2019

Data Science and AI

Module 6

Unsupervised Classification and Clustering

Agenda: Module 6

- **Clustering and Classification**
- **K-Means**
- **K-Nearest Neighbours**
- **DBSCAN**
- **Hierarchical Clustering**

Clustering and Classification

- Introduction
- Clustering Approaches

Supervised versus unsupervised learning

- Most of what we covered so far focuses on **supervised learning** methods such as regression and classification.
- In that setting we observe both a set of features X1,X2,...,Xn for each observation, as well as a response or outcome variable y . The goal is then to **predict** y using X1,X2,...,Xn.
- In unsupervised learning we instead focus on unsupervised learning, we where **observe only the features** X1, X2, . . . , Xn.
- We are not interested in prediction, because we **do not have an associated response variable** y .

Unsupervised learning

- The goal in unsupervised learning is to **discover** interesting things about the measurements:

- is there an informative way to **visualise** the data?

- Can we discover **subgroups** among the variables or among the observations?

Challenges and advantages with unsupervised learning

- Unsupervised learning is **more subjective** than supervised learning, as there is **no simple goal** for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of **growing importance** in a number of **use cases**:
  - subgroups of breast cancer patients **grouped by their gene** expression measurements,
  - groups of shoppers characterized by their **browsing and purchase histories**,
  - movies **grouped by the ratings** assigned by movie viewers.
- It is often **easier to obtain unlabelled data** — from a lab instrument or a computer — than labelled data, which can require human intervention.
  - For example it is difficult to automatically assess the overall sentiment of a movie review: is it favourable or not?

Clustering

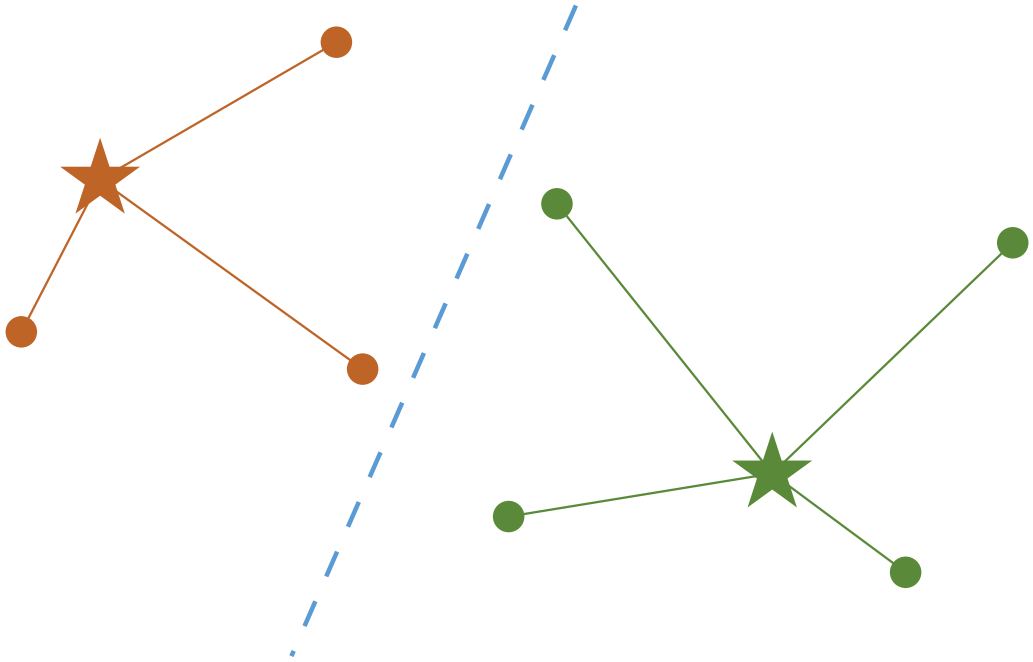
- The task of **aggregating** a set of objects in groups (clusters)
  - Objects in a group are more alike (in some sense) than to those in other groups
- It is a general task to be solved not by one specific algorithm
  - Algorithms can differ on **how to characterise clusters** and **how to find them**

Cluster Membership

- Popular notions of clusters
  - **Distances** between cluster members
  - **Dense areas** of the data space
  - **Intervals** or particular statistical distributions

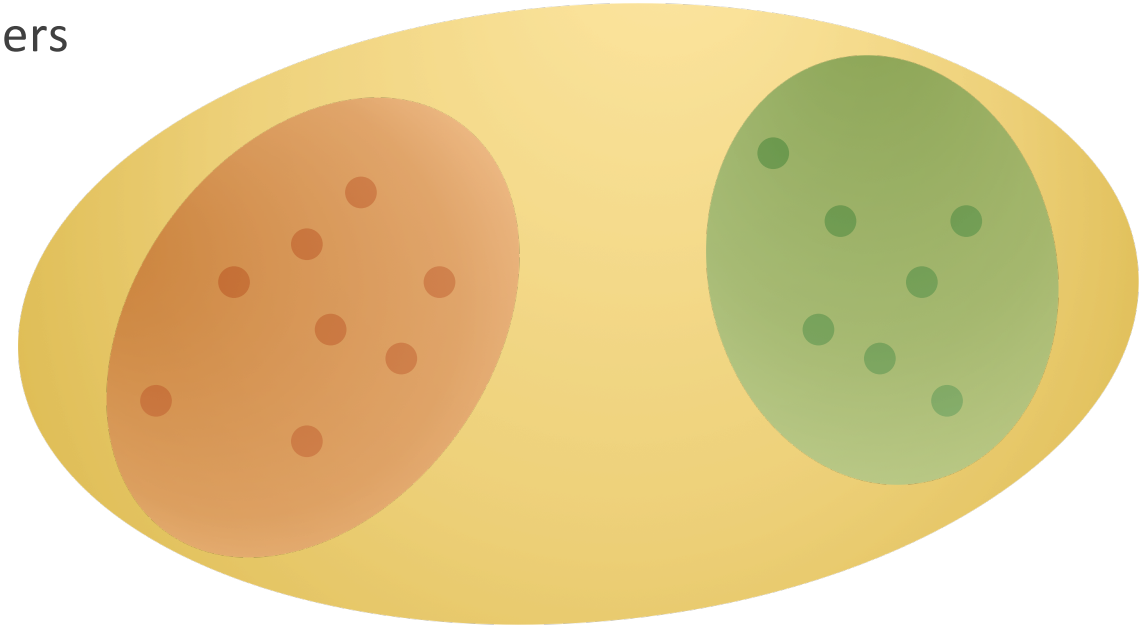
Clustering Approaches

- **Distance** from centre points
  - Determines membership by the smallest distance to known centres



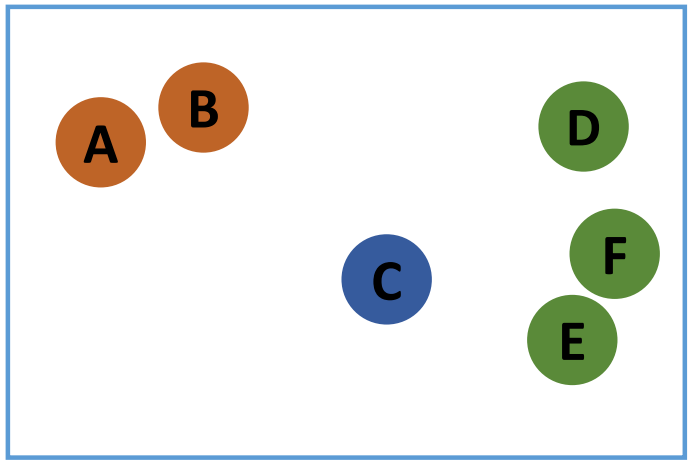
Clustering Approaches

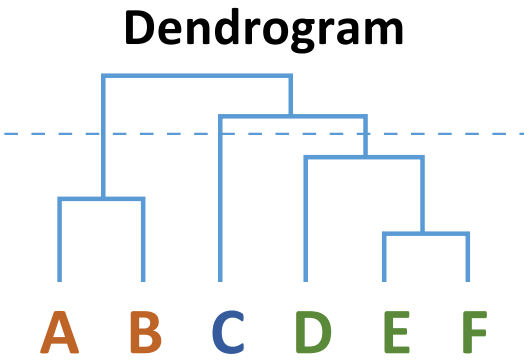
- **Density, Proximity** to other objects
  - Contrast **areas of higher density** with the remainder of the dataset
  - **Sparse areas** separate clusters



Clustering Approaches

- **Hierarchy**
  - Some measure of **proximity** determine the relationship between objects
  - Discovery is made by Division (**Top-Down**) or Aggregation (**Bottom-Up**)





Clustering

- The purpose of the **analysis** and the **characteristics of the dataset** determine the **appropriate** clustering method and its parameters
- A good clustering is one that achieves: **High within-cluster similarity Low inter-cluster similarity**
- Clustering **is not an automatic task**
  - It requires an **iterative process of discovery** that involves trial and failure
  - It is often necessary to modify **data preprocessing** and **model parameters** until the result achieves the desired properties

- Have applications in **pattern recognition**, **data compression**, **image analysis**, **bioinformatics** and others

K-Means

- Overview
- **Algorithm**
- **Assumptions**
- **Pros and Cons**
- K-Means in Python (Lab)
- Improving Performance accuracy

K-Means Overview

- Clustering method
- Goal: Group the examples into K **"homogeneous"** partitions
- Loosely speaking, it is classification without **labels**
- Aims to partition a set of observations into some number of clusters (**K**)
  - Related to the Lloyd's Algorithm
  - Data are split into cells of a Voronoi Diagram

K-Means Overview

- Method of finding out to **which group** a specific object belongs

- Finds objects by the **proximity to central average points** (**Centroids**). k-means aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

- More precisely, find subsets $S_1, S_2, S_3, \dots, S_k$ of the data with means $\mu_1, \mu_2, \mu_3, \dots, \mu_k$ that minimises:

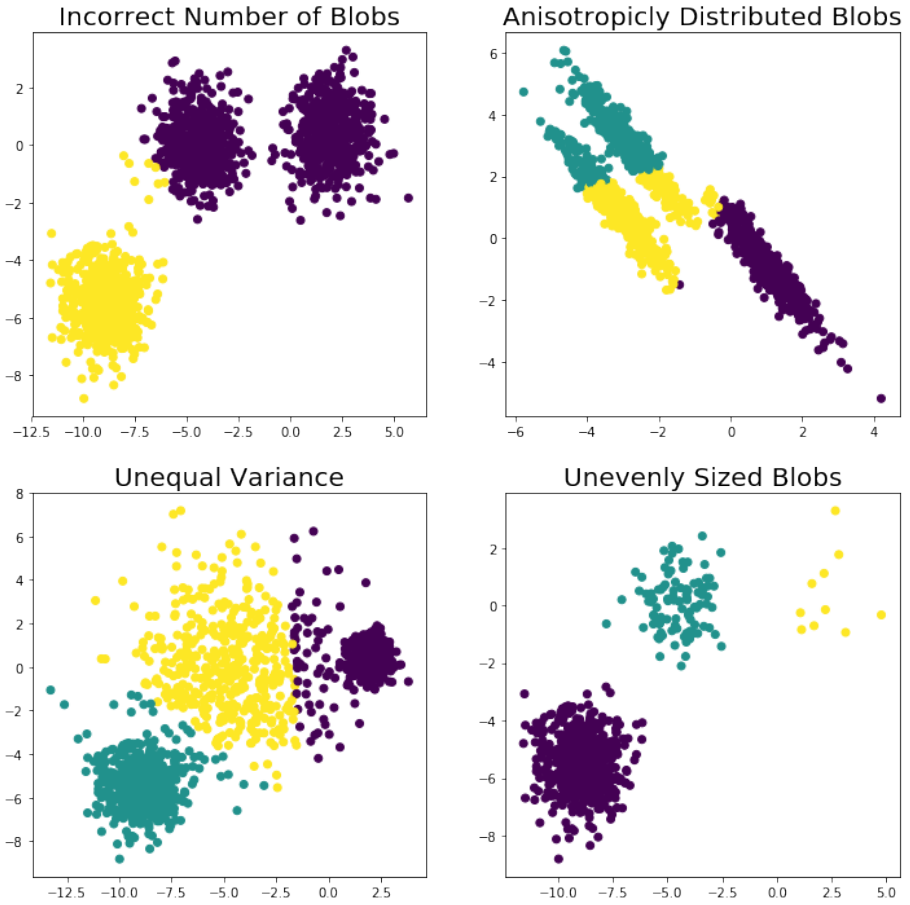
$$argmin \left(\sum_{i=1}^{k} \sum_{j=1}^{n} \| x_j - \mu_i \|^2 \right)$$

K-Means Algorithm

1. Define (**arbitrary**) the number of clusters ***K***
2. Select ***K*** cluster centres randomly
3. Calculate the **distance** between each data point and cluster centres
4. **Assign** the data points to the cluster whose **distance from its centre is minimum**
5. Recalculate all new cluster centres by **by taking the average of all the points assigned to that cluster.**
6. **Repeat** steps 3 to 5 until the **centroids do not change** or reaches a stop criterion

K-Means Assumptions

- The number of clusters ***K* is correct**
- The distribution of the data is **isotropic** (circular/spherical distribution)
- The **variance** is the same for each variable
- The clusters have **roughly** the **same number of objects**



Output from adapted Scikit-learn code

K-Means Pros

- **Fast, robust** and **easier** to understand
- Relatively efficient: O(dknt), where:
  - **d**: number of dimension of each object
  - **k**: number of clusters
  - **n**: number of objects
  - **t**: number of iterations
  - Usually **k, t, d << n**
- Gives better results when a dataset is distinct or well separated from each other

K-Means Cons

- Requires the specification of the **number of cluster centres**
- Exclusive Assignment
  - **Will not be able to split data** if there are overlapping data
- Not invariant to non-linear transformations
  - A **different representation** of data gets **different results**
    - I.E. Cartesian coordinates and Polar coordinates will give different results

K-Means Cons

- **Euclidean distance** might weigh **underlying factors** incorrectly
- Provides the **local optima** of the squared error function which might be different from the **global optima**
- Randomly choosing of the cluster centre **does not guarantee** a fruitful result
- Requires the mean (I.E. **works for numerical data only**)
- Unable to handle **noisy data and outliers**
- Algorithm fails for **non-linear** datasets

K-Means Demo

- Visualising K-Means Clustering
  - Website: Naftali Harris
  - Select method to define initial centroids
  - Select dataset

Lab 6.1: K-Means

- Purpose
  - **Understand** the concept and algorithm of K-Means
  - **Practice** some coding in Python

- Resources
  - Sample data from SciKit-Learn

- Materials
  - Jupyter Notebook (Lab-6_1)

Categorical Variables

- The K-Means algorithm **cannot cluster categorical variables** as there is no measure of proximity
  - The sample space for categorical data is discrete and does not have a natural origin
  - **A Euclidean distance** function on such a space is not meaningful

Ordinal Variables

- Ordinal variables
  - They have **inherent order**, for example, low/ medium/ high (low < medium < high) or school/ college/ university (school < college < university).
- They can be replaced with arithmetic sequence of appropriate values within the **appropriate metric for your data**.
- Requires good understanding of the **domain and its data.**

Categorical Variables – K-mode

- K-mode algorithm uses, instead of distance, dissimilarities (that is, **quantification of the total mismatches between two objects**: the smaller this number, the more similar the two objects).

- A mode is a vector of elements that minimises the dissimilarities between the vector itself and each object of the data.

- We will have as **many modes** as the **number of clusters** we required, since they act as centroids.

Scikit-Learn K-Means - hyperparameters

- **n_clusters**
  - The number of clusters to form as well as the number of centroids to generate.
  - Use domain knowledge, experiment with different values or optimise using elbow curve
- **init**
  - Method for initialization of centroids, defaults to ‘k-means++’
- **n_init**
  - Number of time the k-means algorithm will be run with different centroid seeds.
- **max_iter**
  - Maximum number of iterations of the k-means algorithm for a single run.
- **Tol**
  - Relative tolerance with regards to inertia to declare convergence

Lab 6.1.1: K-Means in sk-learn

- Purpose
  - Applying sklearn k-means model
  -
- Resources
  - Australian athletes data set
- Materials
  - Jupyter Notebook (Lab-6_1_1)

Lab 6.2: K-Means Bad Cases

- Purpose
  - Understand the limitations of K-Means
  - Practice some coding in Python

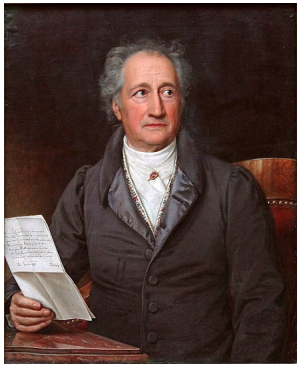
- Resources
  - Sample data from SciKit-Learn

- Materials
  - Jupyter Notebook (Lab-6_2)

K-Nearest Neighbours

- Overview
- Algorithm
- Assumptions
- Pros and Cons
- K-NN in Python
- K-NN Improvements`

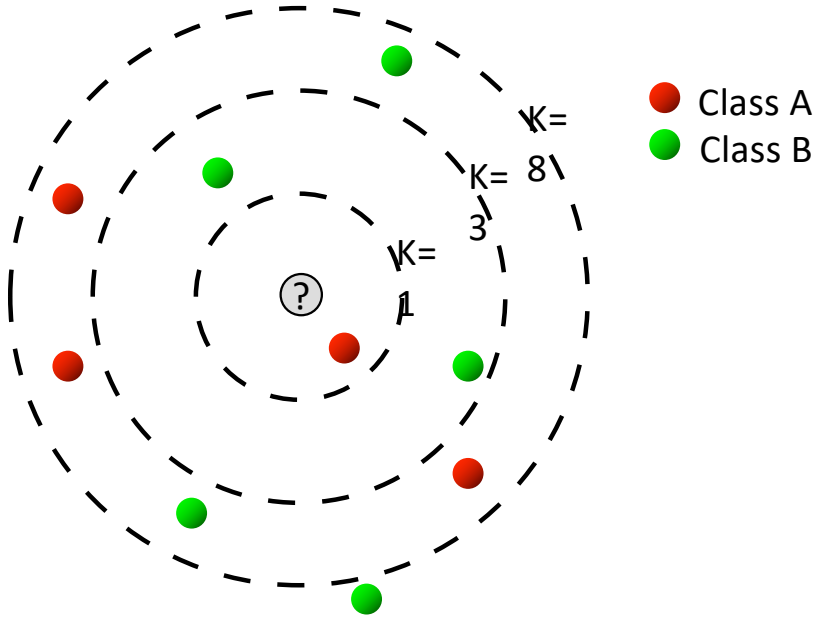
“Tell me with whom you consort and I will tell you who you are”



Johann Wolfgang von Goethe,
(1749-1832)

K-NN Overview

- Based on **feature similarity** or how closely out-of-sample features resemble the training set determines how to classify a given data point



K-NN Overview

- A majority vote of its neighbours **classifies** an object
  - The object is assigned to the class most common among its k nearest neighbours
  - Instance-based algorithm
- Method typically used in **Classification** problems but also applicable for Regression
- **Lazy learning**, meaning that there is **no explicit training** phase before classification
- A powerful classification algorithm used in pattern recognition.

K-NN Overview

- **Learning Algorithm**
  - Store training examples
- **Prediction Algorithm**
  - To classify a new example x by finding the training examples (x_i , y_i) that is nearest to x
  - Guess the class y = majority of nearest y_i
  - To classify a new input vector x, examine the k closest training data points to x and assign the object to the most frequently occurring class
- **Common values for k is 3 or 5** but the right value would **depends on the domain and data**.

K-NN advantages and disadvantages

- **Advantages**
  - **Training is very fast!**
  - Easy to program
  - No optimisation or training required!
  - Classifications accuracy can be very good
  - Can outperform more complex models
- **Disadvantages**
  - **Slow at query time!**
    - Must make a pass through the data for each classification. This can be prohibitive for large data sets.
  - Easily fooled by irrelevant attributes
  - Nearest neighbour breaks down in **high dimensional spaces** because the "neighbourhood" becomes **very large!**.

K-NN Common distance metrics

- **Euclidean Distance** calculates magnitude

$$E(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Cosine Similarity** uses the difference in direction between two vectors

$$Similarity = cos(\theta) = \frac{A \cdot B}{\| A \| \cdot \| B \|}$$

K-NN Algorithm

1. Load the training and test data
2. Choose the value of ***K***
3. For each point in the **test** data
    1. Find the **Distance** to all training data points
    2. Store the distances in a list and sort it
    3. Choose the first ***K*** points
    4. Assign a result to the **test** point based on the
        - **Majority** of classes present in the chosen points, if **Classification**
        - **Mean** or **Median** of present values in the chosen points, if **Regression**

K-NN Assumptions

- Each of the training data has a set of vectors and **category label** associated with each one
  - **Binary**: It is either + or – (for positive or negative classes, in the simplest case)
  - **Multi-class**: Can work equally well with an arbitrary number of classes

K-NN Assumptions

- A single number ***k*** is given
  - This number decides **how many neighbours** influence the classification
    - Where neighbours are defined based on the distance metric
  - K is usually **odd** to prevent **tie situations**
  - The algorithm is called nearest neighbour if ***k=1***

K-NN Pros

- **No assumptions about the data**
  - Useful, for example, for nonlinear data
- **Simple algorithm**
  - Easy to explain and understand/interpret
- High accuracy (**relatively**)
  - Accuracy is pretty high but not as much as better supervised learning models
- **Versatile**
  - Useful for classification or regression

K-NN Cons

- **Keeps** (almost) all of the training data
- **High memory** requirement
- Computationally **expensive**
  - Because the algorithm stores all the training data
- The **prediction phase can be slow** for a large N
- Can be affected by **irrelevant features** and the volume of the data

K-NN Demo

- Visualising K-Means Clustering
  - Website: Vision Stanford
  - Select the available options
  - Observe the results

K-NN Improvements

- **Weighed voting:** A simple and effective way to remedy skewed class distributions is by implementing weights
  - The class of each of the K neighbours has a weight inversely proportional to the distance between that point to the test point provided
  - This ensures that nearer neighbours contribute more to the final vote than the more distant ones
- **Vary distance metrics:** Changing the distance metric for different applications may help improve the accuracy of the algorithm
  - I.E. Hamming distance for text classification

K-NN Improvements

- **Normalise the data**: The distance metric seems more coherent with rescaling

  - For instance, given two features pH and temperature, an observation such as ***x=(3, 25)*** skews the distance metric in favour of temperature
  - Subtracting by the mean and dividing by the standard deviation can address the issue
  - The use of Scikit-learn's normalize() method can be useful

K-NN Improvements

- **Reduce dimensions**: Dimensionality reduction techniques like **PCA** should be executed before applying KNN and help make the distance metric more meaningful

- Approximate Nearest Neighbour techniques such as using **k-d trees** to store the training observations can be leveraged to decrease testing time
  - High dimensions (20+) can affect the perform of such methods

  - **Locality Sensitive Hashing** (LSH) is an alternative for higher dimensions

Scikit-Learn KNN - hyperparameters

- **n_neighbors**
  - int, optional (default = 5)
- **Weights**
  - str or callable, optional (default = ‘uniform’)
  - ‘uniform’ : uniform weights.
  - ‘distance’ : weight points by the inverse of their distance. in this case, closer neighbours of a query point will have a greater influence than neighbours which are further away.
- **Algorithm**
  - {‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}, (default = ‘auto’)
- Algorithm-specific hyperparameters

Lab 6.3: K-NN

- Purpose
  - Understand the concept and algorithm of K-NN
  - Practice some coding in Python

- Resources
  - Sample data from SciKit-Learn

- Materials
  - Jupyter Notebook (Lab-6_3)

Principal Component Analysis (PCA)

- Curse of dimensionality
- What is PCA?
- Variance and covariance
- Eigenvector and eigenvalues
- PCA algorithm

Dimensionality Reduction

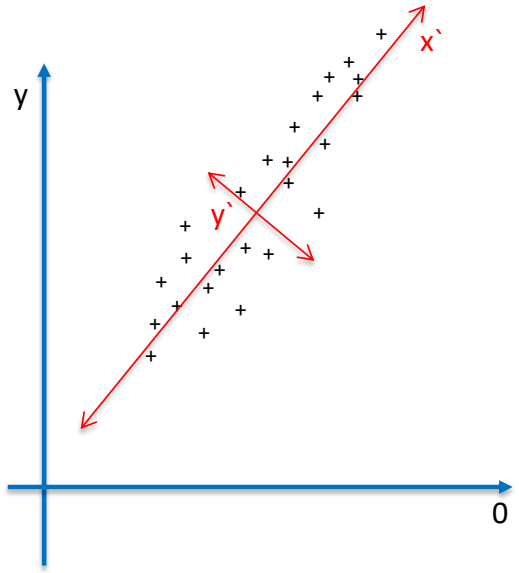
- **Lots of Features = High-Dimensions**
  - Examples:
    - In document classification: features per document = **thousands** of words/unigrams and contextual information.
    - Netflix movie likes: 480,189 users x 17,770 movies
- In most cases, **some dimensions are more important than others!**

The curse of dimensionality

- When the number of features/dimensionality increases, the **volume of the space** increases so fast that the available data become **sparse**.
- To obtain a **statistically significant result**, the amount of data needed often grows exponentially with the dimensionality.
- Observations needed = (sample density in one dimension)^N
  - Where N is the number of features
- Points in **high dimensional spaces** are isolated. The volume of a hypercube with edge, say, d=0.1 is v=0.1^n which is so small to be near any other point in the sample data. This causes e degradation of the performance of algorithms such as K-NN.
- Adding more features can also **increase the noise**, and hence errors.
-

Principal Component Analysis (PCA)

- In case where **data lies on or near a low d-dimensional linear subspace**, there should exist axes that are an effective representation of the data.
- Identifying the axes is done by using classic matrix computation tools (Eigenvectors and eigenvalues).
- PCA can be used for reducing dimensionality by eliminating **less significant** principal components.

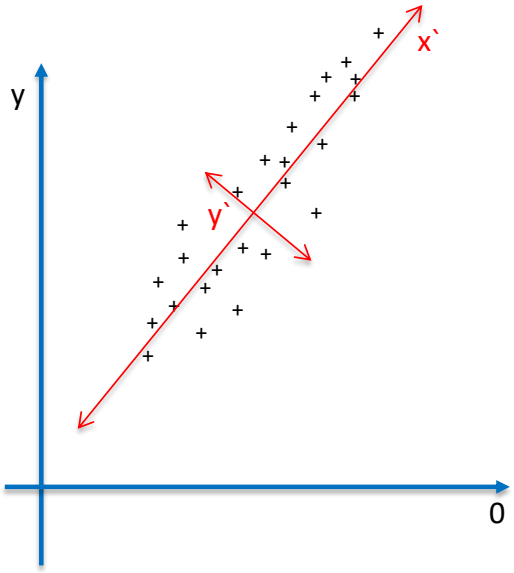


Principal Component Analysis (PCA)

- It is a powerful **unsupervised** learning techniques for extracting hidden (potentially **lower dimensional**) structure from **high dimensional** datasets.
- It is **linear transformation** that chooses a new coordinate system for the data set such that **greatest variance** by an orthogonal projection of the data set comes to lie on the first axis (then called the **first principal component**), the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating **less significant** principal components.
- PCA provides an **approximate representation** of the data.
  - If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard.

Meaning of the Principal Components

- Principle Components (PCs) capture as much **variation** of the data as possible with fewer features.
- PCA detects the **Maximum Variance Direction**
  - The 1st PC is a vector such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections) and the 2nd PC capture the next maximum variation, etc
- PCA minimises the **Minimum Reconstruction Error**
  - The 1st PC a vector such that projection on to this vector yields minimum lose of information when reconstructing the data
- A Principle Component (PC) is a **new synthetic feature**
- The main Principle Components (PCs) **replace the original features**



Variance and Covariance

- Variance and Covariance are a measure of the **"spread"** of a set of points around their centre of mass (mean).
- **Variance**
  - A measure of the deviation from the mean for points in one dimension
- **Covariance**
  - A measure of how much each of the dimensions vary from the mean with respect to each other.
  - A **positive value** of covariance indicates both dimensions increase or decrease together
  - A **negative value** indicates while one increases the other decreases, or vice-versa
  - If covariance is **zero**: the two dimensions are **independent** of each other

Covariance matrix

- Covariance matrix are used to find relationships between dimensions in high dimensional data sets (usually greater than 3) where visualization is difficult
- An **n-by-n matrix** of representing covariance between dimensions of a n-dimension dataset.
- Diagonal is the **variances** of each variable.
- Covariance (x1,x2) = covariance (x2,x1) hence matrix is **symmetrical** about the diagonal.

$$C = \begin{bmatrix} cov(x1,x1) & cov(x1,x2) & cov(x1,x3) \\ cov(x2,x1) & cov(x2,x2) & cov(x2,x3) \\ cov(x3,x1) & cov(x3,x2) & cov(x3,x3) \end{bmatrix}$$

Eigenvectors and eigenvalues

- An eigenvector or 'characteristic vector' of a linear transformation is a vector that **changes by only a scalar factor** when that linear transformation is applied to it.

$$T(\vec{v}) = \lambda \vec{v}$$

- Eigenvectors are useful in PCA because, geometrically, an eigenvector **points in a direction that is stretched by the transformation** and the eigenvalue is the factor by which it is stretched.

PCA, the eigenvector and eigenvalues

- Principal Component Analysis works by:
  - Calculating the **eigenvalues** and **eigenvectors** of the **covariance matrix**
  - Finding that the eigenvectors with the **largest eigenvalues** correspond to the dimensions that have the **strongest correlation** in the dataset.
  - Principle components indicate the direction of maximum variance in the input space which happens to be same as the principal eigenvector of the covariance.

PCA algorithm

- Need to find **eigenvectors** of the **covariance matrix**
- For n original dimensions, covariance matrix is nxn, and has up to **n** eigenvectors. i.e. **n Principle Components**
- **Dimensionality reduction** comes from **ignoring** the components of lesser significance.
- The **Scree Plot** shows the significance of each of the PCs
- You do **lose some information**, but if the eigenvalues are small, you don't lose much.
- Sci-kit-Learn uses a more generalised algorithm (Singular Value Decomposition(SVD)) to compute the Principle Components.

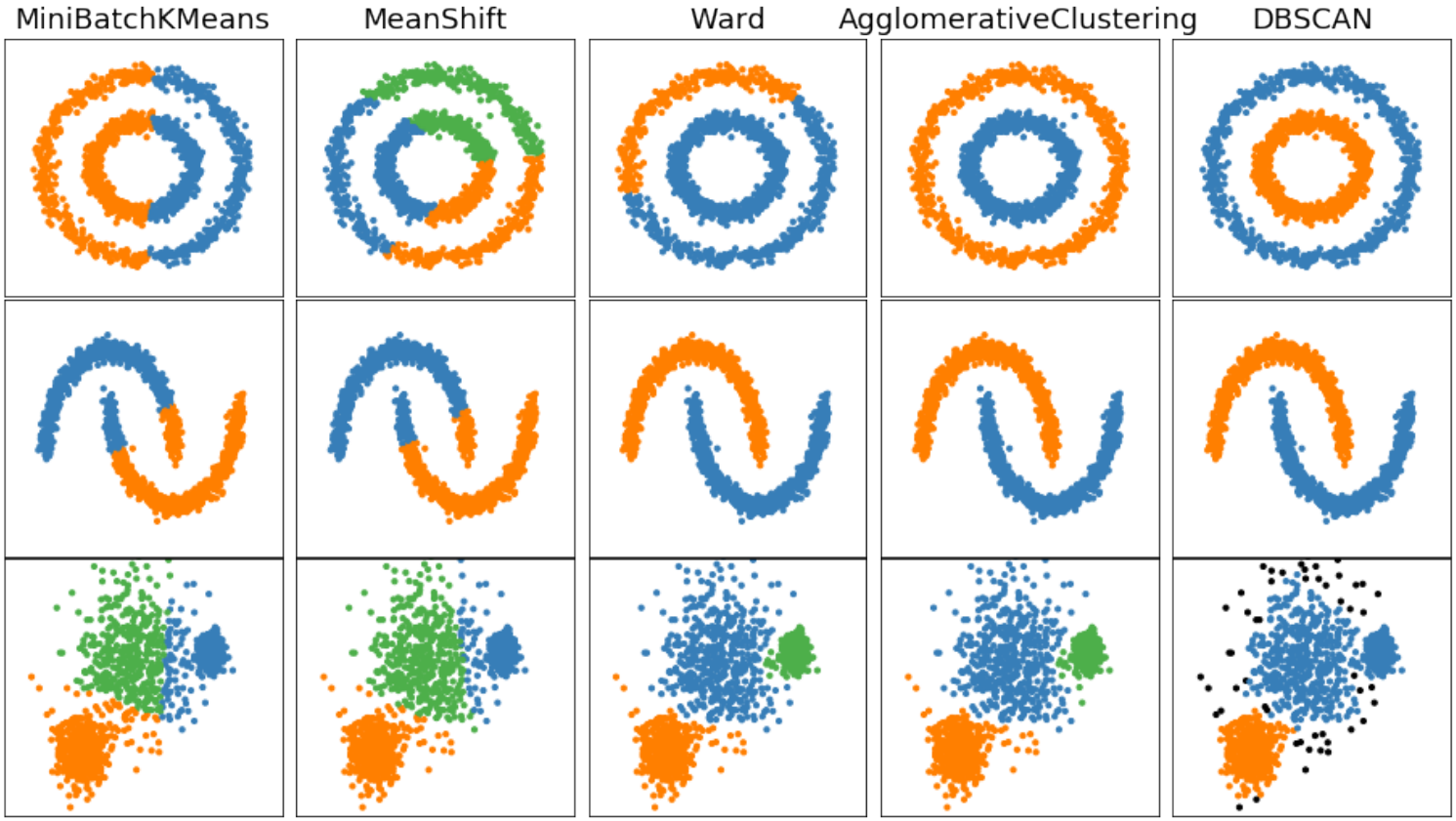
PCA applications

- **Visualisation**
- Feature selection
- **Simplifying** processing by machine learning algorithms
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions a better generalization
- **Noise removal** (improving data quality)
- Compression

Questions?

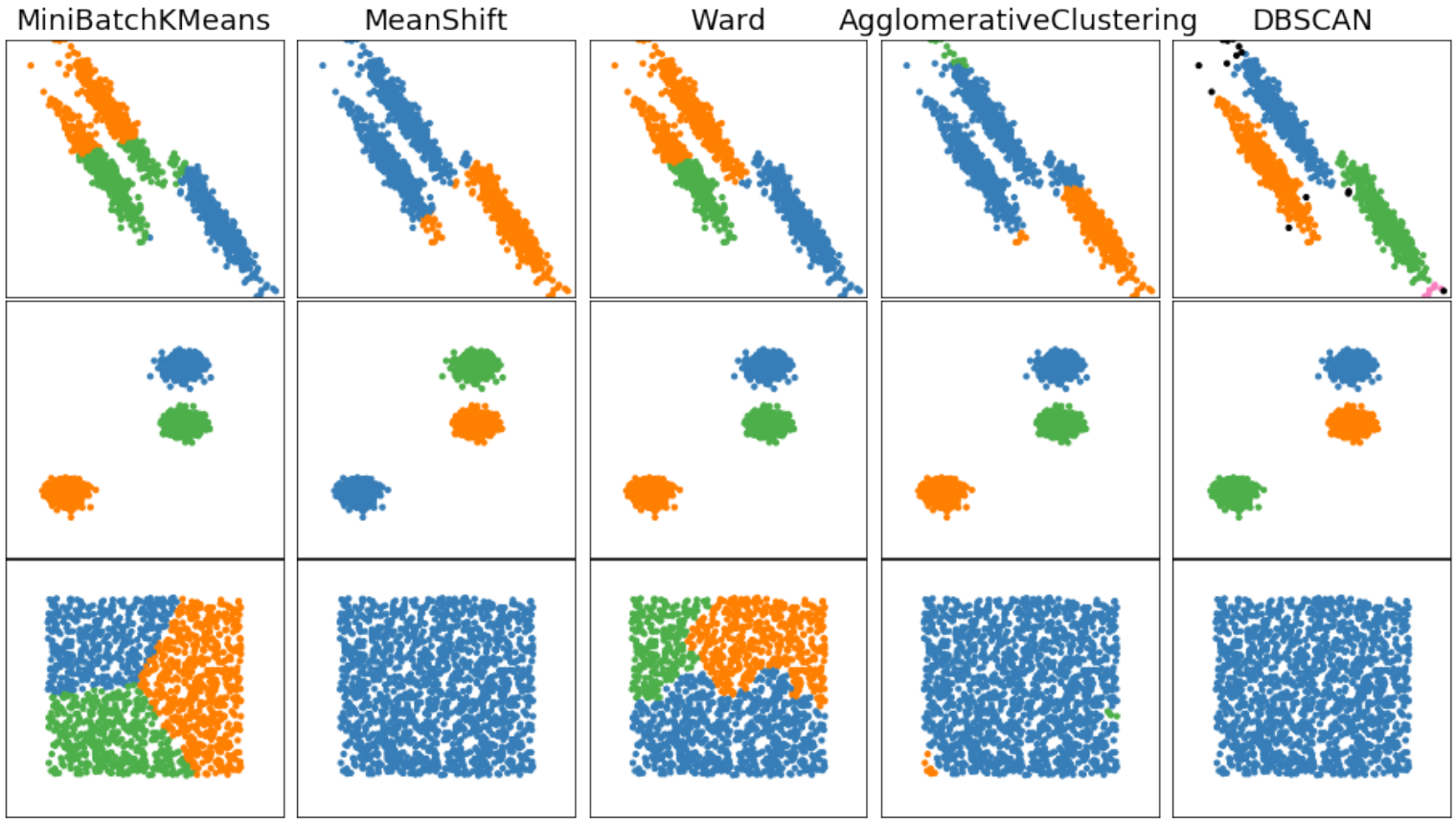
Appendices

Comparison of Some Clustering Methods



Output from adapted Scikit-learn code

Comparison of Some Clustering Methods



Output from adapted Scikit-learn code

End of presentation