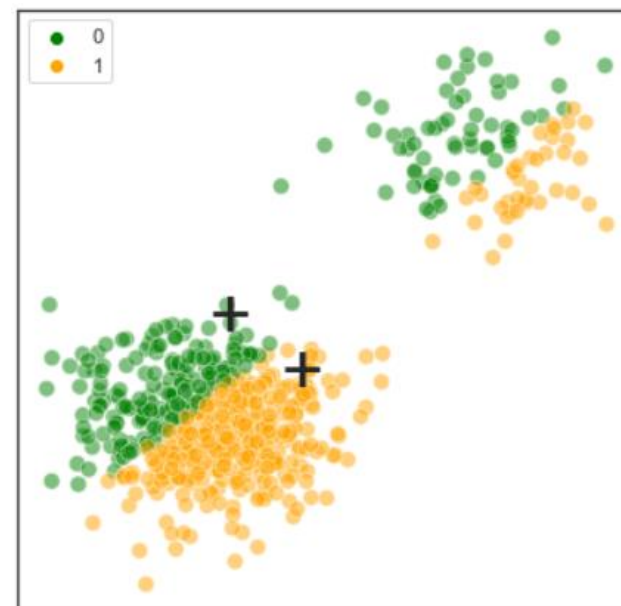
Iteration
#1



Initial Centroids

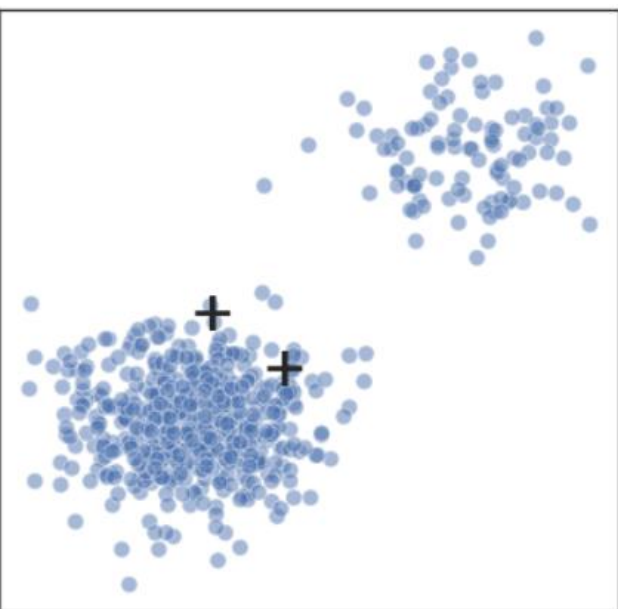


Assign Clusters

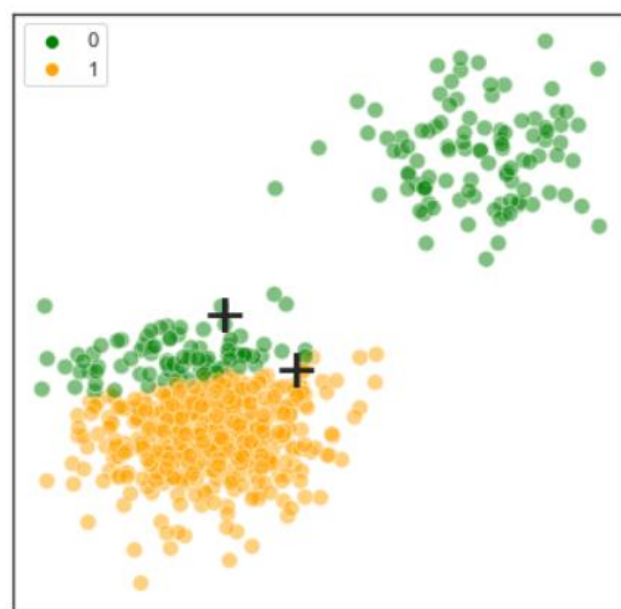


Recalculate Centroids

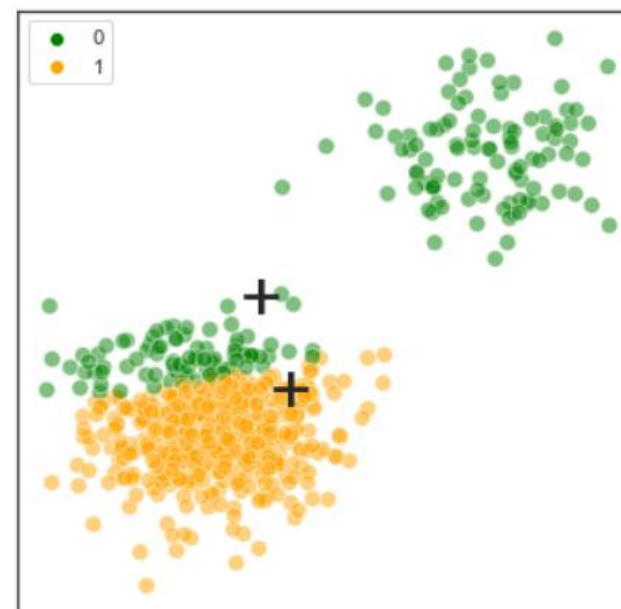
Iteration
#2



New Centroids

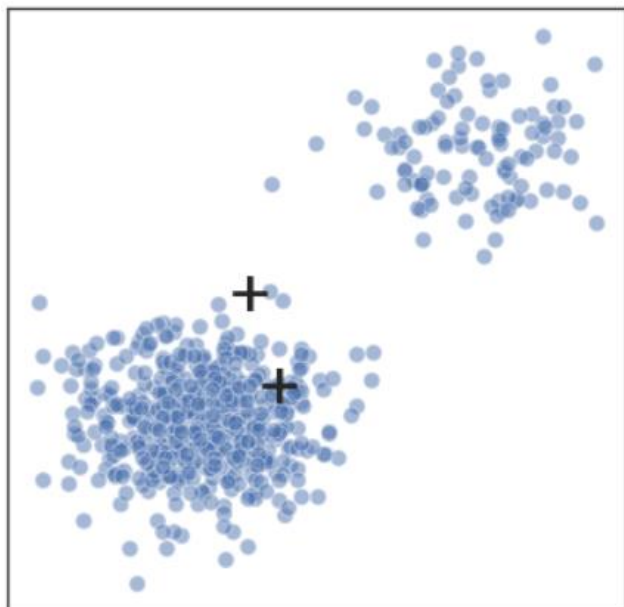


Assign Clusters

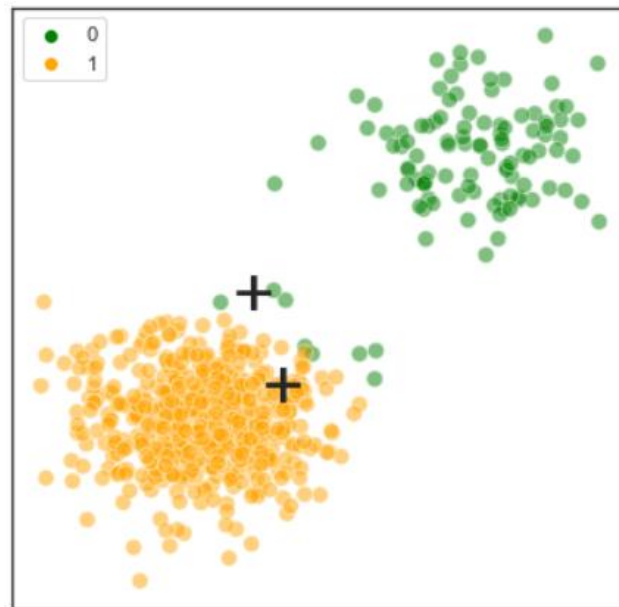


Recalculate Centroids

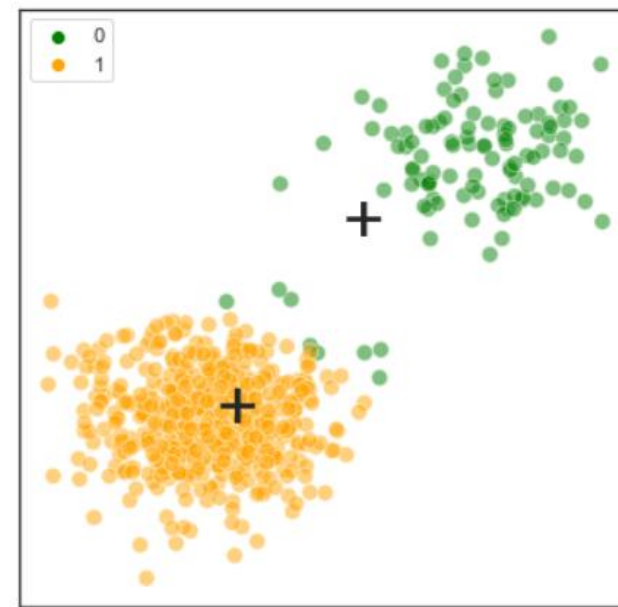
Iteration
#3



New Centroids

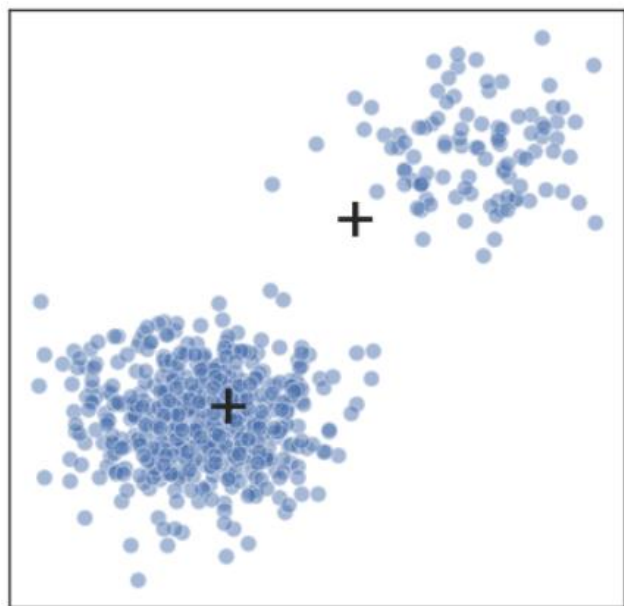


Assign Clusters

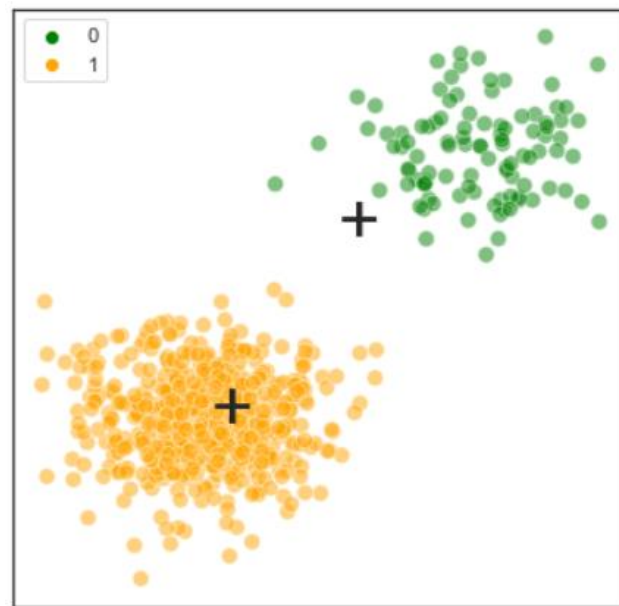


Recalculate Centroids

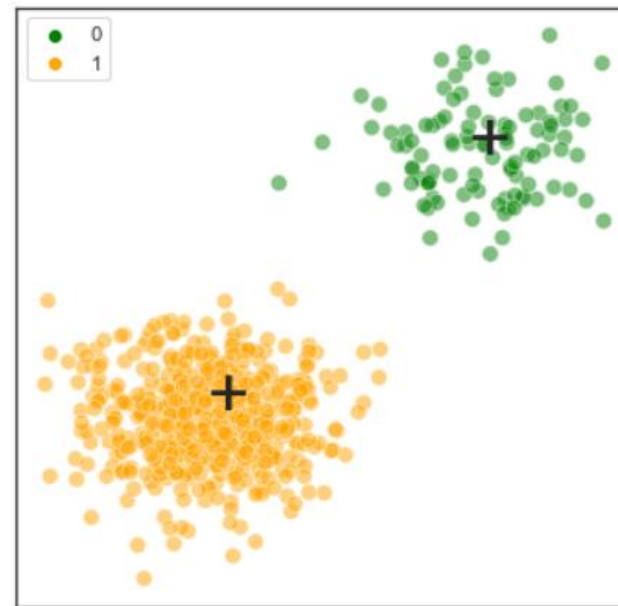
Iteration
#4



New Centroids



Assign Clusters



Recalculate Centroids

k -means Clustering: Steps

1. Provide the number of clusters (k).
2. Randomly initialize k centroids.
3. Assign data points to the nearest centroid.
4. Update the centroids.
5. Repeat steps 3 and 4.

k -means Algorithm

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

Given a set of observations (x_1, x_2, \dots, x_n) , where each observation is a d -dimensional real vector, k -means clustering aims to partition the n observations into k ($\leq n$) sets $S = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS). [\[Wikipedia\]](#)

k -means Algorithm

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

← Calculate squared distances from each data point to its cluster centroid

k -means Algorithm

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

Calculate squared distances from each data point to its cluster centroid

Take the sum of those squared distances within each cluster

k -means Algorithm

Add it across all clusters

k

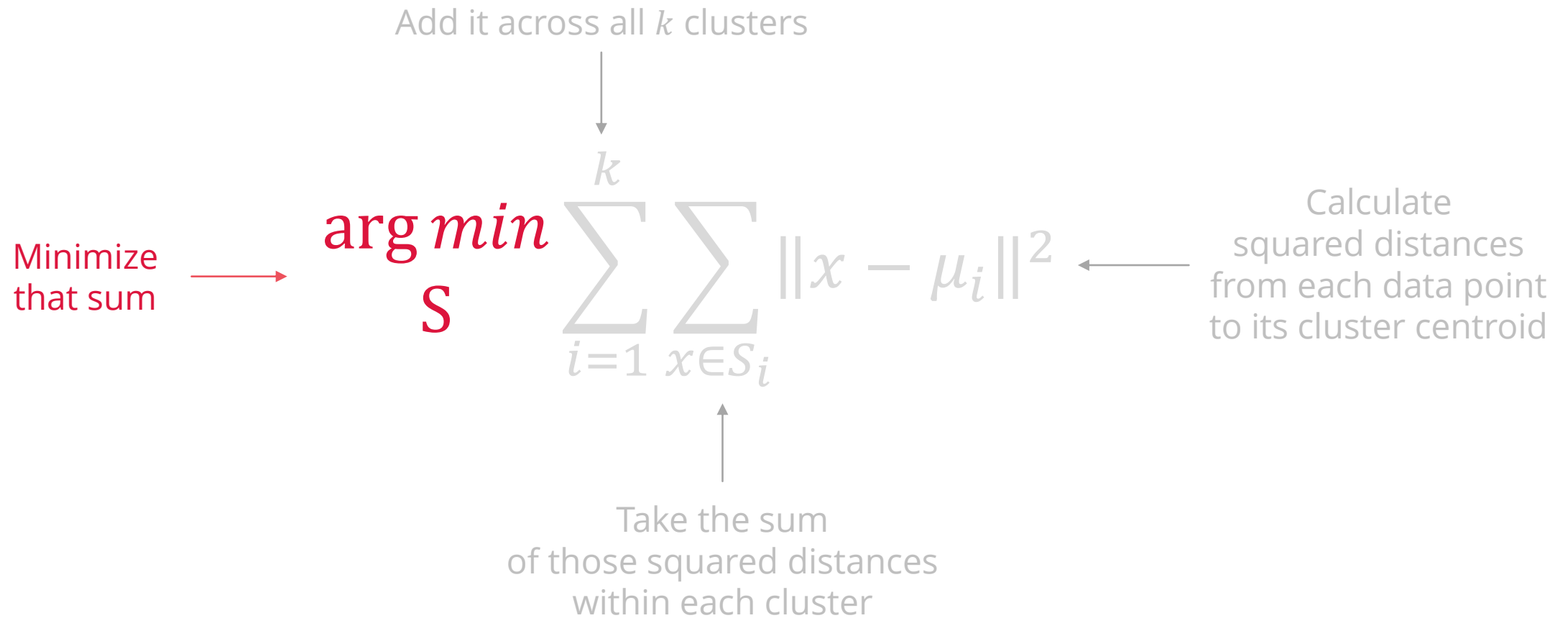
$\arg \min_S \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2$

Calculate squared distances from each data point to its cluster centroid

Take the sum of those squared distances within each cluster

The diagram illustrates the k-means objective function. At the top, the text 'Add it across all clusters' has a downward arrow pointing to the blue k in the summation index of the formula. To the right of the formula, the text 'Calculate squared distances from each data point to its cluster centroid' has an arrow pointing to the squared norm term $\|x - \mu_i\|^2$. Below the formula, the text 'Take the sum of those squared distances within each cluster' has an upward arrow pointing to the inner summation $\sum_{x \in S_i}$. The entire formula is written in a light gray font, with the exception of the k in the index and the $\arg \min_S$ term.

k -means Algorithm



k -means Algorithm

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

Given a set of observations (x_1, x_2, \dots, x_n) , where each observation is a d -dimensional real vector, k -means clustering aims to partition the n observations into k ($\leq n$) sets $S = \{S_1, S_2, \dots, S_k\}$ so as to **minimize the within-cluster sum of squares** (WCSS).

[Wikipedia]

k -means Clustering

Pros

1. Intuitive
2. Widely used and understood
3. Quick to execute

Cons

1. Assumes spherical clusters
2. Challenging to determine k
3. Initialization is important

```
class sklearn.cluster.KMeans (  
    n_clusters=8,  
    init='k-means++',  
    n_init=10,  
    max_iter=300,  
    tol=0.0001,  
    precompute_distances='auto',  
    verbose=0,  
    random_state=None,  
    copy_x=True,  
    algorithm='lloyd')
```

***k*-means Clustering**

```
class sklearn.cluster.KMeans (  
    n_clusters=8,  
    init='k-means++',  
    n_init=10,  
    max_iter=300,  
    tol=0.0001,  
    precompute_distances='auto',  
    verbose=0,  
    random_state=None,  
    copy_x=True,  
    algorithm='lloyd')
```

**The number of clusters to form
as well as the number of centroids
to generate.**


```
class sklearn.cluster.KMeans (  
    n_clusters=8,  
    init='k-means++',  
    n_init=10,  
    max_iter=300,  
    tol=0.0001,  
    precompute_distances='auto',  
    verbose=0,  
    random_state=None,  
    copy_x=True,  
    algorithm='lloyd')
```

Number of time the *k*-means algorithm will be run with different centroid seeds.

The final results will be the best output of `n_init` consecutive runs in terms of inertia (i.e., how much distance did the clusters move).

```
class sklearn.cluster.KMeans (  
    n_clusters=8,  
    init='k-means++',  
    n_init=10,  
    max_iter=300,  
    tol=0.0001,  
    precompute_distances='auto',  
    verbose=0,  
    random_state=None,  
    copy_x=True,  
    algorithm='lloyd')
```

**Maximum number of iterations
of the *k*-means algorithm
for a single run.**

```
class sklearn.cluster.KMeans (  
    n_clusters=8,  
    init='k-means++',  
    n_init=10,  
    max_iter=300,  
    tol=0.0001,  
    precompute_distances='auto',  
    verbose=0,  
    random_state=None,  
    copy_x=True,  
    algorithm='lloyd')
```

**Set a user-defined seed
for reproducible results.**

If `int`, `random_state` is the seed used by
the random number generator.

Recommendation: Always set a seed (e.g., 314) to
ensure reproducible results.

→ → → *k*-means Clustering Tutorial

13_clustering_intro.ipynb

DBSCAN

Density-Based Clustering: DBSCAN

Density-Based Spatial Clustering of Applications with Noise

ϵ (eps)

1

The minimum distance between two points for them to be considered **neighbors**.

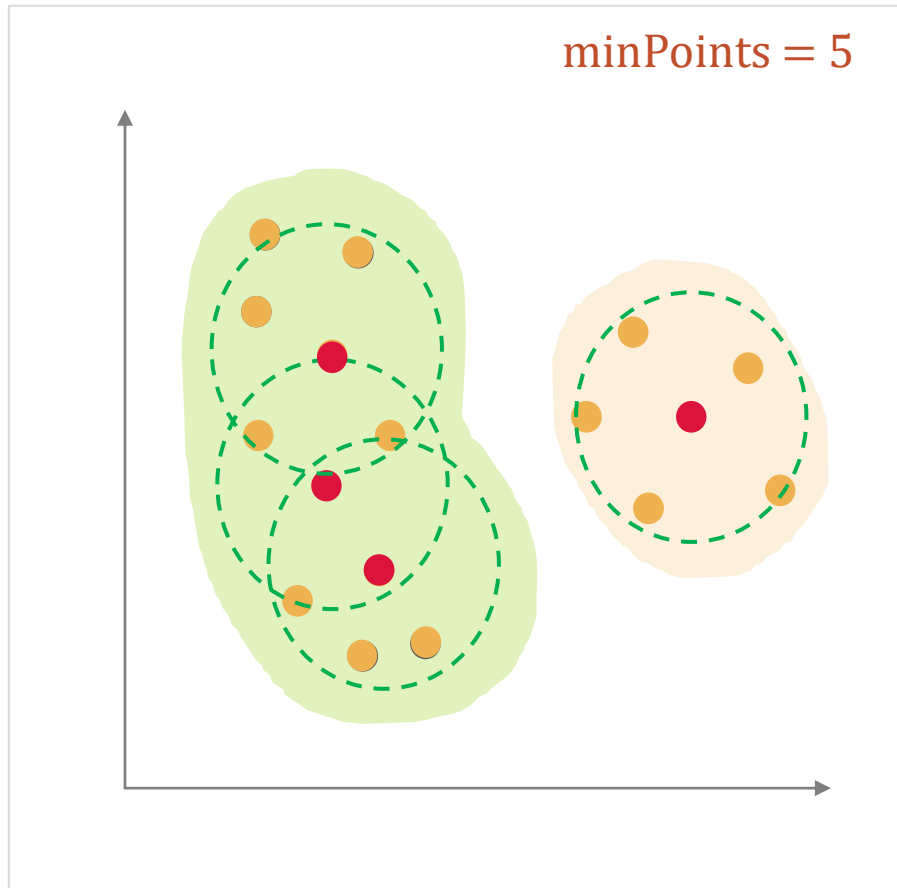
minPoints

2

The minimum number of points to form a dense **region**.

1. Find **core points**: points with at least minPoints points in their neighborhood (as defined by ϵ).
2. Find **boundary points**: points in the neighborhood of core points.
3. If two core points are near each other, assign them and all of their boundary points to the same cluster.

DBSCAN



ILLUSTRATIVE

1. Find **core points**: points with at least **five** points in their **neighborhood** (as defined by ϵ).
2. Find **boundary points**: points in the neighborhood of core points.
3. If two or more **core** points are near each other, assign them and all of their boundary points to the same cluster.

DBSCAN



Pros

1. Deterministic
2. Robust to noise
3. Can handle clusters of arbitrary shapes



Cons

1. Driven by density; requires connected regions to be of sufficiently high density
2. Difficulty in dealing with datasets with varying density


```
class sklearn.cluster.DBSCAN (  
    eps=0.5,  
    min_samples=5,  
    metric='euclidean',  
    metric_params=None,  
    algorithm='auto',  
    leaf_size=30,  
    p=None,  
    n_jobs=None)
```

**Perform DBSCAN clustering from
vector array or distance matrix.**

```
class sklearn.cluster.DBSCAN (
```

```
    eps=0.5,
```

```
    min_samples=5,
```

```
    metric='euclidean',
```

```
    metric_params=None,
```

```
    algorithm='auto',
```

```
    leaf_size=30,
```

```
    p=None,
```

```
    n_jobs=None)
```

**The maximum distance (ϵ)
between two samples
for them to be considered
as in the same neighborhood.**

```
class sklearn.cluster.DBSCAN (  
    eps=0.5,  
    min_samples=5,  
    metric='euclidean',  
    metric_params=None,  
    algorithm='auto',  
    leaf_size=30,  
    p=None,  
    n_jobs=None)
```

The number of samples
in a neighborhood
for a point to be considered
as a **core** point.

→ → → **DBSCAN Clustering Tutorial**

13_clustering_intro.ipynb

Clustering Methods: Summary

1

Distance-Based

Bad for:

- Non-globular clusters
- Clusters with different numbers of points

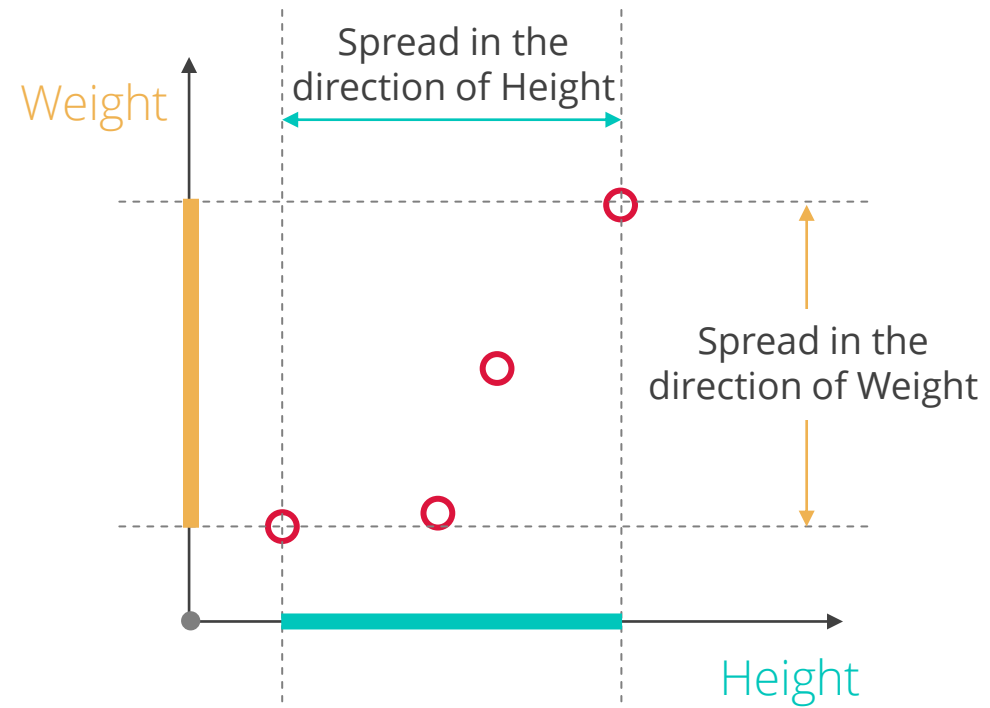
2

Density-Based

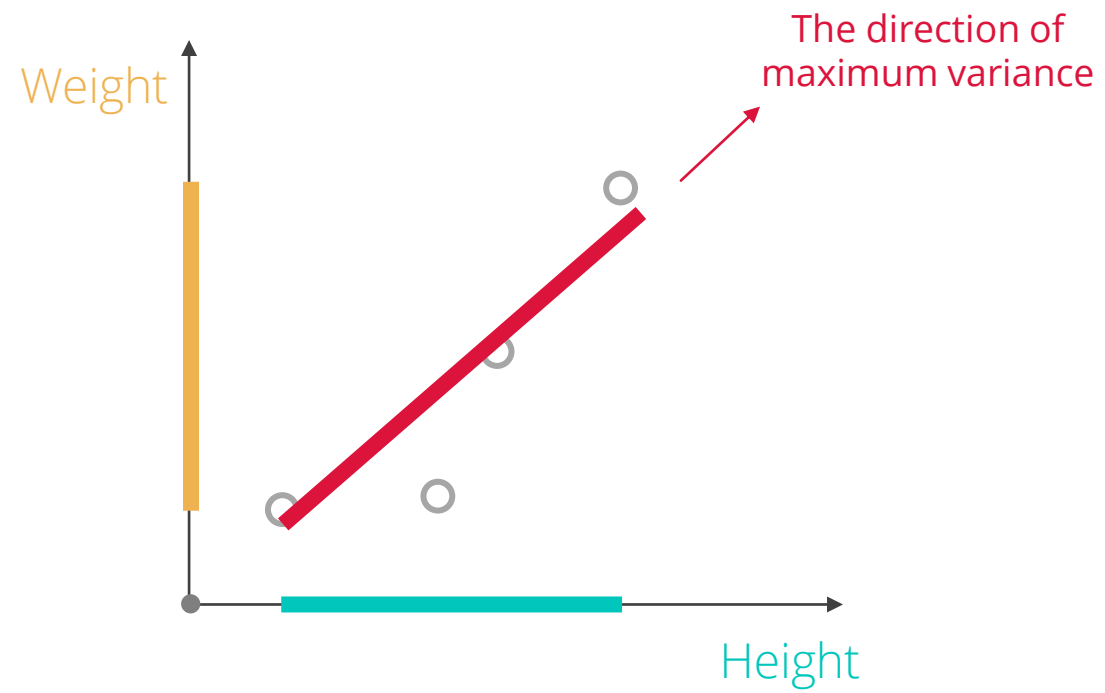
Bad for:

- Overlapping distributions
- Clusters with different densities

Principal Component Analysis

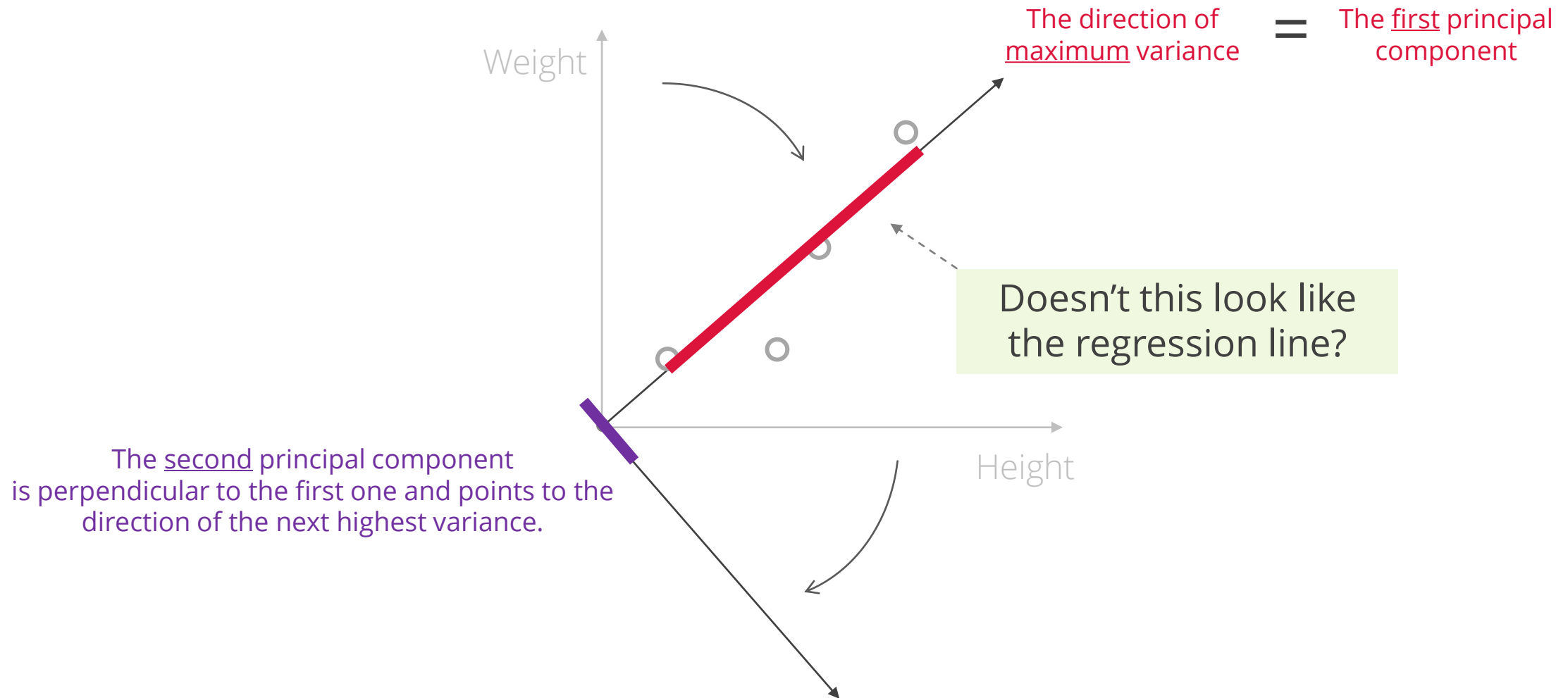


What is the direction of maximum spread?

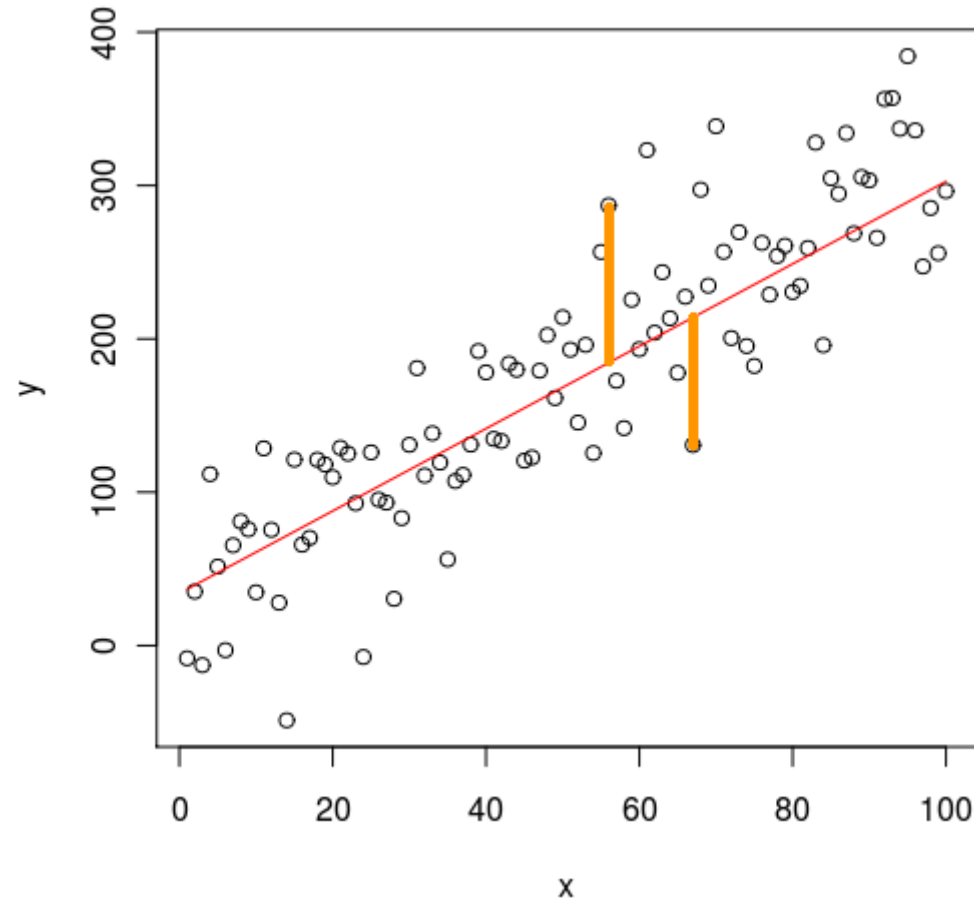


What is the direction of maximum spread?

A Change in Perspective

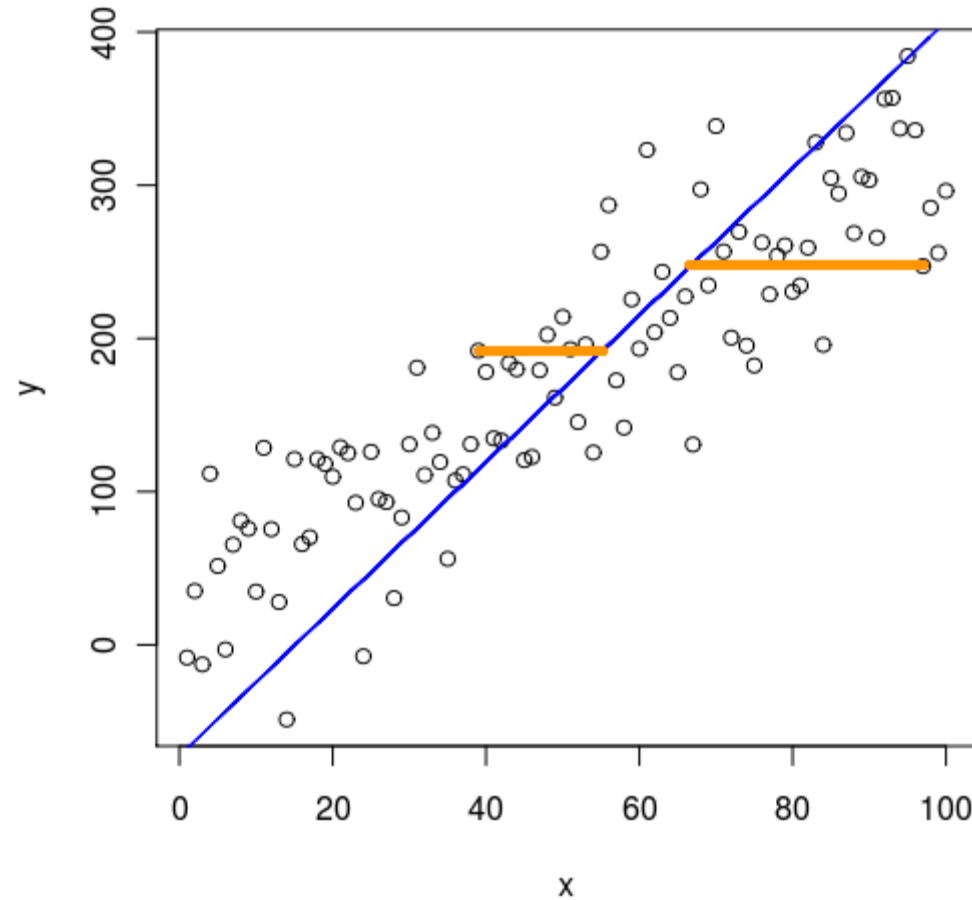


Regression Model: $y \sim x$



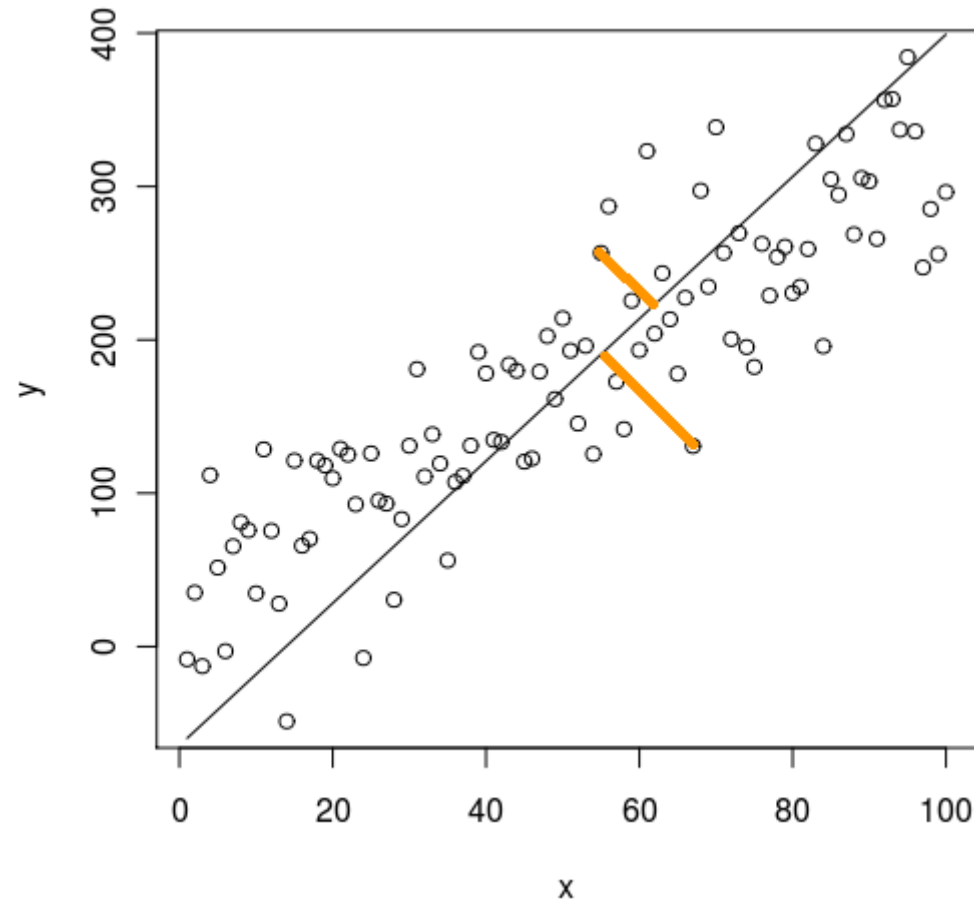
The $y \sim x$ regression model minimizes the error in the **vertical** (y) direction.

Regression Model: $x \sim y$



The $x \sim y$ regression model minimizes the error in the **horizontal** (x) direction.

PCA



PCA minimizes the error **orthogonal** (perpendicular) to the model line!

[Principal component analysis \(PCA\) vs ordinary least squares \(OLC\): a visual explanation](#)

Rotation of the Axes [ILLUSTRATIVE]

Height	Weight
1.74	59
1.69	75
1.75	55
1.50	51

In the original data set, each row is expressed in terms of the following two dimensions (axes):
Height and Weight.

You can now choose to **drop** the second Principal Component if it doesn't help explain much variance.

PC1	PC2
0.14	0.91
-0.78	0.08
0.65	0.18
0.13	-0.76

After the PCA is performed, the data set is expressed in terms of the two new dimensions (axes):
Principal Component #1 and #2.

**Feature
Reduction**

Chaining PCA with a Model



```
class sklearn.decomposition.PCA (  
    n_components=None,  
    copy=True,  
    whiten=False,  
    svd_solver='auto',  
    tol=0.0,  
    iterated_power='auto',  
    random_state=None)
```

**Linear dimensionality reduction
using Singular Value Decomposition
of the data to project it to a lower
dimensional space.**

```
class sklearn.decomposition.PCA (  
    n_components=None,  
    copy=True,  
    whiten=False,  
    svd_solver='auto',  
    tol=0.0,  
    iterated_power='auto',  
    random_state=None)
```

Number of components to keep.

If `n_components` is not set then
all components are kept


```
class sklearn.decomposition.PCA (  
    n_components=None,  
    copy=True,  
    whiten=False,  
    svd_solver='auto',  
    tol=0.0,  
    iterated_power='auto',  
    random_state=None)
```

**Set a user-defined seed
for reproducible results.**

If `int`, `random_state` is the seed used by
the random number generator.

Recommendation: Always set a seed (e.g., 314) to
ensure reproducible results.

→ → → **PCA Tutorial**

14_pca.ipynb

Evaluating Clusters




1 The observer's vantage point



1 The observer's vantage point

2 The projection space (number of dimensions)



Bellatrix
245 light years

Betelgeuse
624 light years

Alnilam
1,342 light years

Rigel
772 light years

Reification



Pure Fiction

Convenient ways of summarizing data,
with no other meaning

Objective Truth

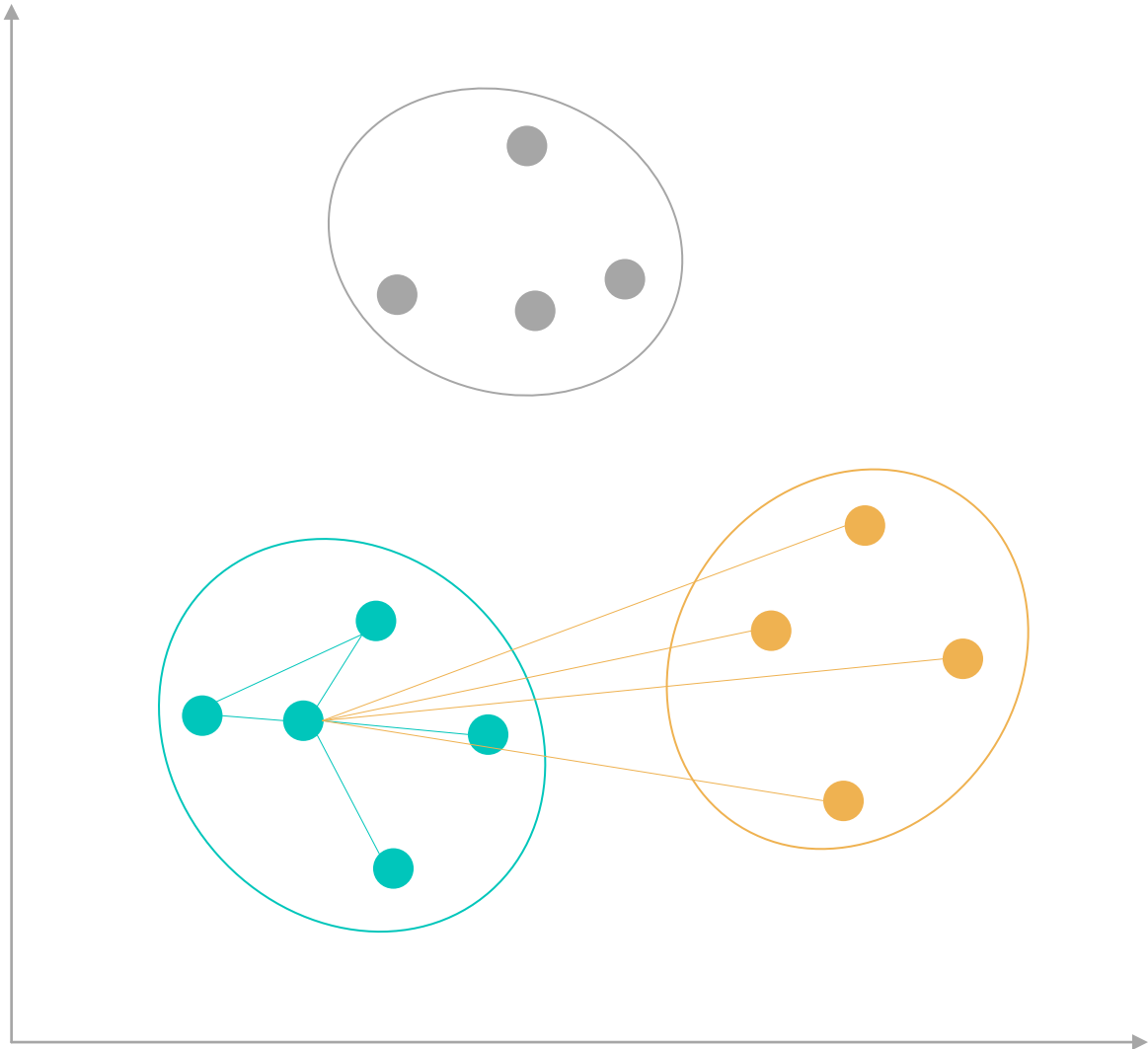
Reflections of the real divisions
of the world into distinct types

- 1 Clusters should generalize well.
- 2 Clusters should generalize to new features.
- 3 Clusters should fit into a theory or narrative.

Evaluating Clusters

1. **Silhouette scores**
2. Segment (Cluster) profile
3. ANOVA and MANOVA
4. Cubic Clustering Criterion (CCC)
5. Predict cluster membership using classification

Silhouette Score



a = Average distance to all data points within its own cluster

This is a measure of how well a point is assigned to its cluster. The smaller the value, the better the assignment.

b = Average distance to all data points within the closest cluster

Silhouette Score

$$s = \frac{b - a}{\max(a, b)}$$

a = Average distance to all data points within its own cluster

b = Average distance to all data points within the closest cluster

Silhouette Score

$$s = \frac{b - a}{\max(a, b)}$$

a = Average distance to all data points within its own cluster

b = Average distance to all data points within the closest cluster

$$s = \begin{cases} 1 - a/b, & \text{if } a < b \\ 0, & \text{if } a = b \\ b/a - 1, & \text{if } a > b \end{cases}$$

Therefore: $-1 < s < 1$

For S to be close to 1 we require $a \ll b$.

→ → → **PCA + Clustering Tutorial**

15_clustering.ipynb

→ → → **Homework**

DAPT-631 Homework Assignment #4

Due date: 12-MAY-2024