# Unsupervised Learning

Sammie Omranian

Summer 2024

# Unsupervised Learning

- Unsupervised learning involves training a model to find patterns in a dataset without pre-labeled responses.

Supervised vs. Unsupervised Learning

- Supervised Learning: Labeled data, goal is to predict outcomes.
- Unsupervised Learning: Unlabeled data, goal is to find hidden patterns.

# Unsupervised Learning

Three common unsupervised learning algorithms:

- Clustering: Grouping similar items.
- Hierarchical Clustering: to build a hierarchy of clusters
- Dimensionality Reduction: Simplifying data without losing important information.

# Clustering

- Clustering is a data mining technique which groups unlabeled data based on their similarities or differences.

- Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information.

- Clustering algorithms can be categorized into a few types, specifically
  - Exclusive
  - Hierarchical

# Clustering


source

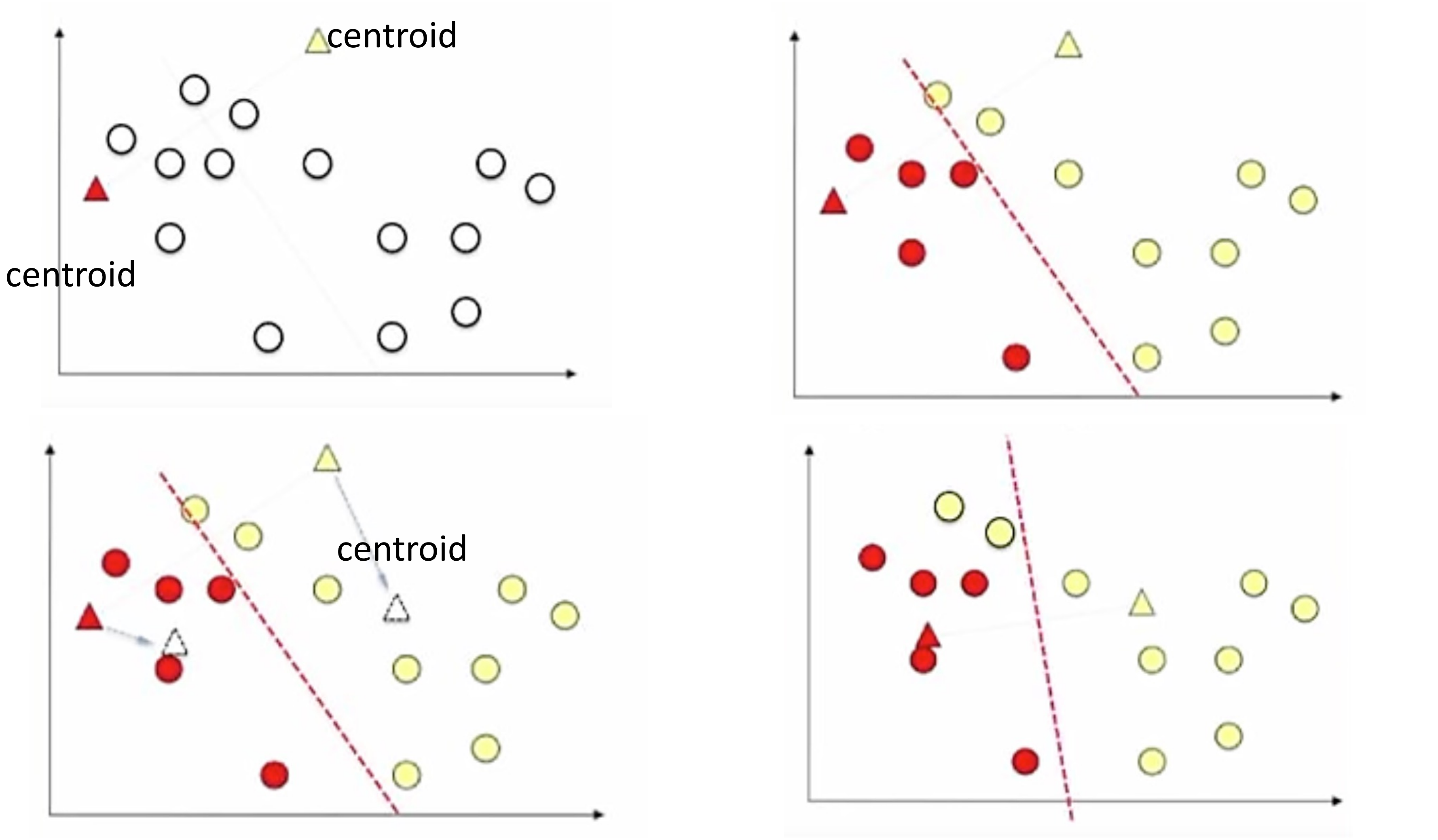
# K-means clustering

- The **K-means clustering** algorithm is an example of exclusive clustering.
- K-Means clustering aims to group n data points into k clusters in which each observation belongs to the cluster with the nearest mean.
- A larger K value will be indicative of smaller groupings whereas a smaller K value will have larger groupings and less granularity.
- K-means clustering is commonly used in market segmentation, document clustering, and image segmentation.

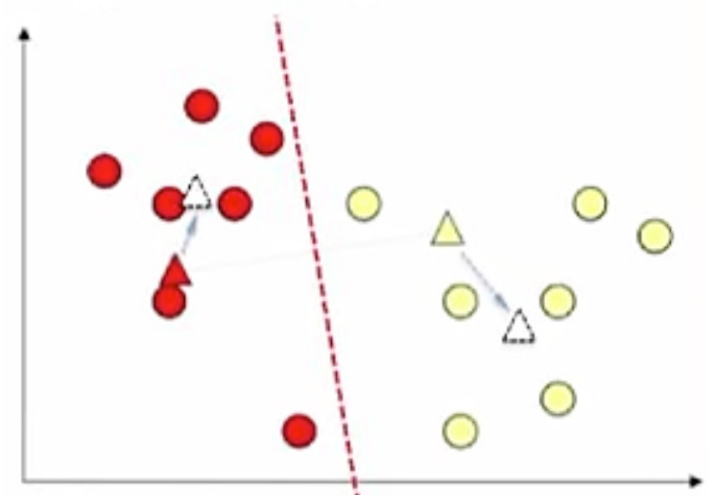
# K-means clustering

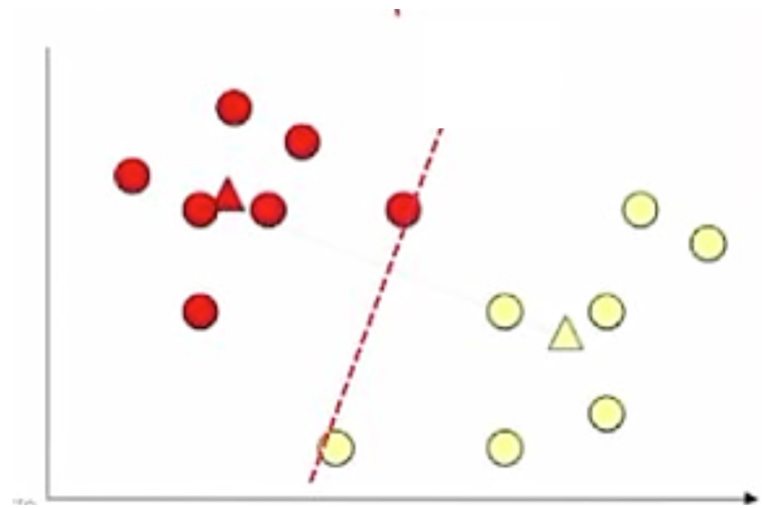
- input $k$ , set of points $x_1, x_2, \dots, x_n$
- starts by placing K points (centroids: $c_1, c_2, \dots, c_k$) at random locations in space.
- repeat until convergence:
  - for each point $x_i$ :
    - find nearest centroid $c_j$
    - assign the point $x_i$ to cluster j
  - for each cluster j = 1, 2, …, k:
    - new centroid $c_j$ = mean of all points $x_i$ assigned to cluster j in previous step

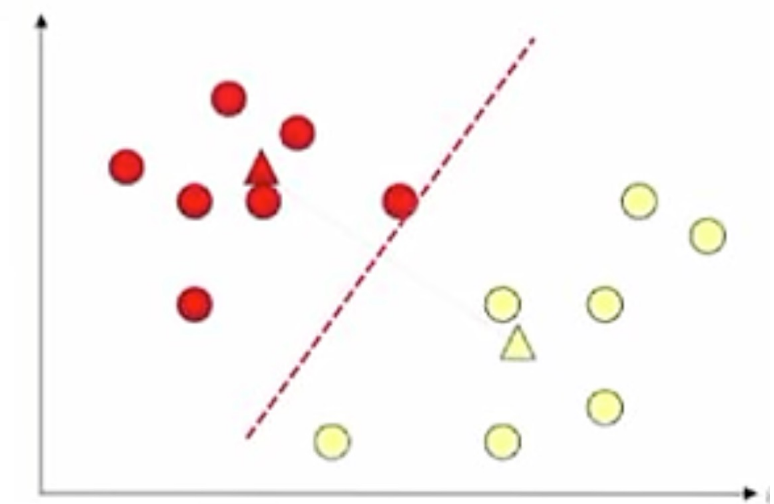
# K-means clustering



# K-means clustering







Convergence!

# K-means clustering

## KMeans

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

initialization method of centroids

Number of clusters

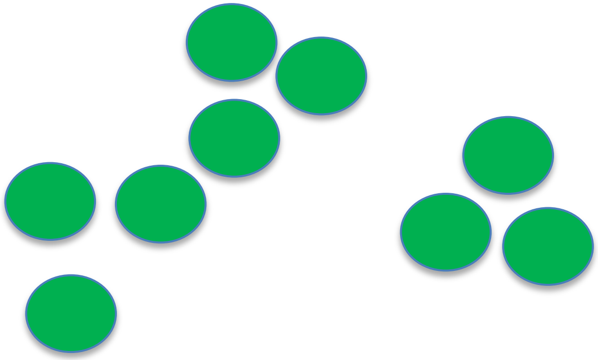
[source]()

# Hierarchical Clustering

Selecting K! Question of granularity

How coarse or fine-grained is the structure in your data?

The main problem of K-means is what a good k?
No clustering algorithm can pick k.



How many clusters do you see?

# Hierarchical Clustering

Instead of picking K, let's find a hierarchy of the structure:

At the top level – coarse grained

At low levels – fine grained

How to cluster:

Topmost cluster → have every points in the cluster

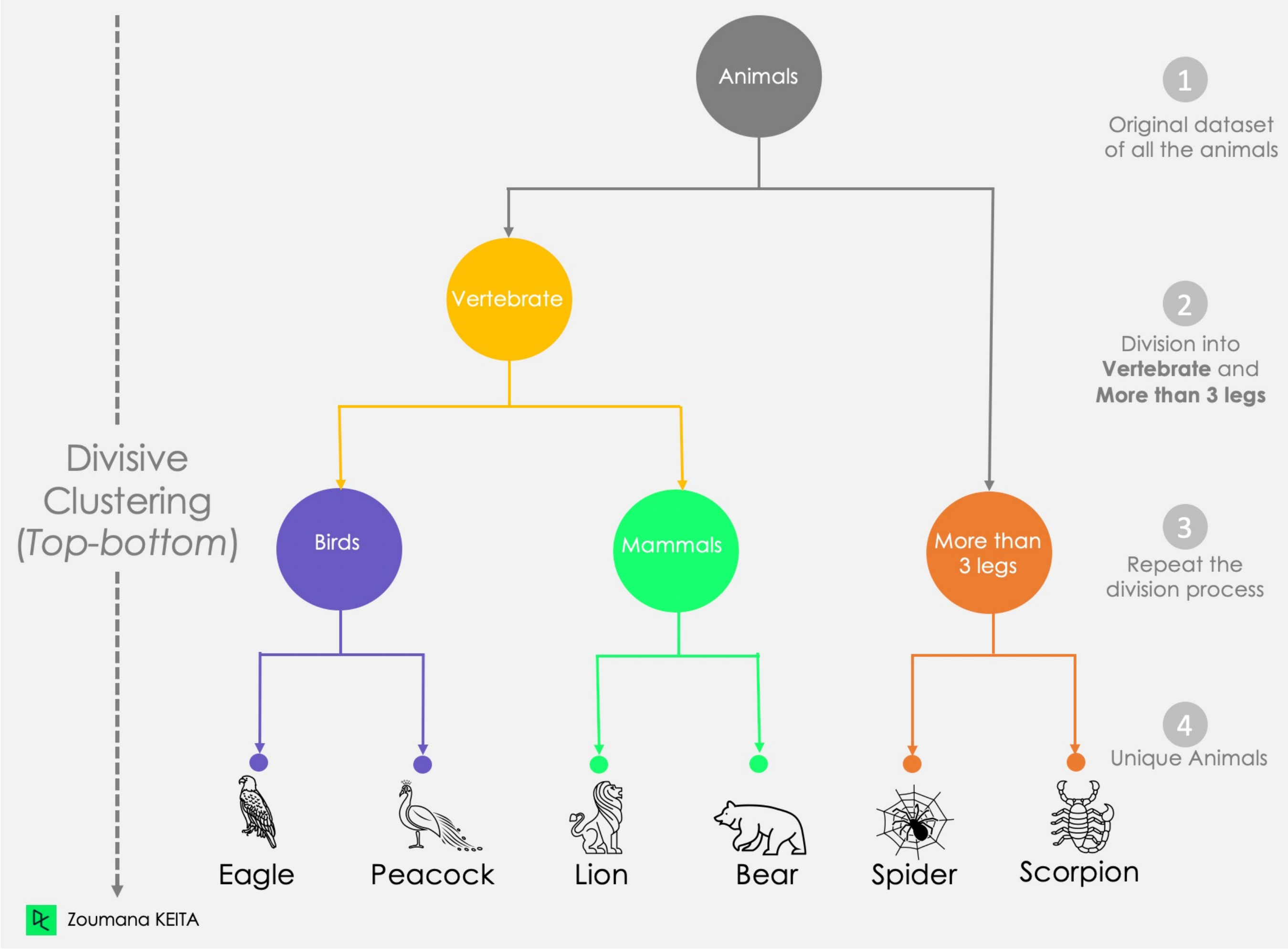
Bottom clusters → have n set of singletons

So, you will end up with a tree!

# Hierarchical Clustering

Two types of algorithm

- Divisive(Top-bottom)
- agglomerative (Bottom-up)




[Image source]()

# Hierarchical Clustering

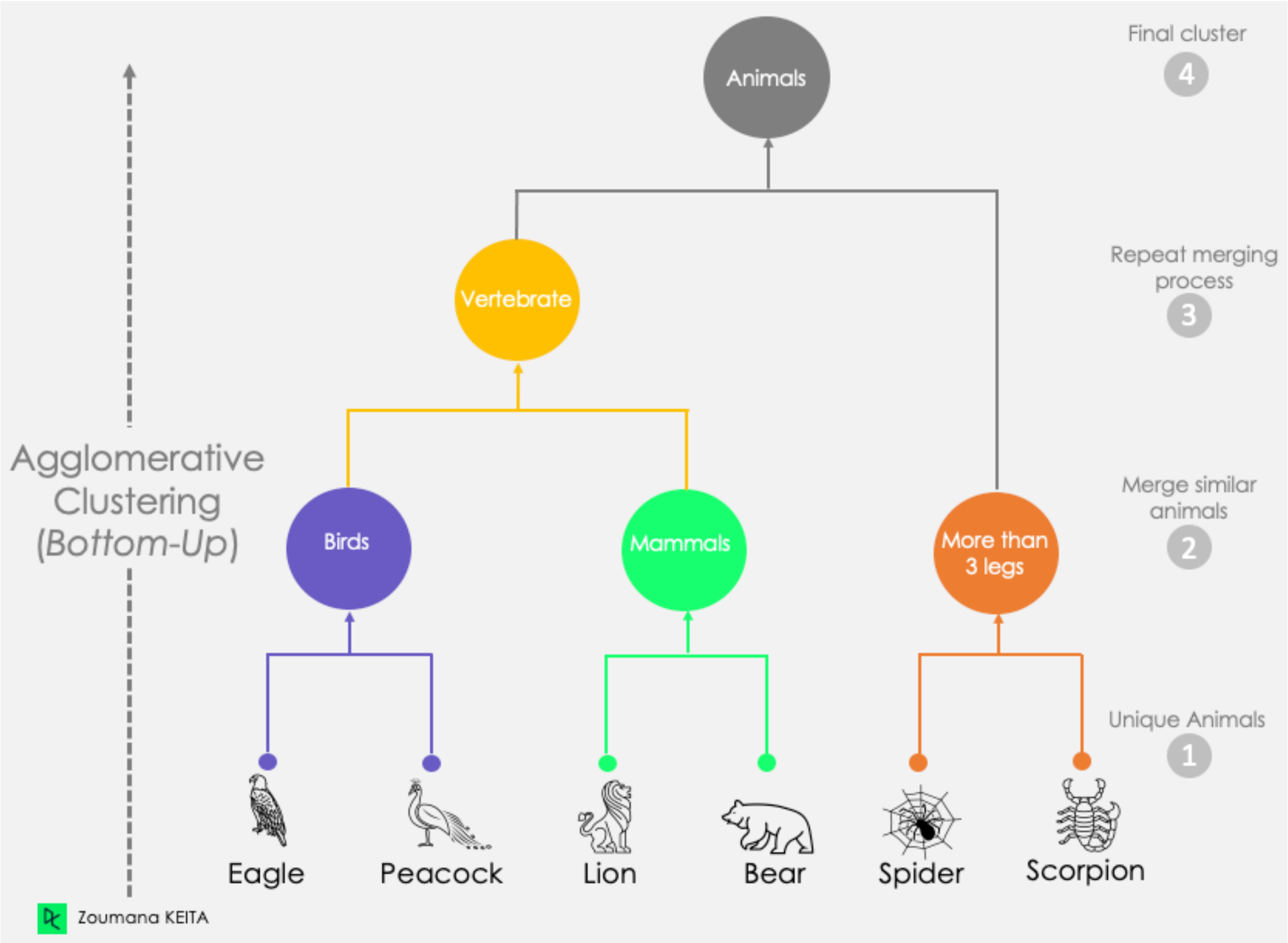



Image source

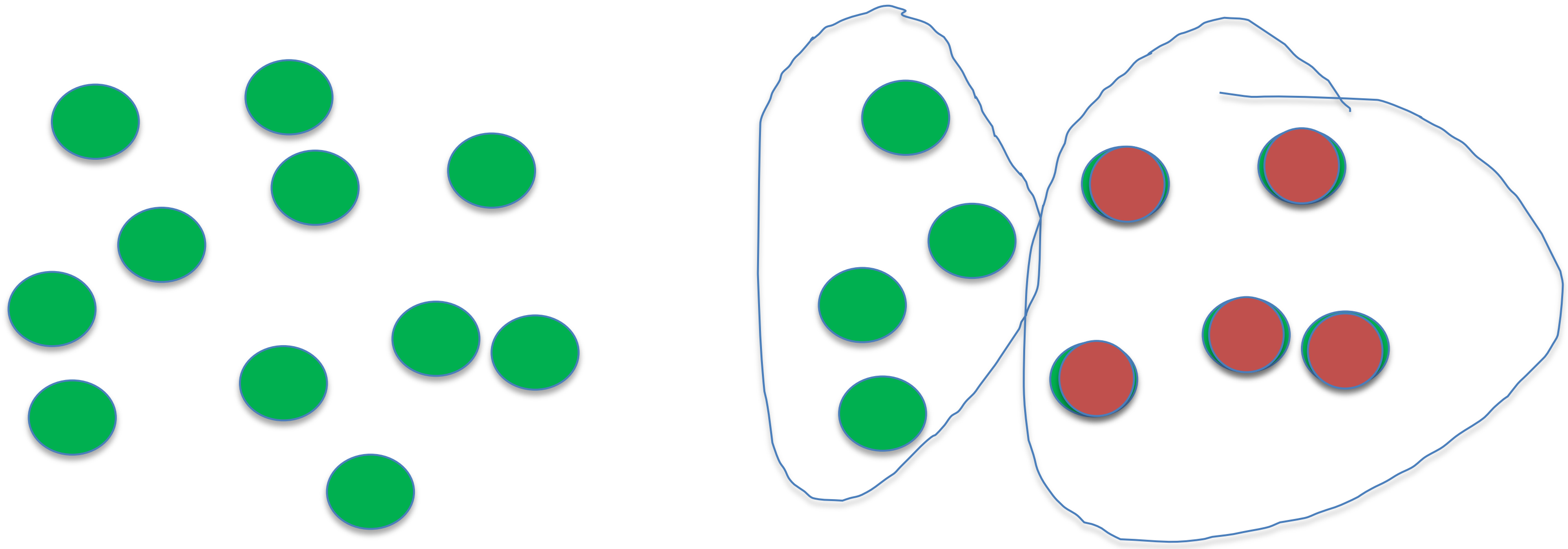
# Hierarchical Clustering

Hierarchical algorithm (top-down approach):

Run K-means algorithm on the original data $x_1, x_2, \ldots, x_n$

For each of the resulting clusters $c_i$,  i = 1, 2, …, k

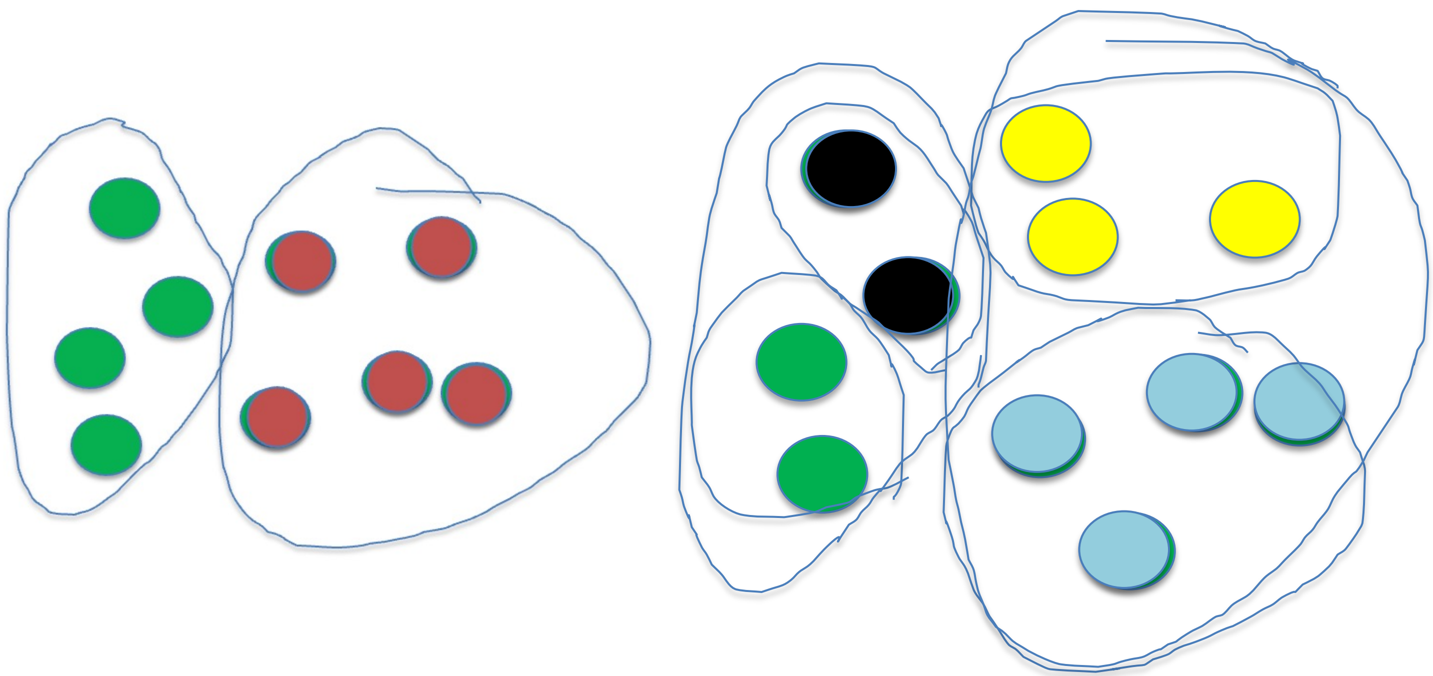
Recursively run K-means on points in cluster $c_i$



# Hierarchical Clustering

Hierarchical algorithm (top-down approach):

- run K-means algorithm on the original data $x_1, x_2, \ldots, x_n$
- for each of the resulting clusters $c_i$, i = 1, 2, …, k
  - recursively run K-means on points in cluster $c_i$



# Hierarchical Clustering

Applications of Hierarchical Clustering

- Biology
  - The clustering of DNA sequences is one of the biggest challenges in bioinformatics.

- Image processing
  - Hierarchical clustering can be performed in image processing to group similar regions or pixels of an image in terms of color, intensity, or other features.

- Social network analysis
  - Hierarchical clustering can be used to identify groups or communities and to understand their relationships to each other and the structure of the network as a whole.

# Hierarchical Clustering

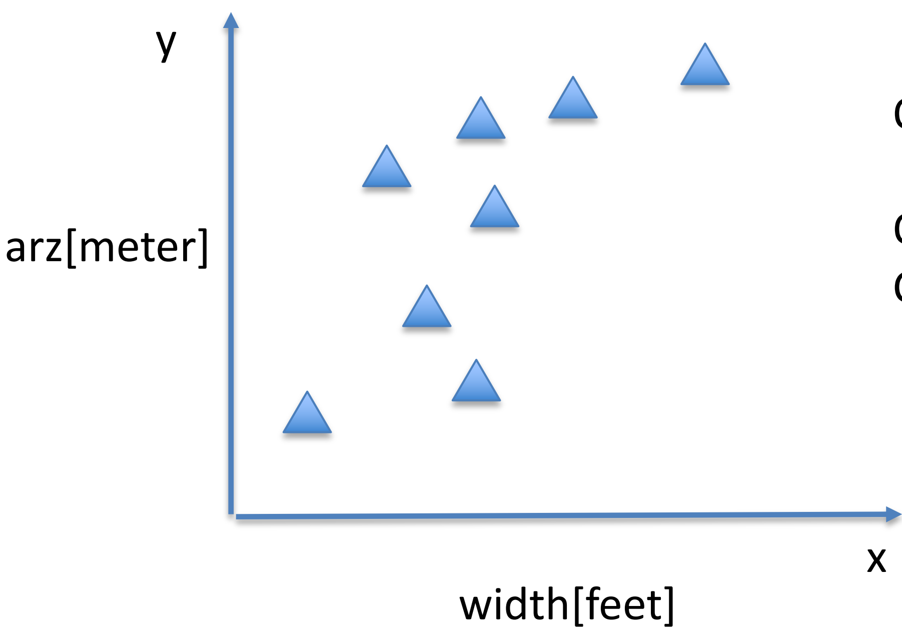
- **scipy.cluster.hierarchy.linkage(*y*, *method*=*'single'*, *metric*=*'euclidean'*, *optimal_ordering*=*False*)**

- source

# Principal Component Analysis (PCA)

- Suppose we have a dataset and we want to find the dimensionality of the data

- Each data point represent a pair (x, y), where x and y are real numbers.




Question: really two dimension?

On the surface is but what is under the surface? Can I find out?

# Principal Component Analysis (PCA)

- A dataset including 5 features:

  - X1: number of car accidents
  - X2: number od burst water pumps
  - X3: number of school closure
  - X4: number of patients with broken legs or arms
  - X5: snowplow cost

  - All of these are strongly influence by one factor! What is that?

# Principal Component Analysis (PCA)

Applications:

Texts - each word a separate feature

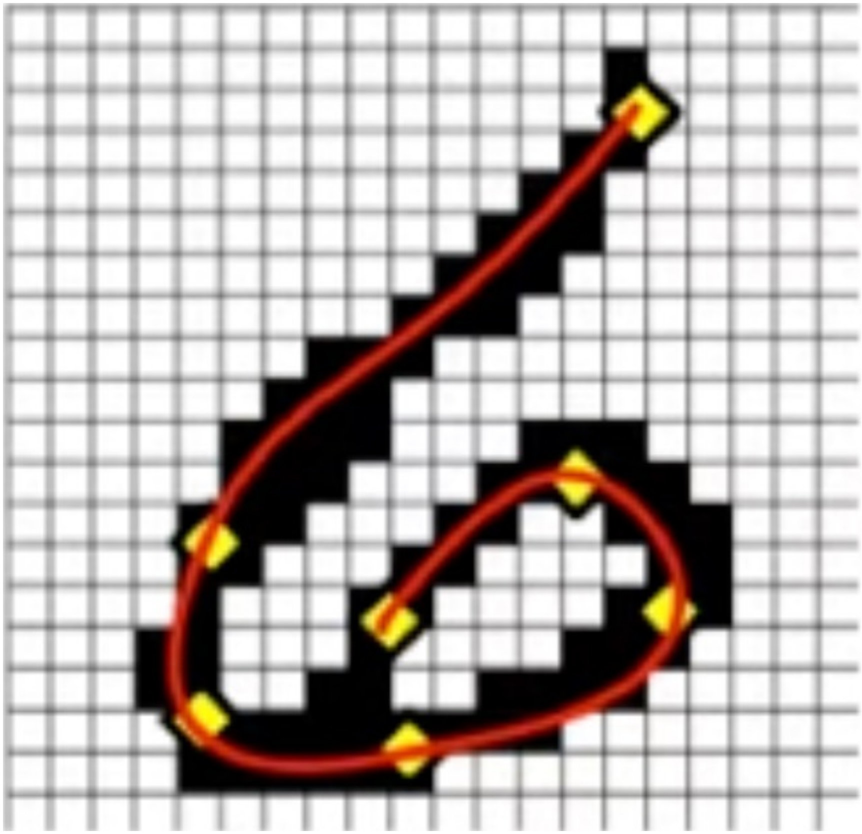
Image - each pixel a separate feature

Millions of features!!!

High dimentional

# Principal Component Analysis (PCA)

Example 1



Example 2

"Grouping, clustering, and categorizing data points into nested, ordered, and structured layers exemplifies the process of hierarchical algorithms."

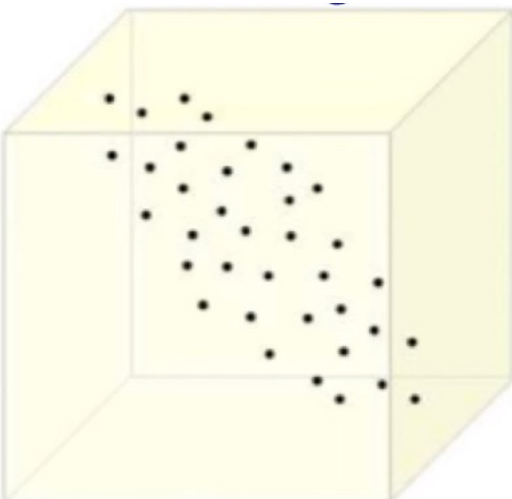
"Grouping data into nested layers illustrates hierarchical algorithms."

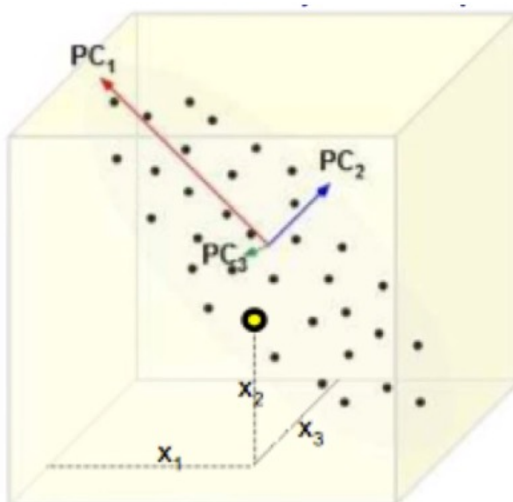
# Principal Component Analysis (PCA)

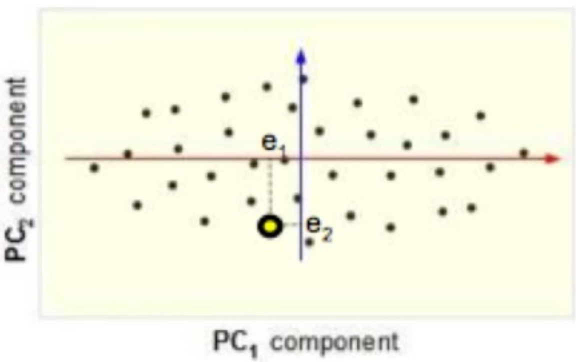
- We suppose we have our low dimensional data embedded in a high dimension space.
- Defines a set of principal components
  - 1st : direction of the greatest variability in the data
  - 2nd : perpendicular to 1st , greatest variability of what's left
  - Repeat this until d (original dimensionality)

- First m<<d components become m new dimensions
  - Change coordinates of every datapoint to these dimensions

# Principal Component Analysis (PCA)





# Principal Component Analysis (PCA)

PCA

*class* sklearn.decomposition.**PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None*)

source