Data Analysis and Machine Learning using Python

Lecture 7: Image recognition; dimensionality reduction; unsupervised learning

May 4 2024

Today:

- Homework 8 will be posted tomorrow
- Quick review of homework 6
- Example: Image recognition using MLP classifier
- Dimensionality reduction
- Example: Principal component analysis
- Unsupervised learning
- Example: K-means algorithm

Dimensionality reduction

- We saw before that having a large number of variables in fitting or a large number of features is challenging:
  - Determines dimension D of space in which we have to find the best solution (minimum of χ^2 or minimum of loss function)
  - Danger of finding false minima
  - Number of data points or training samples required grows as power of D
  - “Curse of dimensionality”

Dimensionality reduction

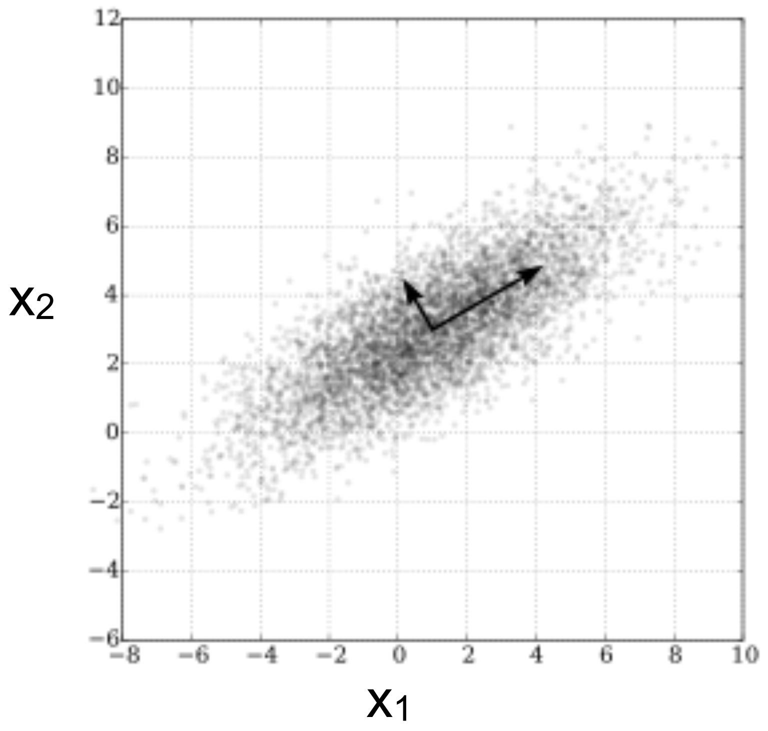
- "Dimensionality reduction":
  - reduce number of features in dataset while using as little relevant information as possible
  - project initial higher dimensional feature space to a lower dimensional one
  - can be a pre-processing step before training the ML model. E.g.,
    - remove features that carry little or no relevant information
    - combine redundant features (that carry similar or correlated information) into single feature

Dimensionality reduction

- Example approaches
  - **Principal component analysis**
    - Data are linearly transformed into a new coordinate system where the directions (“principal components”) capture the largest variations in the data
  - Missing value ratio
    - Eliminate sparse variables that have many missing values (above some threshold)
  - Backward feature elimination
    - Start with all “n” features, train and evaluate models. Eliminate variables one-by-one, retrain and re-check. Remove features from final model that make no or insignificant difference
  - Forward feature selection
    - Start with with one or few features, and add only those that bring the most gain in the models performance
- many other approaches…

Principle component analysis (PCA)

- Invented by Pearson in 1901. Independently reinvented many times and known under many different names (*singular value decomposition, eigenvalue decomposition, factor analysis…*) in different forms/fields

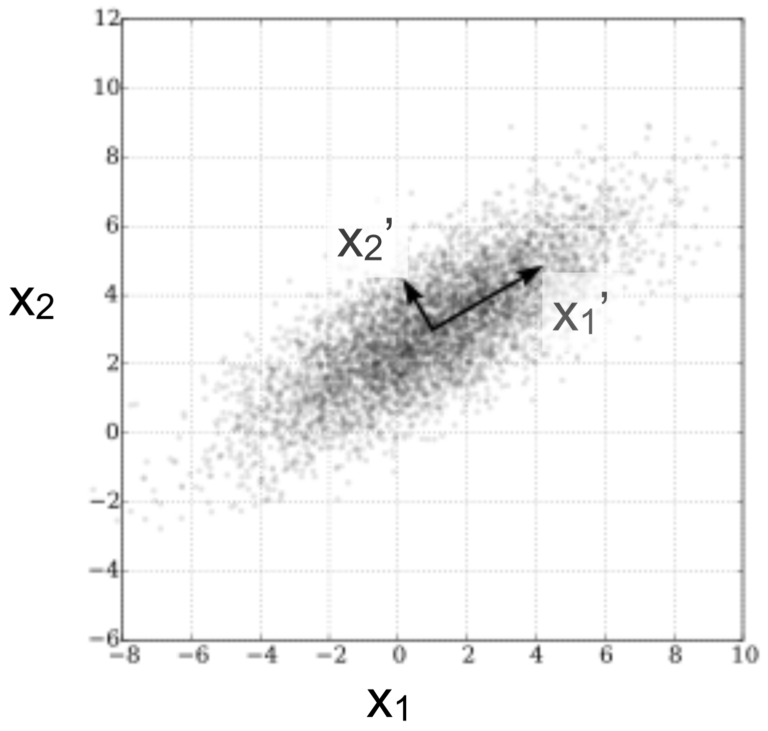


Example:

- two features x1, x2
- each feature is Gaussian distributed
- they are linearly correlated
  - if you know value of x1, you can predict value of x2 with some uncertainty

Principle component analysis (PCA)

- Invented by Pearson in 1901. Independently reinvented many times and known under many different names (*singular value decomposition, eigenvalue decomposition, factor analysis…*) in different forms/fields

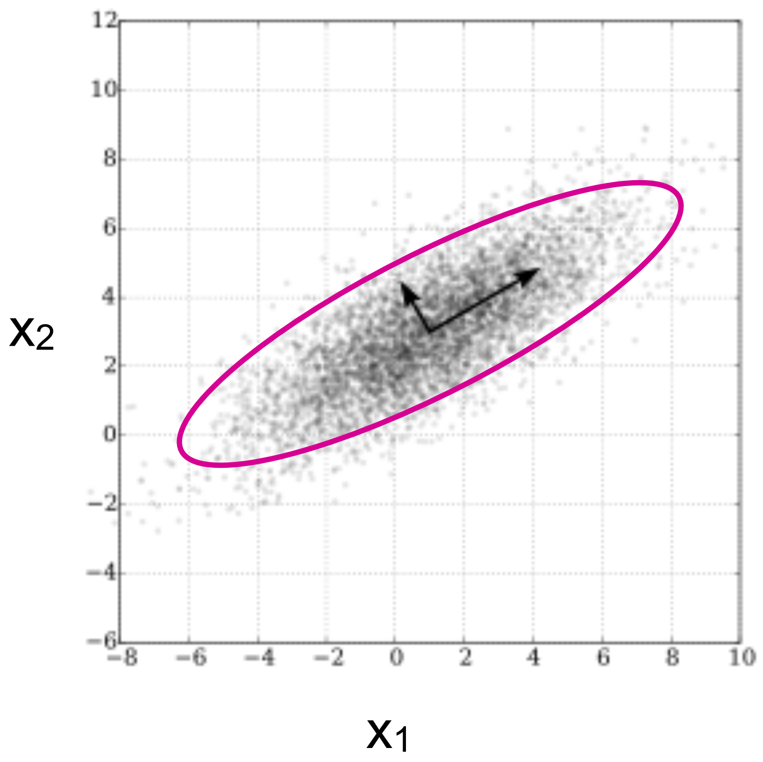


Example:
- two features x1, x2
- each feature is Gaussian distributed
- they are linearly correlated
    - if you know value of x1, you can predict value of x2 with some uncertainty
    - Can go to new coordinate system x_1' and x_2', where all information is in x_1' and x_2' is just noise

Principle component analysis (PCA)

Think of PCA as fitting a n-dimensional ellipsoid. Axis of ellipsoid are the principal components. Short axis = small variance.

1. center each variable on 0
2. compute covariance matrix
3. calculate eigenvalues and eigenvectors of covariance matrix
4. normalize orthogonal eigenvectors



Covariance matrix describes the covariance between each pair of variables
i.e., how much does variable x change of one varies variable y

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

$cov_{x,y}$	= covariance between variable x and y
x_i	= data value of x
y_i	= data value of y
$\bar{x}$	= mean of x
$\bar{y}$	= mean of y
N	= number of data values

If x and y are independent (uncorrelated) the covariance will be zero

Principle component analysis (PCA)

- Let’s look at an example of PCA using scikit-learn….

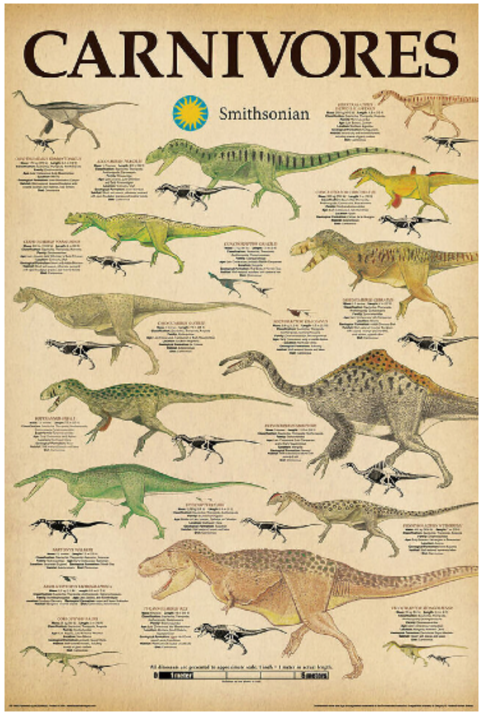
Principle component analysis (PCA)

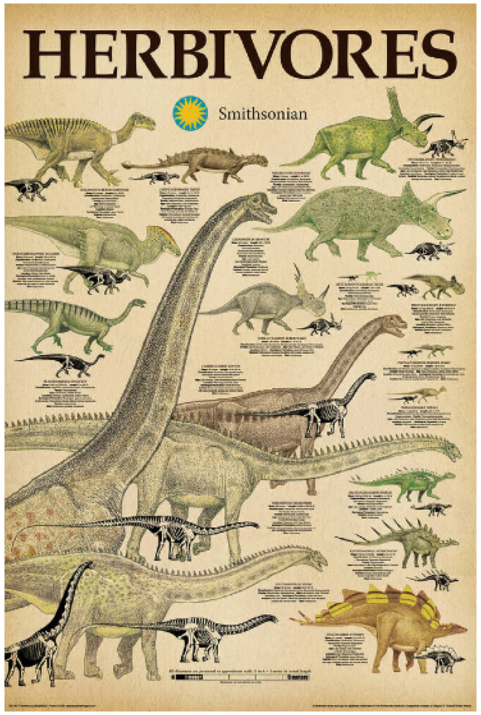
- Note that the PCA was able to extract useful features from the data without being told any target values
- Moving from supervised to **unsupervised learning**

Unsupervised learning

- In **unsupervised machine learning** models are given **unlabeled** data and allowed to discover patterns and structure
- The model tries to find similarities, differences, patterns without human intervention or guidance
- Typical use-cases of unsupervised learning
  - **Clustering**: Find subsets of data in the feature space that "belong" together. E.g. *k-means clustering* algorithm
  - **Anomaly detection**: Find subsets of data that are different from the rest of the data (e.g., in a *time series)*

Unsupervised learning





Pterosaurs
The Flying Reptiles

You may not know what the difference is, but looking at these pictures, there are **clearly distinct classes** of dinosaurs

Unsupervised learning

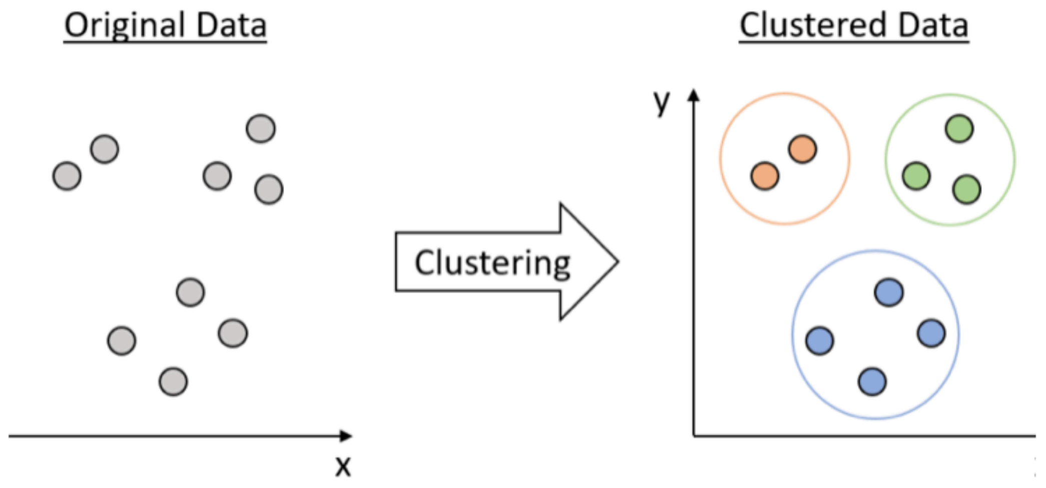
- Challenges for unsupervised learning
  - Generally, more difficult
  - How do we evaluate the results if there is no label with the correct answer?
    - External evaluation: Have a (human) expert look at results
    - Internal evaluation: Define an objective function to study e.g. success of clustering

Unsupervised learning

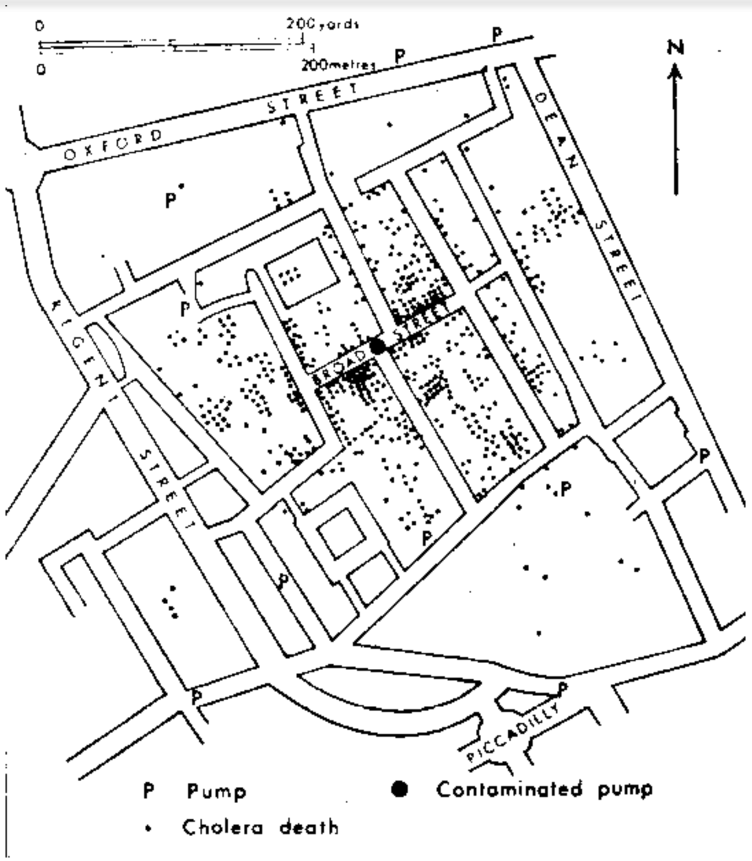
- Why is unsupervised learning useful?
  - Labeling large datasets can be very costly/labor intensive
  - we may not know what the classes in the data actually are (knowledge discovery and data mining)
  - learn about structure in the data to help design classifiers

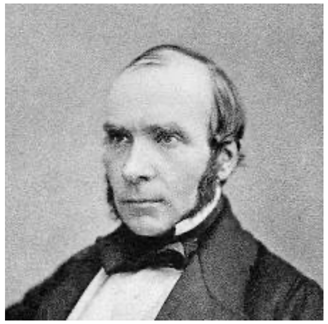
Clustering

- Fundamental unsupervised learning technique
- Find groups of objects that are "similar" as clusters in feature space, distinct from "less similar" objects in other clusters



Example: John Snow’s cholera map





John Snow used map to find cluster of cholera deaths in London in 1850’s

Found cluster around a certain water pump, support hypothesis of waterborne transmission (as opposed to commonly assumed airborne transmission)

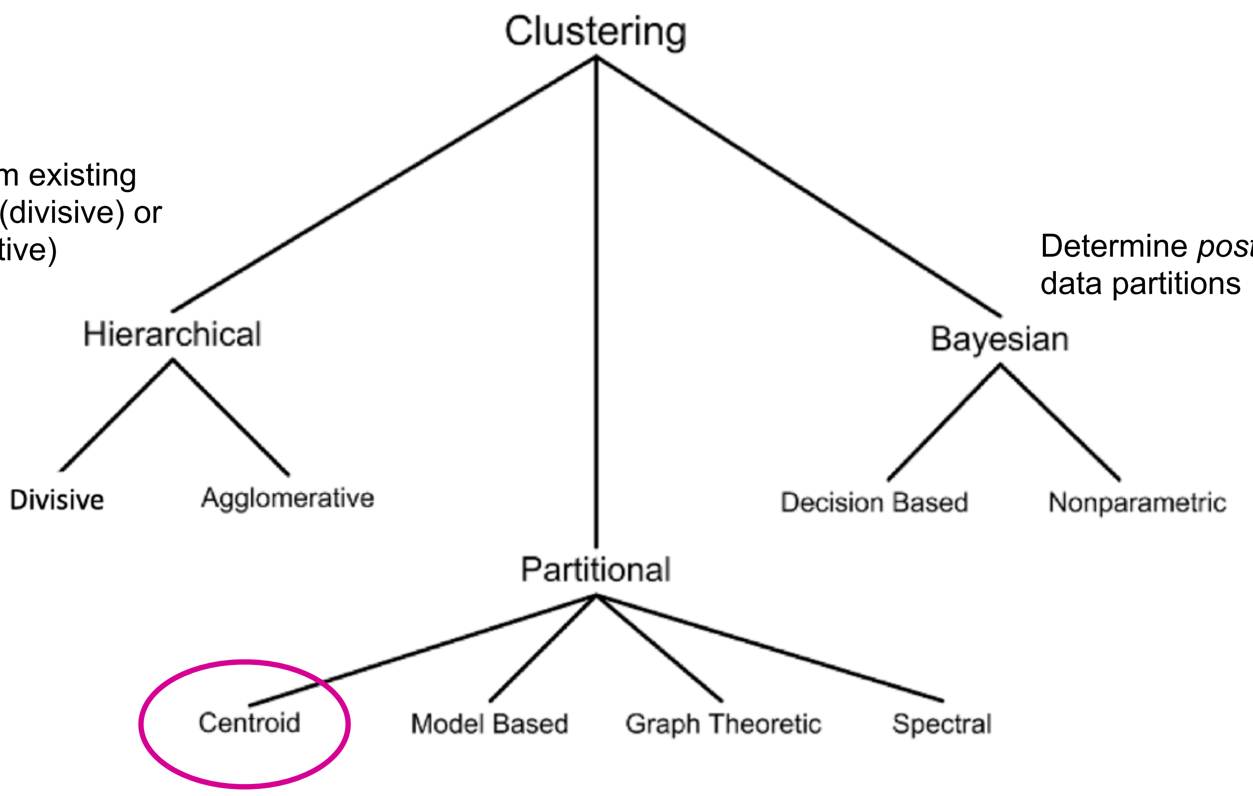
Ingredients for clustering

- A **measure** of similarity (or proximity in feature space)
- A **criterion function** to evaluate clustering
- An **algorithm** to perform clustering and optimize by finding maximum (or minimum) of criterion function
- There are many different clustering algorithms…

Types of clustering algorithms

Make new clusters from existing ones, either top-down (divisive) or bottom-up (agglomerative)

Determine *posteriori* distribution of data partitions



Determine all clusters simultaneously

k-means clustering is a classic unsupervised learning technique and widely used partitional clustering algorithm

k-means clustering

- Algorithm:

1. Pick k different random seed points in feature space as initial cluster centers (centroids)
2. Assign each sample to the closest centroid using distance measure
3. Calculate new centroid position using samples belonging to cluster
4. Repeat steps 2 and 3 until some convergence criterion is met

Stopping criterion

- Example stopping criteria:
  - Number of re-assigned samples is 0 or below threshold
  - Change in centroid position is 0 or below threshold
  - Change in criterion function is 0 or below threshold
    - Example criterion function: sum of squared errors (SSE): $SSE = \sum_j \sum_x d(x, m_j)^2$, where x are the samples in the j-th cluster, m_j is the centroid of the j-th cluster and d is the distance measure

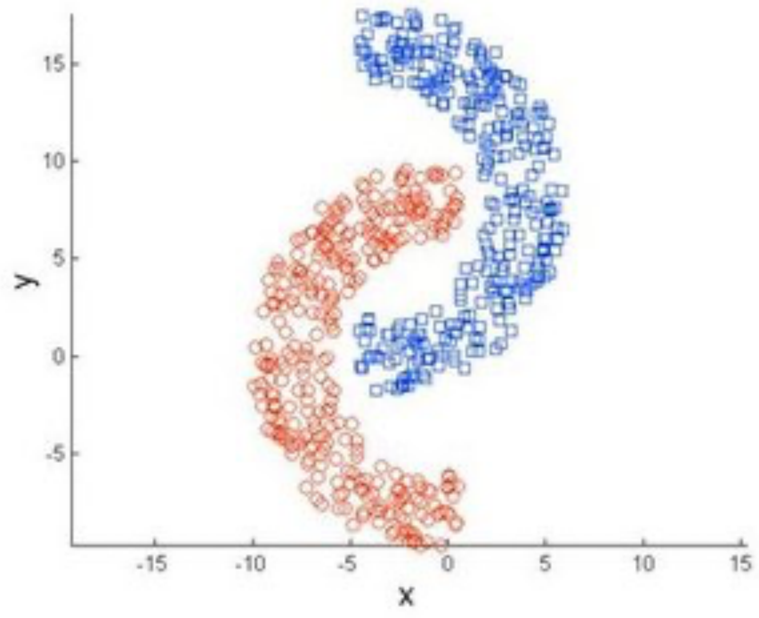
What’s good about k-means?

- Simple algorithm
- Easy to understand
- Easy to implement
- Efficient: time is O(kn) for n datapoints (linear)
- will find a *local* minimum of SSE criterion function
  - (but is it the global minimum?? Hard!)
- → most popular clustering algorithm

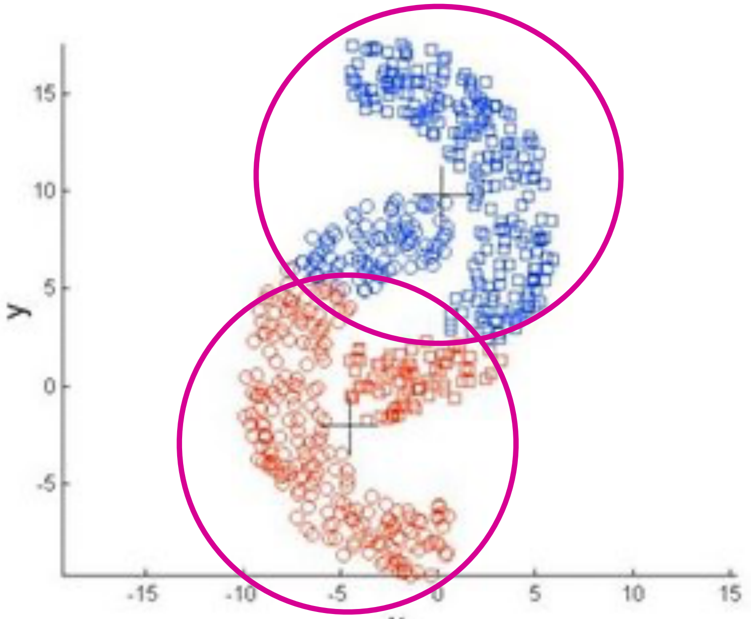
Challenges

- k is specified by the user - what to choose?
- Sensitive to initial seeds
- Sensitive to outliers - large pull on SSE
- Not suitable for clusters that are not hyper-spheres
- Need to be able to define mean - what to do with categorical data?

Hyper-spheres



Original Points



K-means (2 Clusters)

Even without colors, a human would pick two crescent shaped clusters

k-means only does (hyper-) spheres

k-means clustering

- Let’s look at k-means clustering in sklearn
- The algorithm tries to pick cluster centroids that minimize the **inertia** or within-cluster sum of squares
  - $\sum_{i=0}^{n} min(|x_i - \mu_j|)^2$ for samples x_i and centroids μ_j

Anomaly detection

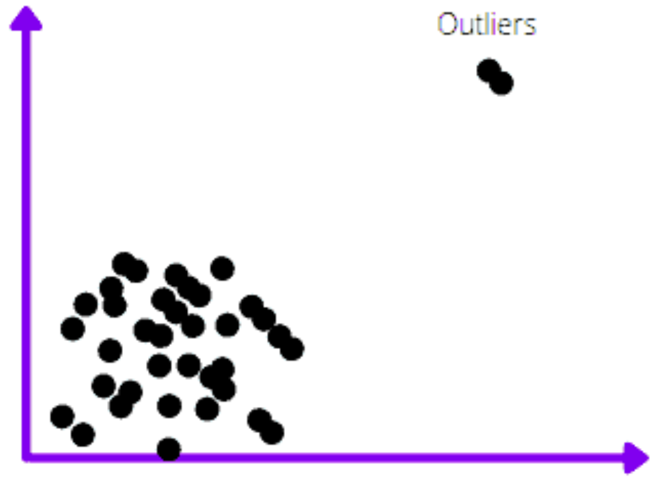
- Sometimes referred to as outlier detector or novelty detection
- Identify rare objects, events or observations that deviate from the majority of the data
- Can be created by a different mechanism or process, but need to be aware of random (statistical) fluctuations
- A common problem in many domains, from discoveries in science to medicine, financial markets etc

Anomaly detection

- Three basic categories:
  - Supervised: just like for other classification problems, have labeled training set of normal and anomalous objects and train classifier. Then apply classification on unknown data
  - Semi-supervised: some small fraction of labeled data, often only “normal” data
  - Unsupervised: Only unlabeled data. This is the most common application of automated anomaly detection

Anomaly detection

- Many methods have been proposed.
- Obviously, clustering can plat a role here:

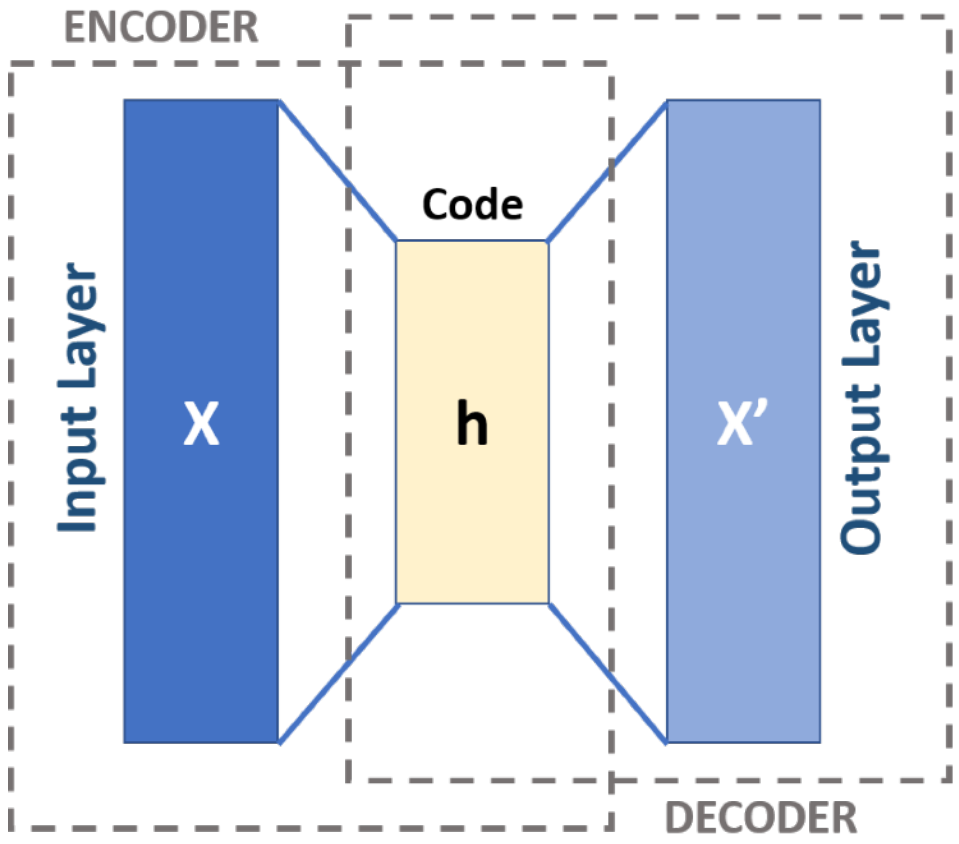


- As always, this will be much more difficult in a high-dimensional space

Anomaly detection with *Autoencoders*

- Autoencoders are a class of neural networks that can be employed for anomaly detection
- They combine many of the concepts we have discussed:
  - deep neural networks
  - dimensionality reduction
  - unsupervised learning
  - anomaly detection

What is an autoencoder?



- Autoencoder transfers input X to output X’
- Two neural networks: Encoder and Decoder
- Key is the intermediate Code layer, which generally has lower dimension than the input layer
- This forces the Encode to learn how to reduce dimensionality of the input data
- Train autoencoder to make output X’ as close to X as possible

Anomaly detection with *Autoencoders*

- How can we use this for anomaly detection?
- If the anomalous event is different from the normal events that the encoder knows how to compress
- → compression (dimensionality reconstruction) will not work as well
- → decoded features X’ will be more dissimilar from X than for normal events