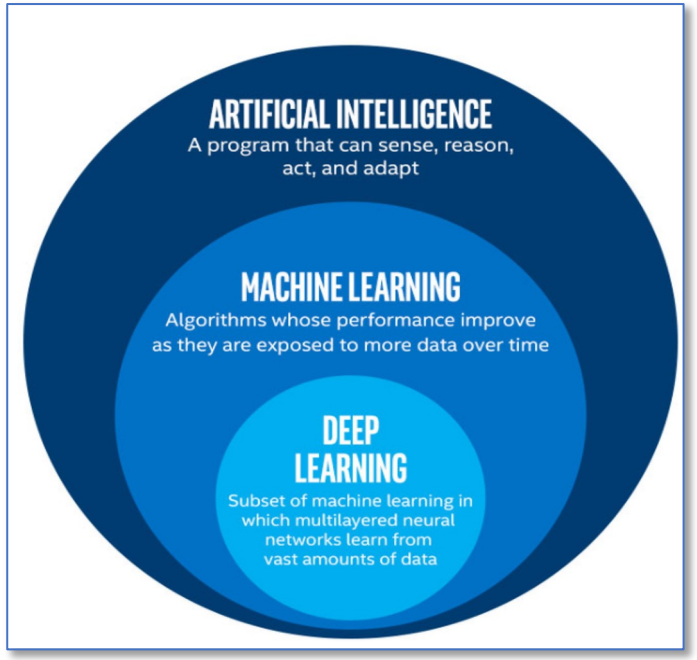
Session 4: Machine Learning Basics

4/12/2023

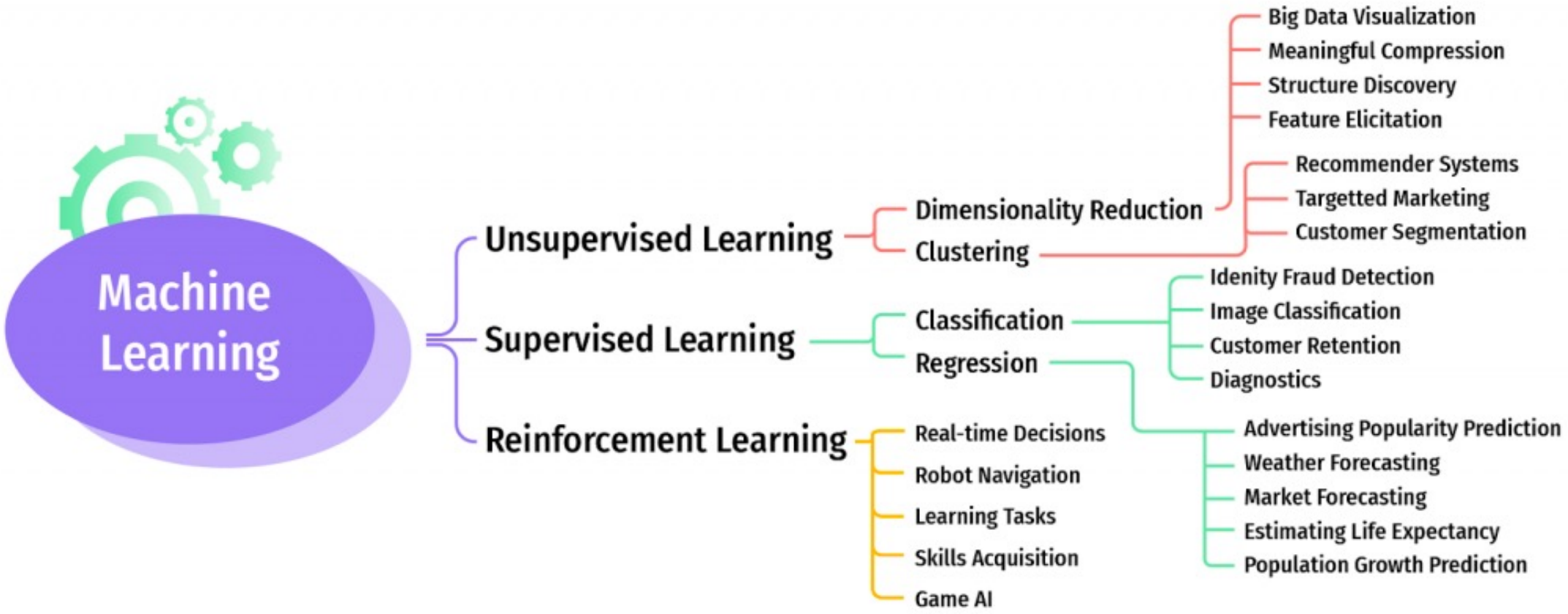
NCME 2023 Data Science Training Workshop
Instructors: Oren Livne & Jiangang Hao

Machine Learning

A term coined by Auther Samuel in 1959, refers to a set of methodologies that allow computers to “learn” the relationship among numerical representations of data without explicit instructions by human experts

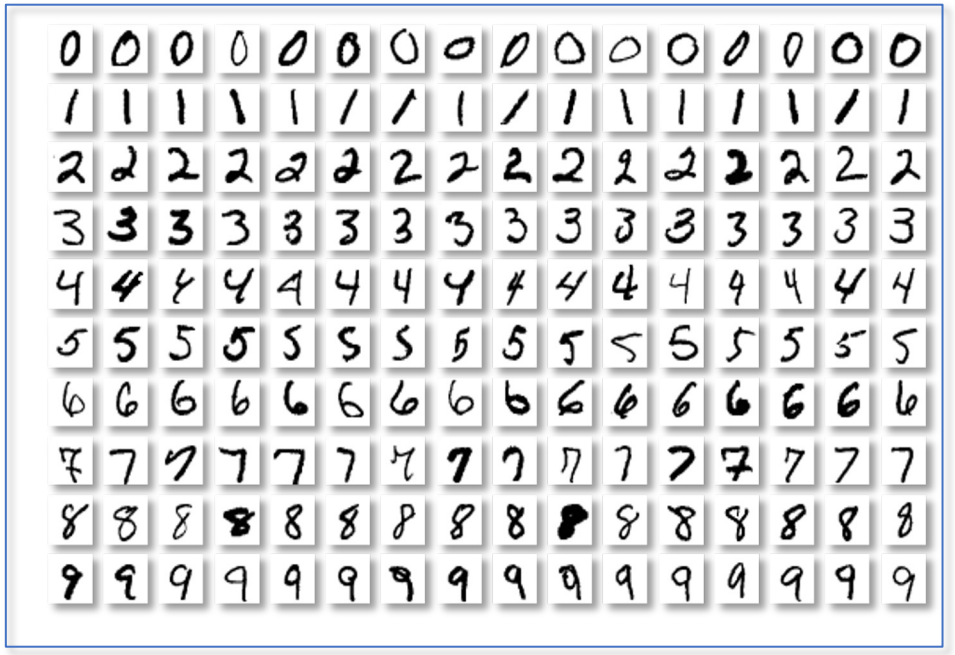


Main Paradigms of Machine Learning

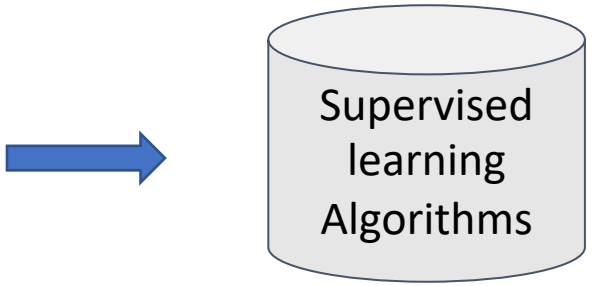


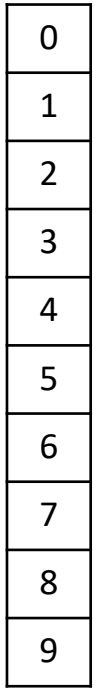
Supervised learning

Machine learns the relationship (mapping) between input variables (features) and output variables (labels)



data



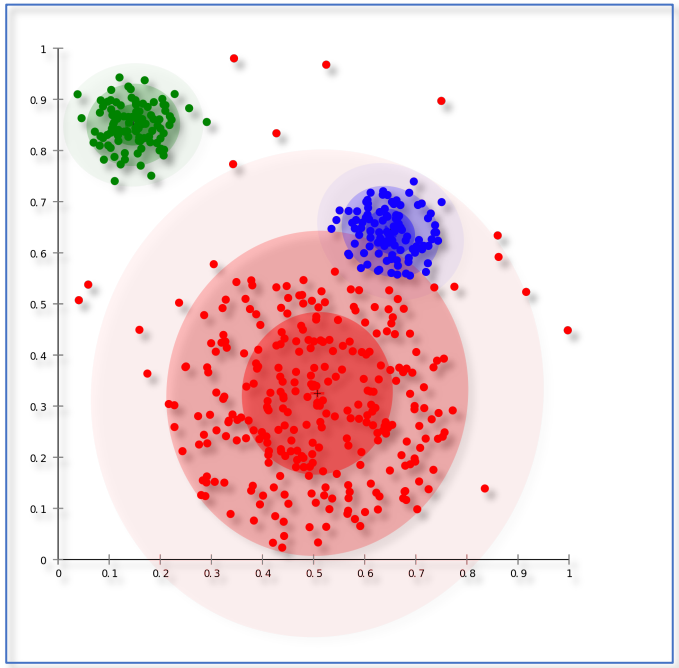


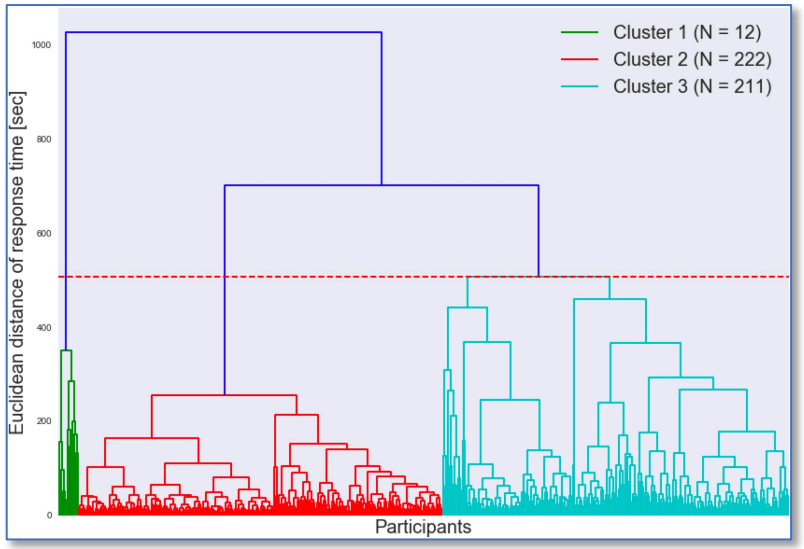
labels

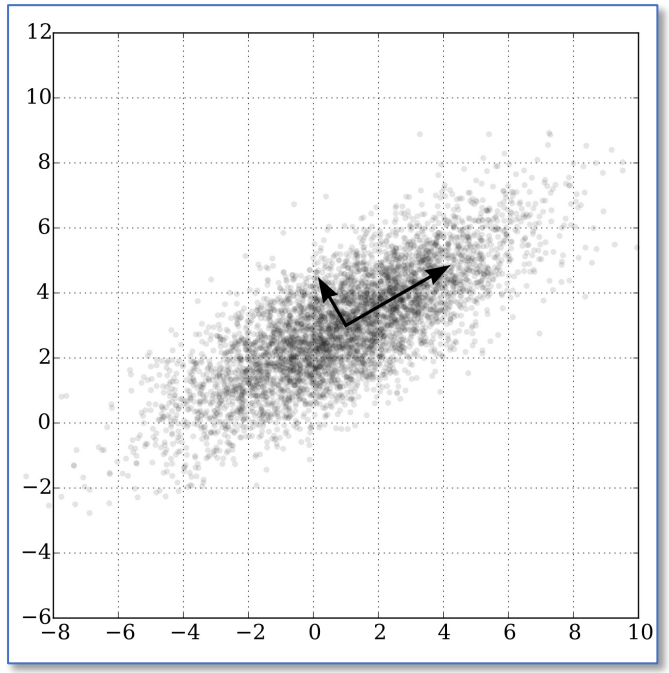
Unsupervised learning

Methodology that discovers concentrations, associations, or correlations in data.

- Cluster analysis
- Dimensionality reduction

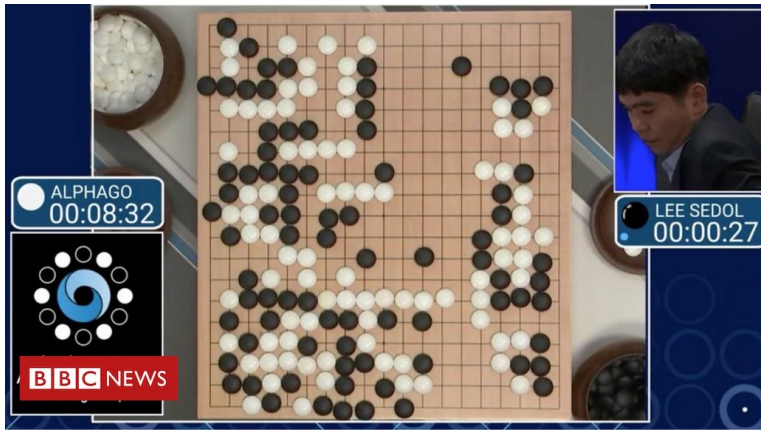


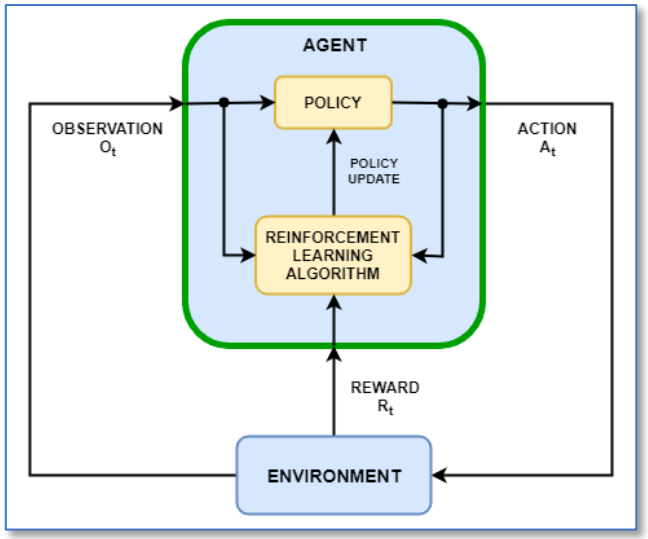




Reinforcement Learning

- Machine learns optimal strategies by means of a goal-oriented exploration of a parameter or state-space to optimize a reward function (Sutton, & Barto, 1998).
- Key components
  - Agent
  - States
    - Hidden
    - Observable (observation)
  - Actions: what the agent can do
  - Policy/strategy: tell what the agent should do
  - Reward: what the agent gets after an action

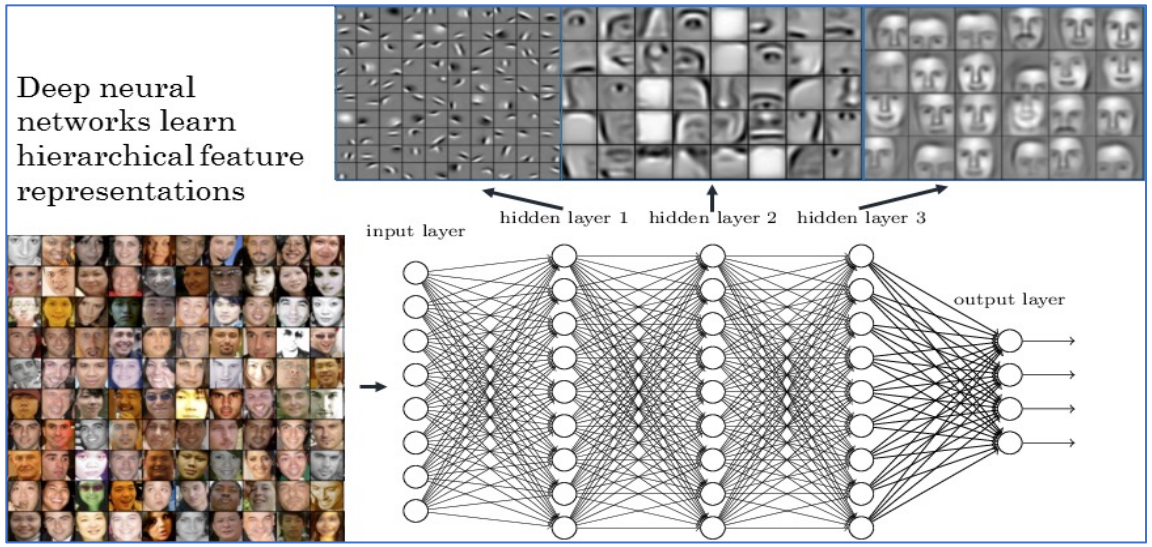




From MathWorks

Representation Learning

- In supervised learning, feature representation is a crucial step
- Features design
  - By human experts – time consuming and expensive
  - By algorithms - representation learning
- End-to-end learning



https://www.strong.io/blog/deep-neural-networks-go-to-the-movies

Supervised Learning

Question 1

- How supervised learning is different from statistical inference, as they both aim at mapping the relationship between the independent and dependent variables?

I. Machine learning is more prediction-driven while statistical inference emphasizes both prediction and model parameter estimation.

II. Statistical inference emphasizes developing a probabilistic model to characterize the data and then estimating the model parameters based on some (usually well-studied) probability distribution functions, while machine learning emphasizes computation algorithms that can efficiently carry out the inference process by minimizing certain loss functions that do not necessarily have a probabilistic underpinning.

III. Statistical inference usually deals with data with a small number of variables obtained through planned *experiments* or quasi-experimental comparisons while machine learning handles sparse and high-dimensional data with a large number of variables (features), usually obtained through passive and uncontrolled *observations* (RWE).

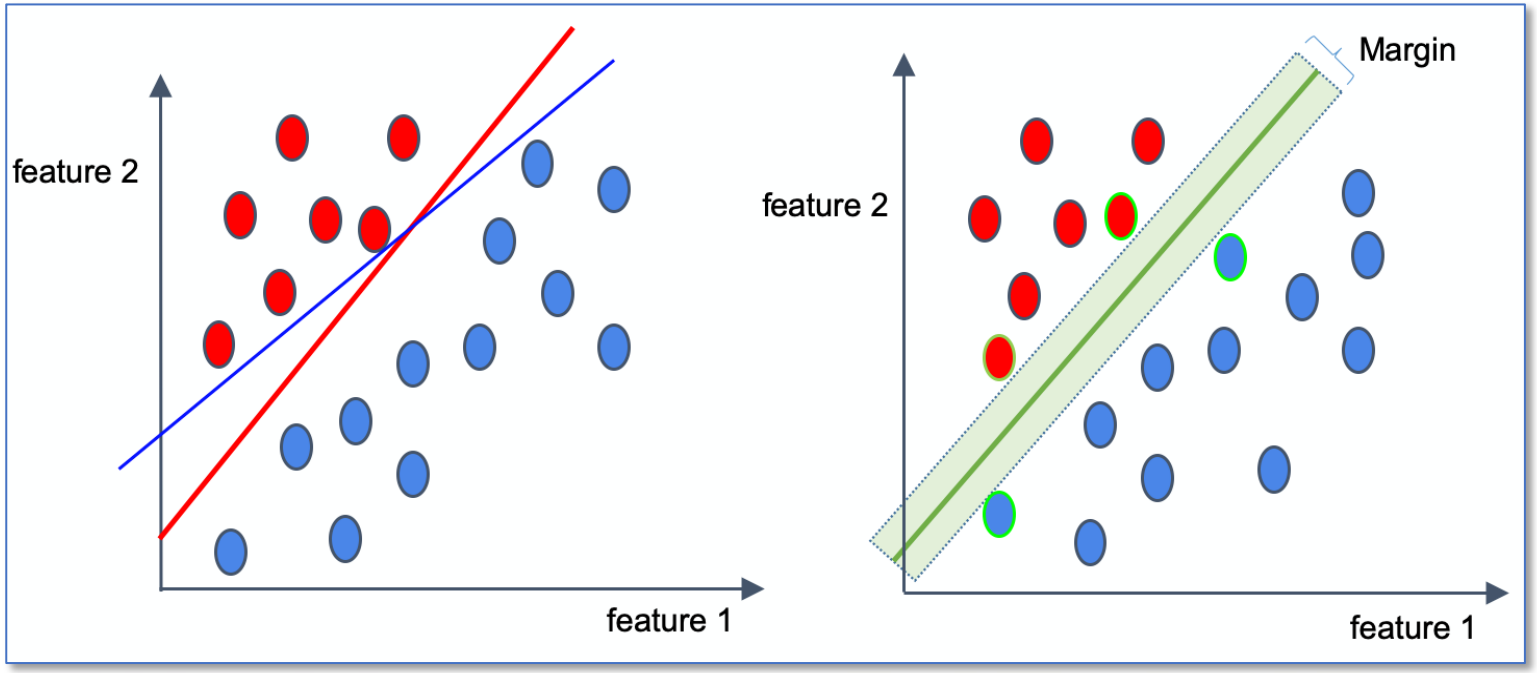
Hao & Ho, 2019; Hao, 2021

Question 2

- Which, if any, supervised learning methods are consistently superior to the others?
  - It is generally data and problem dependent
  - Some methods work well based on many explorations
    - SVM, Random Forest, XGBoost (Gradient Boosting Machine)
    - Deep Neural Network

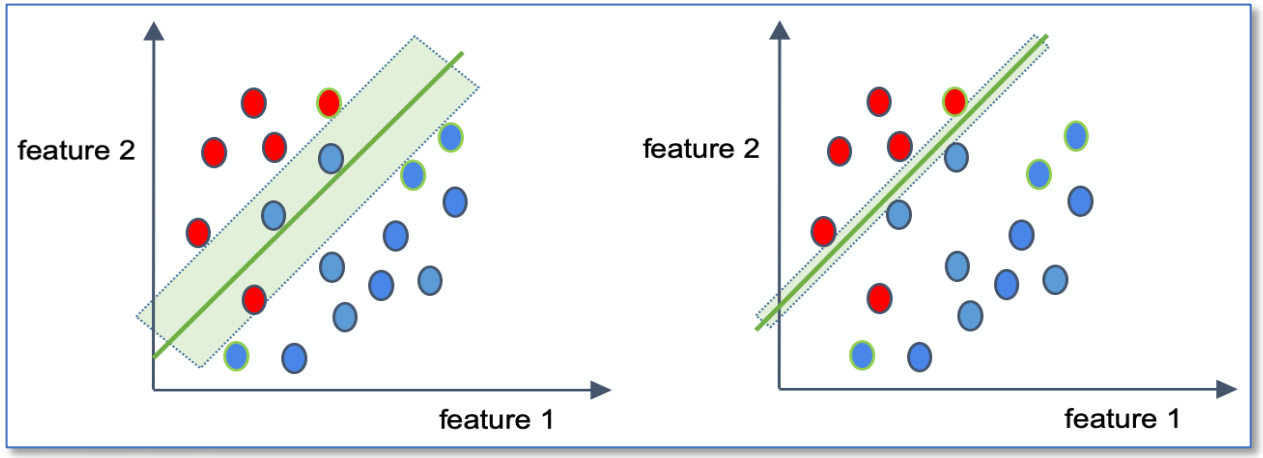
Support Vector Machine

It aims at finding a decision hyperplane in the feature space so that data points can be separated into different categories with a maximum margin that is defined as the distance of the closest points (support vectors) from each category to the decision hyperplane (Vapnik, 1963).



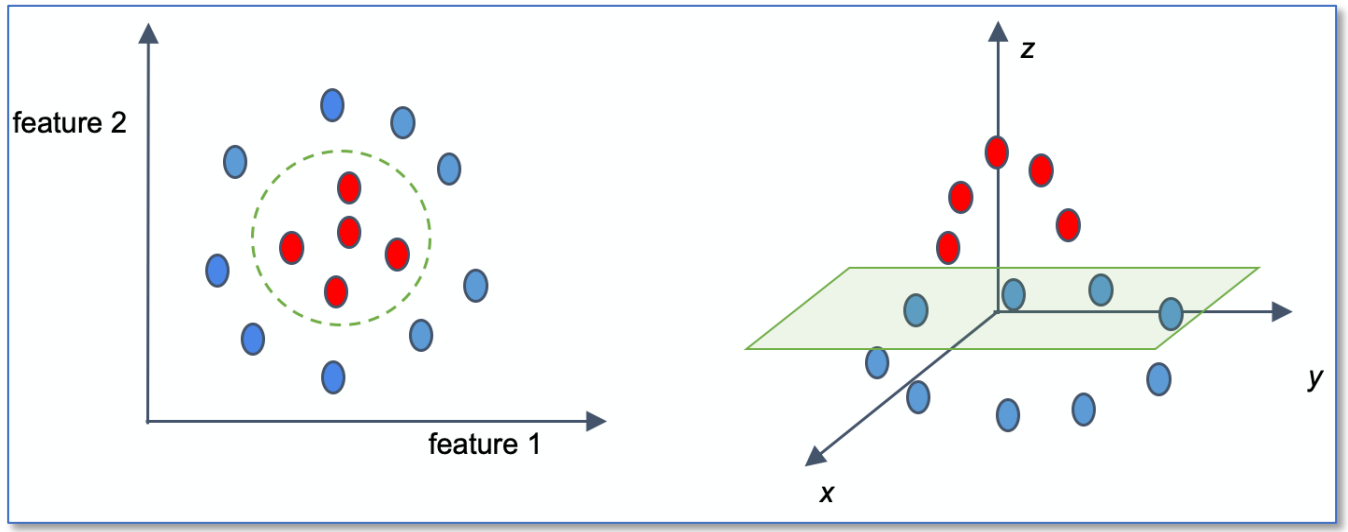
Non-linearly Separable Case

- Choose between a decision hyperplane that has a larger margin but more misclassified data points, and the one with a narrower margin but fewer misclassified data points, as illustrated in the left and right panels
- A hyperplane with a narrower margin may lead to better classification accuracy based on the training data, but is more susceptible to overfitting, while a hyperplane with a larger margin has lower classification accuracy with respect to the training data, but it is more robust against overfitting.
- In SVM algorithm, the regularization hyper-parameter (*C*), is used to control this balance. Choosing a larger *C* indicates that one favors a narrower margin with a lower tolerance of misclassified data points



Kernel Trick

- The transformation of the original feature space into a higher dimensional space to make the data linearly separable is done through a procedure called kernel trick.
- The characteristic of data being more likely to be linearly separable when being projected into a higher dimensional space via some non-linear transformation is known as Cover's theorem (Cover, 1965).
- In SVM algorithms, one can choose the kernel type – another hyperparameter. Typical kernels are polynomial and RBF.



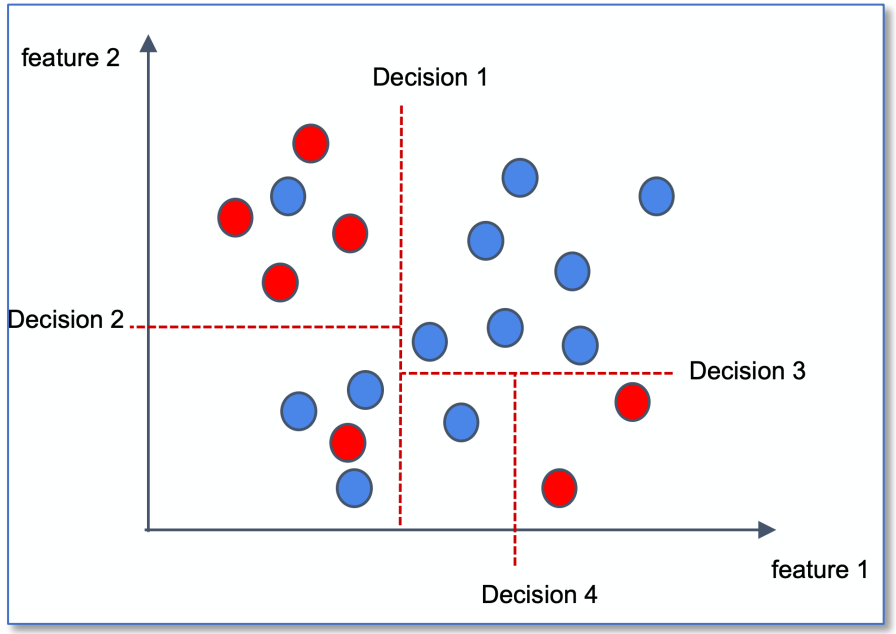
$$\begin{cases} x = feature\ 1 \\ y = feature\ 2 \\ z = (feature\ 1)^2 + (feature\ 2)^2 \end{cases}$$

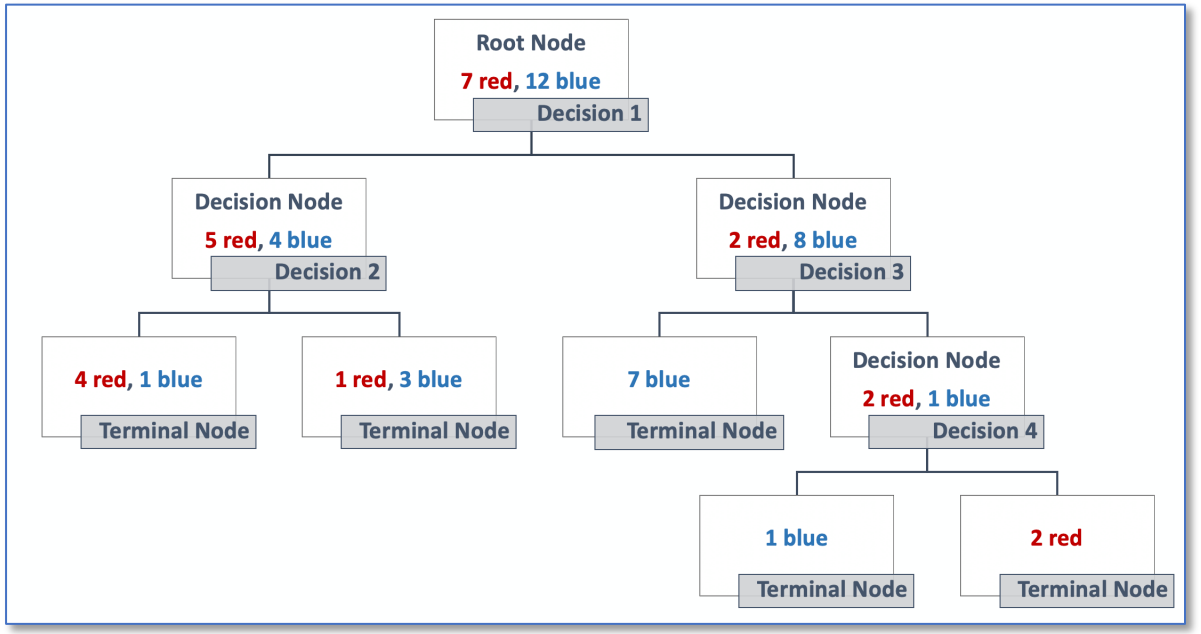
Ensemble Learning

- Given there are many supervised learning methods, it is always tantalizing to think whether some sort of “average” from an ensemble of learners will lead to better predictive performance. Research studies in this direction leads to ensemble learning (Dietterich, 2000).
- Two leading ideas regarding how an ensemble of learners is constructed:
  - Bagging (bootstrap aggregating) (Breiman, 1996)
    - The ensemble is created through bootstrapping the dataset (Efron & Tibshirani, 1994)
    - The outputs from the learners from each sample will be combined (average or voting) to improve the predictive performance.
    - Example: Random Forest (Ho, 1995; Breiman, 2001)
  - Boosting (Friedman, 2001)
    - The ensemble is constructed sequentially, rather than in parallel, as in the bagging approach.
    - A learner is first trained on the full dataset and subsequent learners are added by fitting the residuals (after applying all preceding learners), by which new learners will assign more weights to the poorly predicted observations by the preceding learners.
    - Example: Gradient Boosting Machine

Decision Tree

A decision tree works by forming decision rules on the features recursively to minimize some loss function of the classification.





Decision Tree Cont'd

- Loss function
  - Gini index as used in the CART algorithm (Breiman et al., 1984).
  - Information Gain as used in the Iterative Dichotomiser 3 algorithm (Quinlan, 1986).
- Stopping rule
  - Condition under which no further splitting will happen.
  - E.g., minimum leaf size
- Pruning
  - A procedure to remove the sub-nodes from decision nodes performed after the training to avoid overfitting.
- Main drawback:
  - Instability of the results, as a small change in a dataset may lead to quite different decision rules.

Random Forest

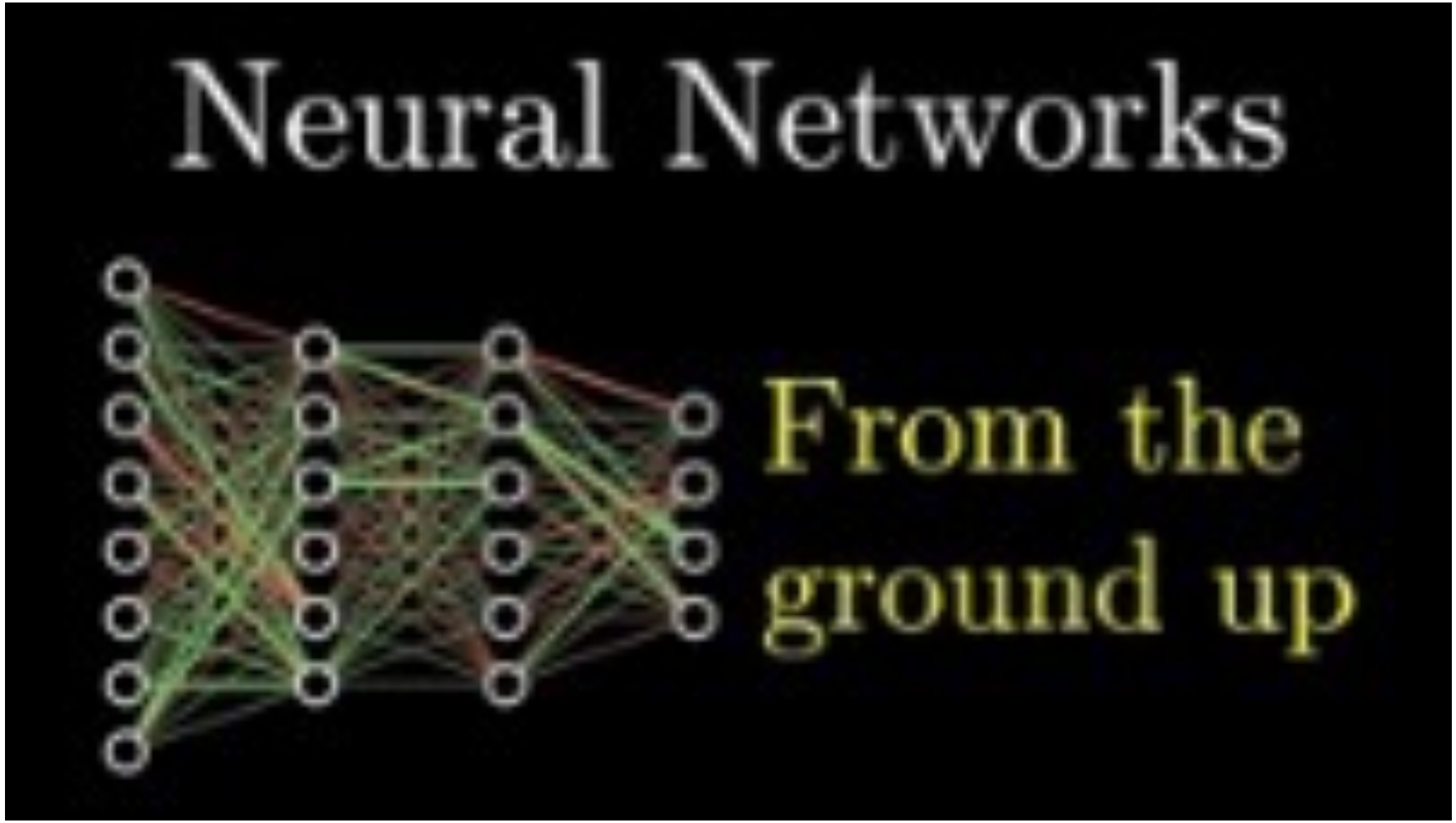
- Suppose we have N observations and M features in a dataset, then a typical random forest classifier goes as follows:
    1. Create K samples of the dataset using the bootstrapping technique, and each sample has N observations
    2. For each of the K samples, a decision tree is applied. In each of the decision nodes of each of the trees, a decision is made based on the most discriminative one of a randomly chosen F features from the total M features – random subspace or feature bagging (Ho, 1998).
    3. The average/voting of multiple trees will be the final estimator
- Hyper-parameters
    - The number of trees (K)
    - The size of the feature subspace (F)

Gradient Boosting Machine

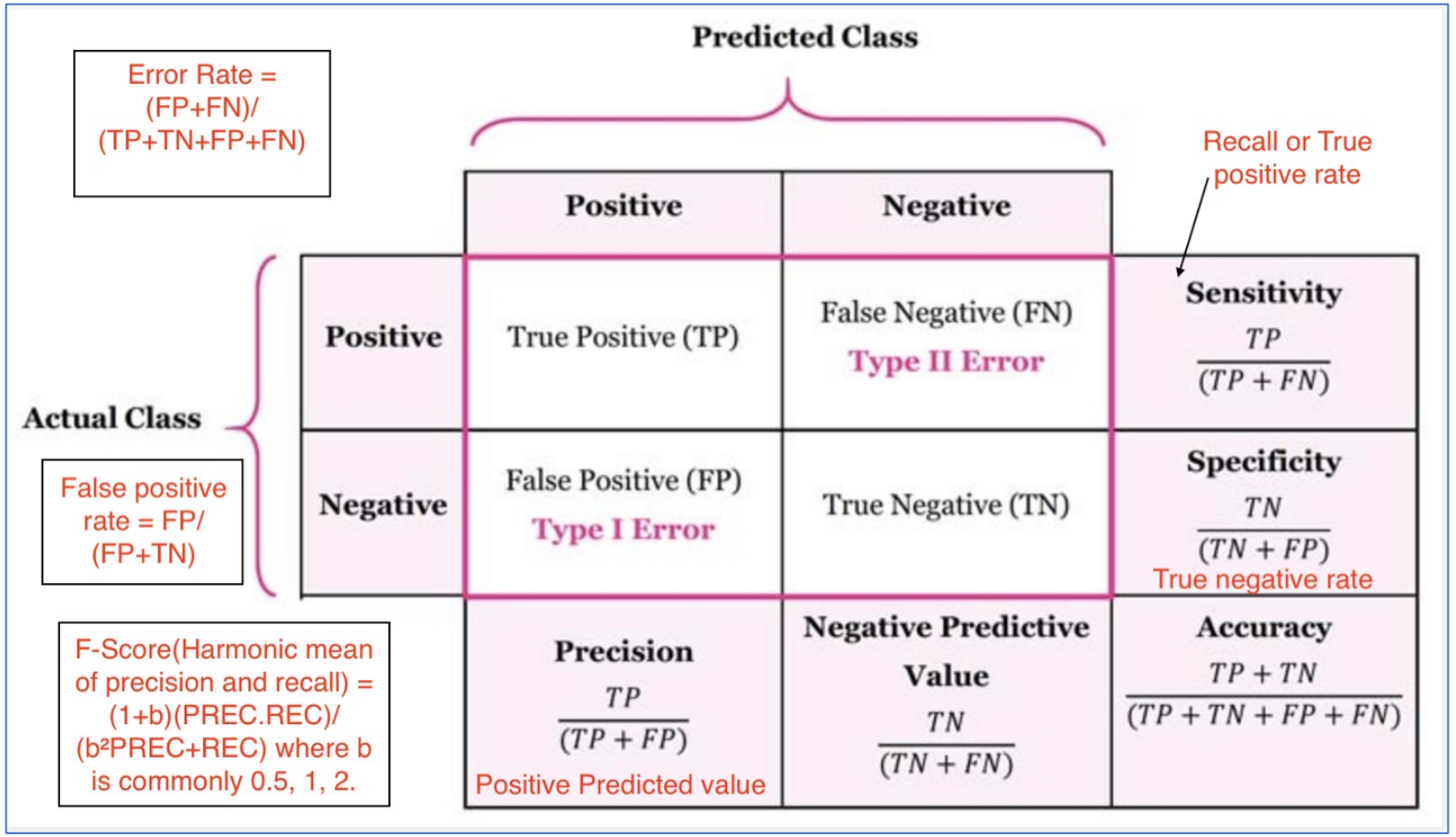
- Assumption: the ideal mapping between the dependent and independent variables could be approximated by the sum of many functions belonging to the same parametric family.
- The parameters are fixed iteratively as following
  1. Fit the data ($\boldsymbol{X}$, y) with a base tree $h_0(\boldsymbol{X};\boldsymbol{\alpha}_0)$
  2. Add another tree $h_1(\boldsymbol{X};\boldsymbol{\alpha}_1)$ by fitting the residuals $(\boldsymbol{X}, y - h_0(\boldsymbol{X};\boldsymbol{\alpha}_0))$
  3. Repeat the procedure leads to $F(\boldsymbol{X}) = \sum_{m=0}^{M} \beta_m h_m(\boldsymbol{X};\boldsymbol{\alpha}_m)$
- Where is the "Gradient" from?
  - For loss function: $L(y, F(\boldsymbol{X})) = \frac{1}{2}[y - F(\boldsymbol{X})]^2$
  - The Residual: $y_i - F_m(\boldsymbol{X}_i) = -\frac{\partial L(y_i,\ F(\boldsymbol{X}_i))}{\partial F(\boldsymbol{X}_i)}\Big|_{F(\boldsymbol{X})=F_{m-1}(\boldsymbol{X})} \equiv -g_m(\boldsymbol{X}_i)$

See the book chapter for more details

Neural Network

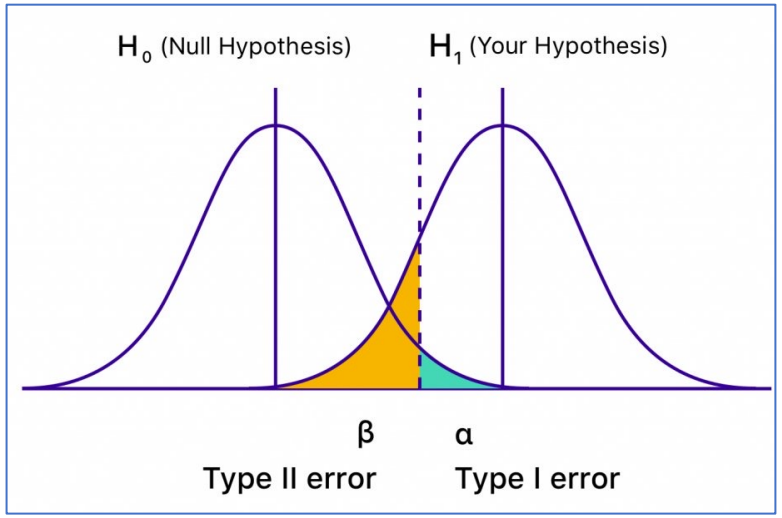


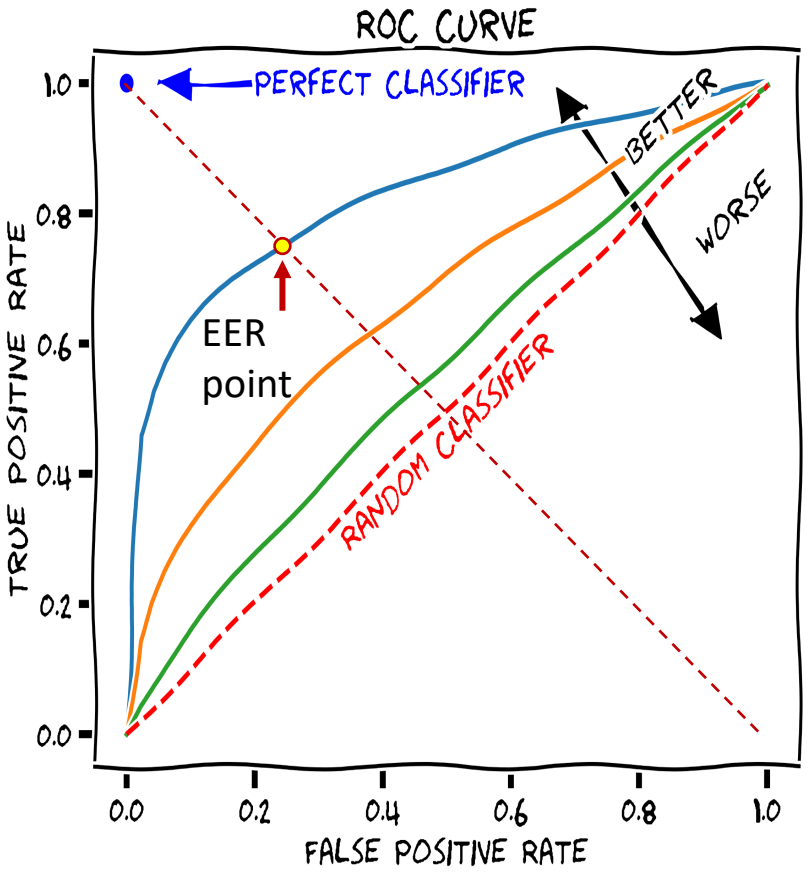
Evaluation Metrics for Binary Classification



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

Receiver Operating Characteristic - ROC



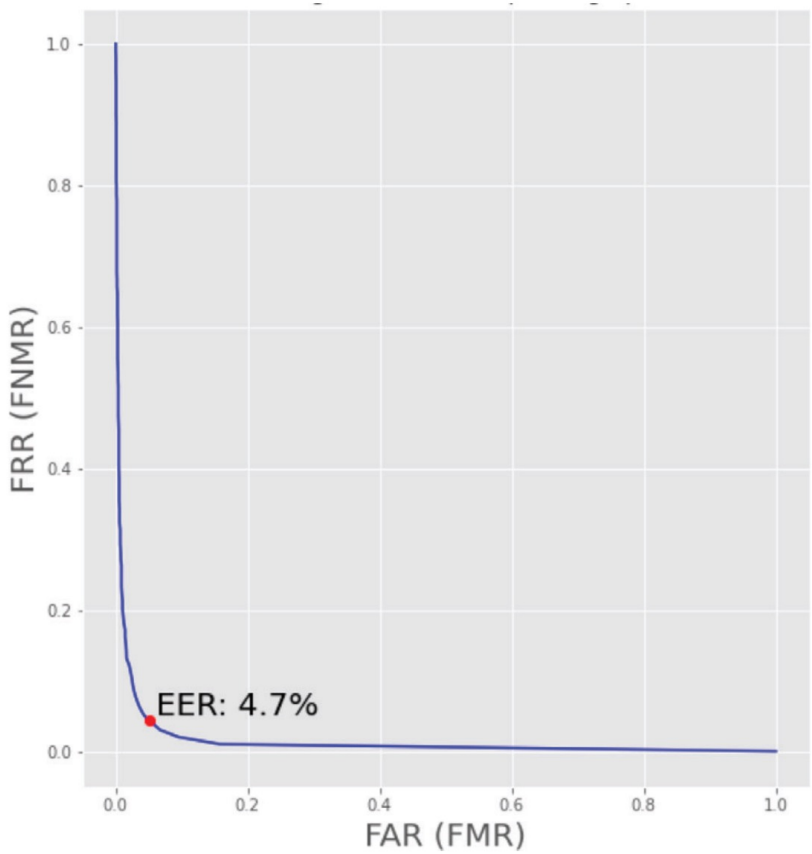


- Area under curve (AUC):
  - 0.5: non discriminative
  - 0.7 – 0.8: acceptable
  - 0.8 – 0.9: excellent
  - > 0.9: outstanding
- Equal Error Rate
  - Fingerprint: 0.2%
  - Signature: 2%
  - Voice: 2%
  - Keystroke: 4.7%
  - TouchID: 0.002%
  - FaceID: 0.001%

Example Application – Keystroke Biometrics

- Writing process data – keystrokes
- Feature representation of keystrokes
- Mapping between keystroke features and labels (same person or not)

Feature name	Definition	Within-person correlation
inword_logIKI_median*	Median duration of in-word keystrokes, measured in log milliseconds	0.95
inword_logIKI_mean	Mean duration of in-word keystrokes in log milliseconds	0.95
wordinitial_logIKI_median*	Median duration of word-initial keystrokes in log milliseconds	0.92
append_interword_interval_logIKIs_mean	The mean log interkey intervals for keystrokes that add white space between words	0.92
wordinitial_logIKI_mean	Mean duration of word-initial keystrokes in log milliseconds	0.92
append_interword_interval_logIKIs_median*	The median log interkey interval for keystrokes that add white space between words	0.91
append_interword_interval_speed_median*	The speed of keystrokes that add white space between words, measured in characters per second	0.91
wordinitial_char_per_sec_median*	Median speed of typing the first character of a word, in characters per second	0.91
iki400_AppendBurst_len_mean	Mean length in characters of bursts of append keystrokes where no pause is greater than 400 milliseconds	0.91
iki400_AllActionBurst_len_mean	Mean length in characters of bursts where all keystrokes count as part of the burst, and bursts end on pauses longer than 400 milliseconds	0.90
initial_backspace_char_per_sec_median*	The median speed of the first in a series of backspace actions, measured in characters per second	0.90
iki200_AppendBurst_len_mean	Mean length in characters of bursts of append keystrokes where no pause is longer than 200 milliseconds	0.90
initial_backspace_logIKI_median*	The median log interkey interval for backspace actions that appear first in a series of backspace actions	0.89



(Choi, Hao, Deane & Zhang, 2021)

Unsupervised Learning

Cluster Analysis

- Group similar things together
- What is “similar”
  - Similarity is measured by certain distance in a metric space

A **metric space** is an ordered pair (M, d) where M is a set and d is a metric on M, i.e., a function

$$d\colon M \times M \to \mathbb{R}$$

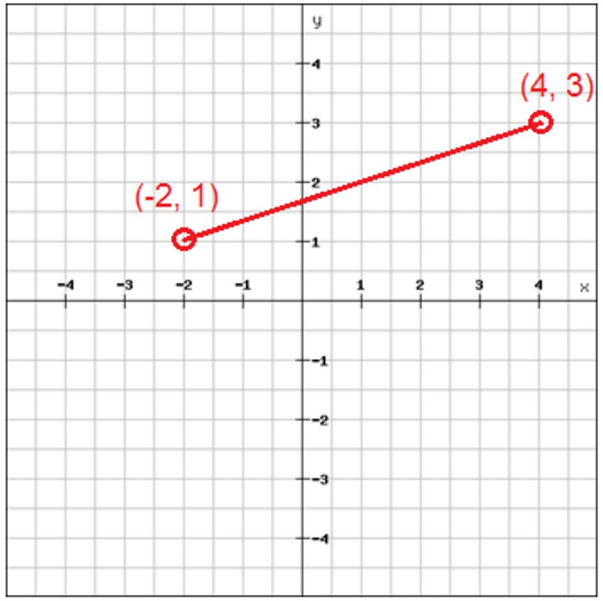
such that for any $x, y, z \in M$, the following holds:[2]

1. $d(x, y) = 0 \Leftrightarrow x = y$ identity of indiscernibles
2. $d(x, y) = d(y, x)$ symmetry
3. $d(x, z) \leq d(x, y) + d(y, z)$ subadditivity or triangle inequality

- Hard clusters vs. fuzzy clusters
- Clusters are not unique, depending on the ways you look at or interpret the data

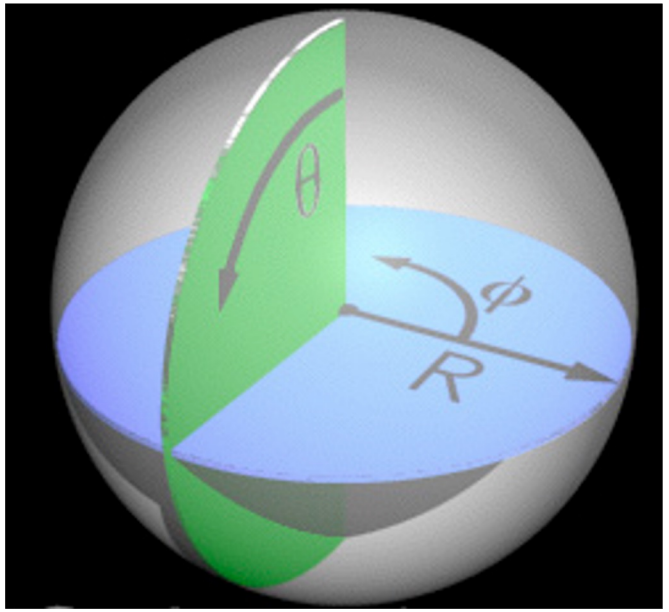
Example Distance

- Euclidean Space



$$ds^2 = dx^2 + dy^2$$

- Non-Euclidean Space



$$ds^2 = R^2 d\theta^2 + R^2 \sin^2\theta d\phi^2$$

Popular Distances in ML

- Euclidean distance
- Minkowski distanc $D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$
- Mahalanobis distan $d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$
- Cosine distanc $1 - \frac{u \cdot v}{||u||_2 ||v||_2}$
- Edit distance: the minimum number of editing operations to bring two strings the same

Analyzing Process Data from Game/Scenario-Based Tasks: An Edit Distance Approach (Hao, Shu, & von Davier, 2015)

Distances in SciPy package

Distance functions between two numeric vectors `u` and `v`. Computing distances over a large collection of vectors is inefficient for these functions. Use `pdist` for this purpose.

braycurtis(u, v[, w])	Compute the Bray-Curtis distance between two 1-D arrays.
canberra(u, v[, w])	Compute the Canberra distance between two 1-D arrays.
chebyshev(u, v[, w])	Compute the Chebyshev distance.
cityblock(u, v[, w])	Compute the City Block (Manhattan) distance.
correlation(u, v[, w, centered])	Compute the correlation distance between two 1-D arrays.
cosine(u, v[, w])	Compute the Cosine distance between 1-D arrays.
euclidean(u, v[, w])	Computes the Euclidean distance between two 1-D arrays.
jensenshannon(p, q[, base])	Compute the Jensen-Shannon distance (metric) between two 1-D probability arrays.
mahalanobis(u, v, VI)	Compute the Mahalanobis distance between two 1-D arrays.
minkowski(u, v[, p, w])	Compute the Minkowski distance between two 1-D arrays.
seuclidean(u, v, V)	Return the standardized Euclidean distance between two 1-D arrays.
sqeuclidean(u, v[, w])	Compute the squared Euclidean distance between two 1-D arrays.
wminkowski(u, v, p, w)	Compute the weighted Minkowski distance between two 1-D arrays.

Distance functions between two boolean vectors (representing sets) `u` and `v`. As in the case of numerical vectors, `pdist` is more efficient for computing the distances between all pairs.

dice(u, v[, w])	Compute the Dice dissimilarity between two boolean 1-D arrays.
hamming(u, v[, w])	Compute the Hamming distance between two 1-D arrays.
jaccard(u, v[, w])	Compute the Jaccard-Needham dissimilarity between two boolean 1-D arrays.
kulsinski(u, v[, w])	Compute the Kulsinski dissimilarity between two boolean 1-D arrays.
rogerstanimoto(u, v[, w])	Compute the Rogers-Tanimoto dissimilarity between two boolean 1-D arrays.
russellrao(u, v[, w])	Compute the Russell-Rao dissimilarity between two boolean 1-D arrays.
sokalmichener(u, v[, w])	Compute the Sokal-Michener dissimilarity between two boolean 1-D arrays.
sokalsneath(u, v[, w])	Compute the Sokal-Sneath dissimilarity between two boolean 1-D arrays.
yule(u, v[, w])	Compute the Yule dissimilarity between two boolean 1-D arrays.

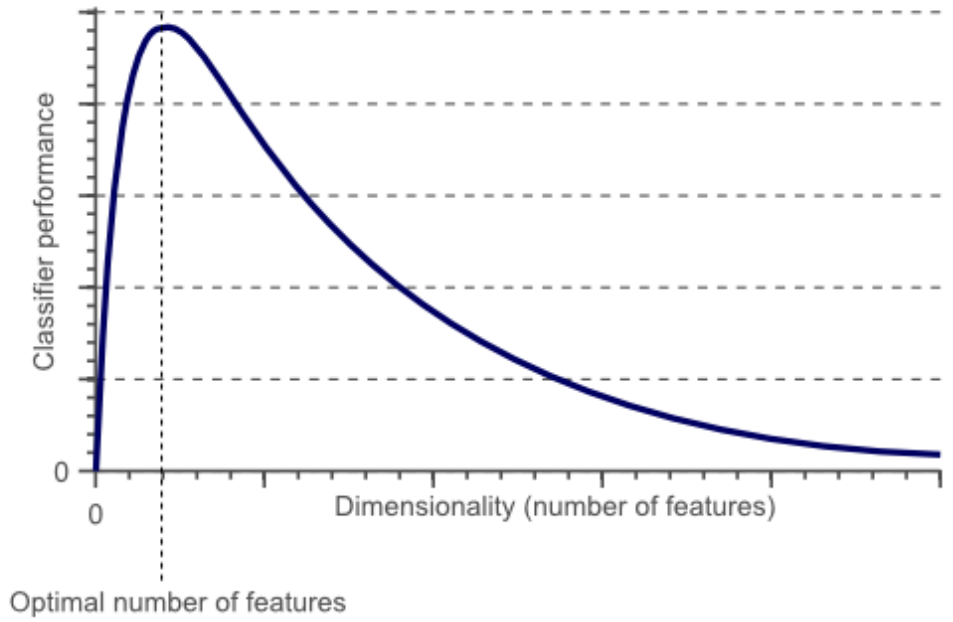
hamming also operates over discrete numerical vectors.

High Dimensional Space

- High dimensional space is beyond our intuition
- Do not take things for granted in high dimensional space
- Curse of dimensionality

$$\lim_{d \to \infty} E\left(\frac{\text{dist}_{\max}(d) - \text{dist}_{\min}(d)}{\text{dist}_{\min}(d)}\right) \to 0$$

Beyer et al., 1997



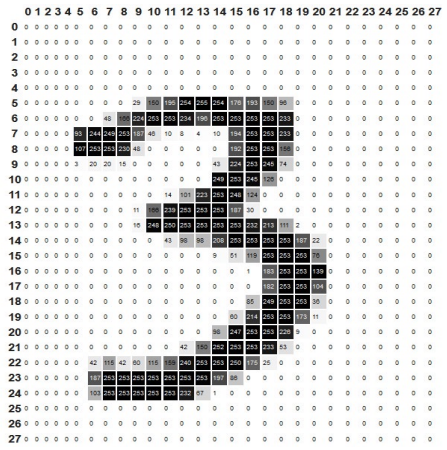
https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

Manifold Hypothesis

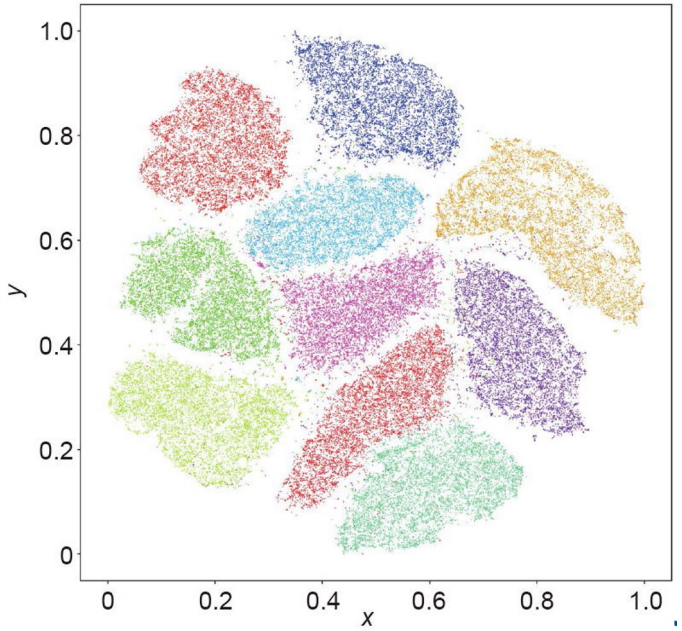
Real world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space.

MNIST dataset

Each handwritten digit image is 28 by 28 pixels, leading to a space of 784 dimensions



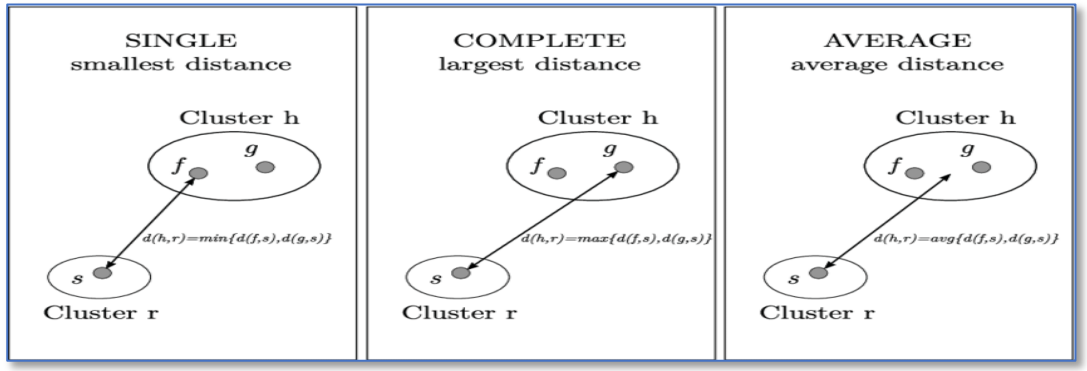
TSNE embedding



https://www.sciencedirect.com/science/article/pii/S2095809919302279#b0015

Hierarchical Clustering

- Hierarchical clustering: method seeking to build a hierarchy of clusters.
- Two strategies:
  - Divisive – top down approach
  - Agglomerative – bottom up approach
- Distance: pairwise distance between the points
- Linkage: determines the distance between sets of points as a function of the pairwise distances between points.

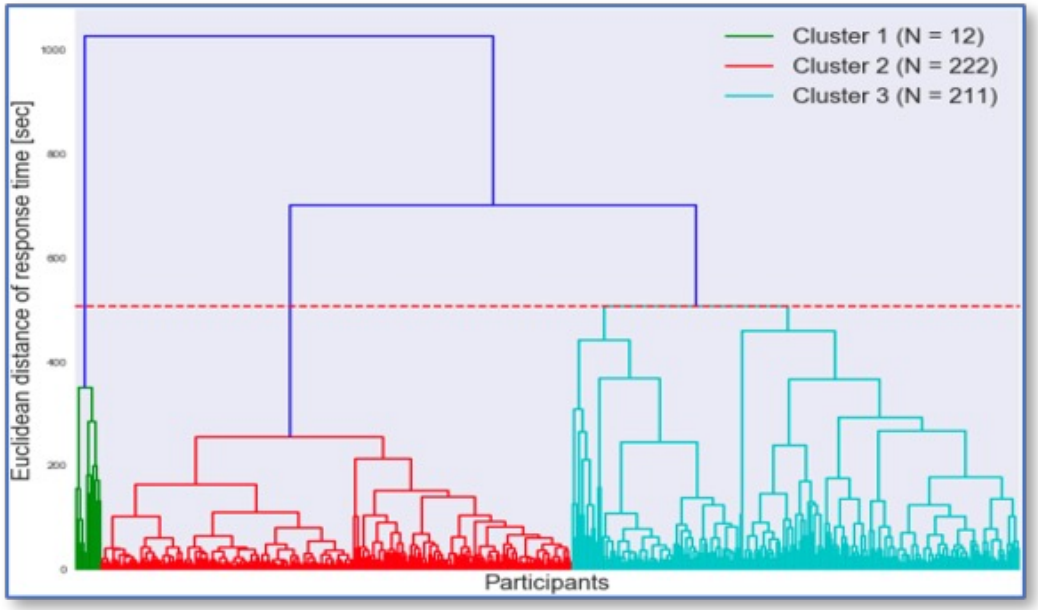


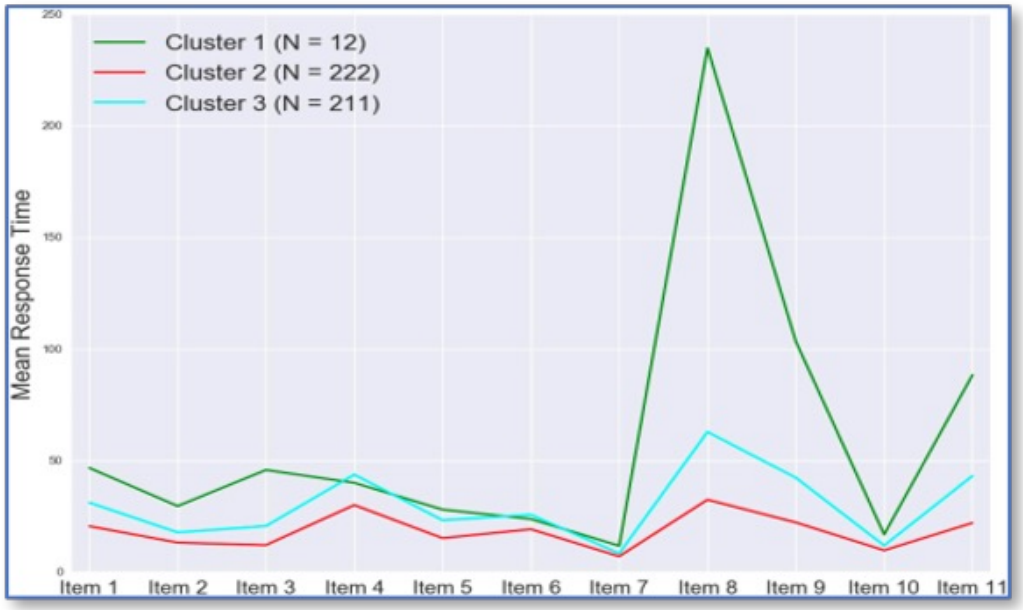
- Ward's minimum variance criterion: minimizes the total within-cluster variance.

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

Example Application – Student Clustering

- Based on student's response time to different items, cluster them into different groups (response style)





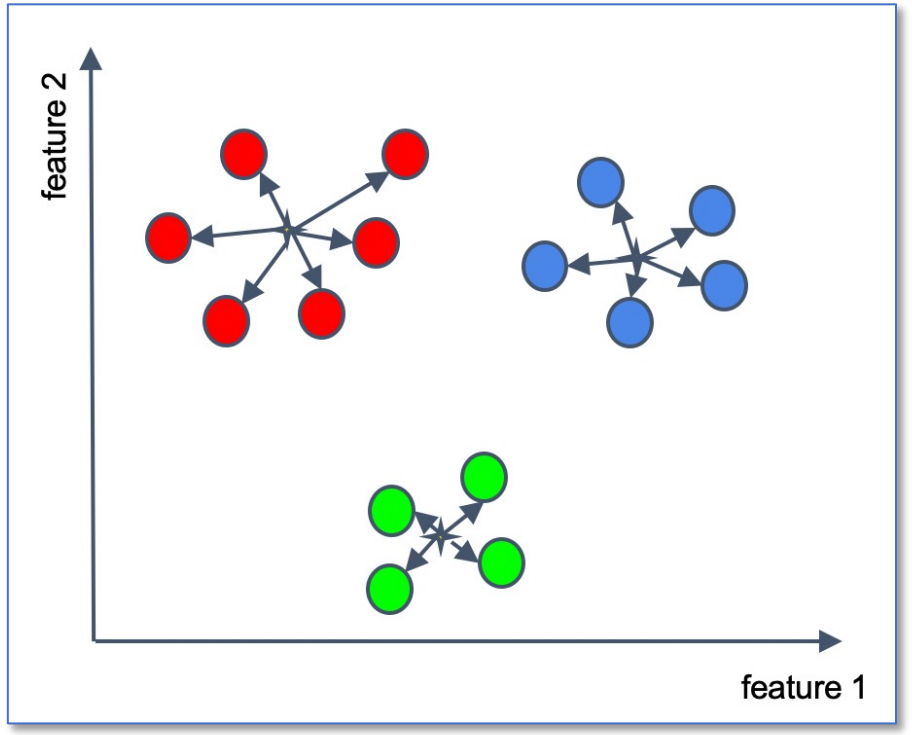
Lee, Hao, et al., 2019

K-means

- K-means (NP hard computation): divides a set of N samples X into K disjoint clusters C, each described by the mean μj of the samples in the cluster. The means are commonly called the cluster “centroids”
- The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of-squares criterion:

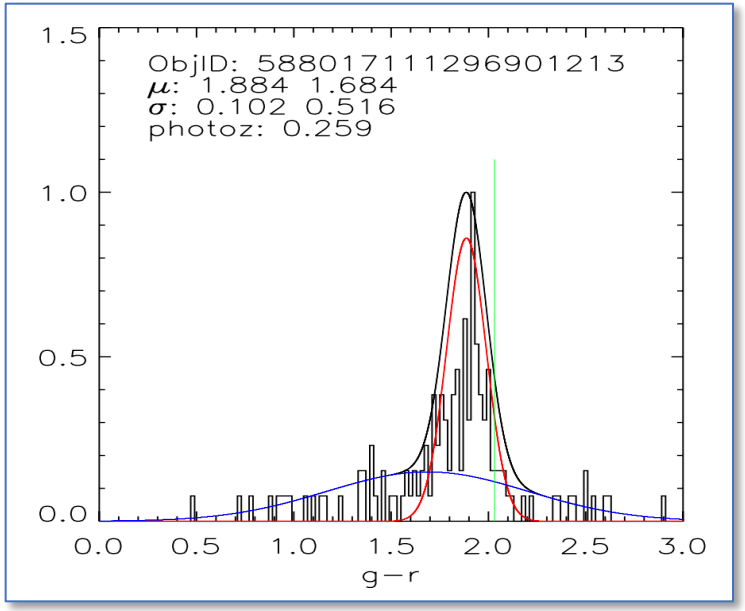
$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

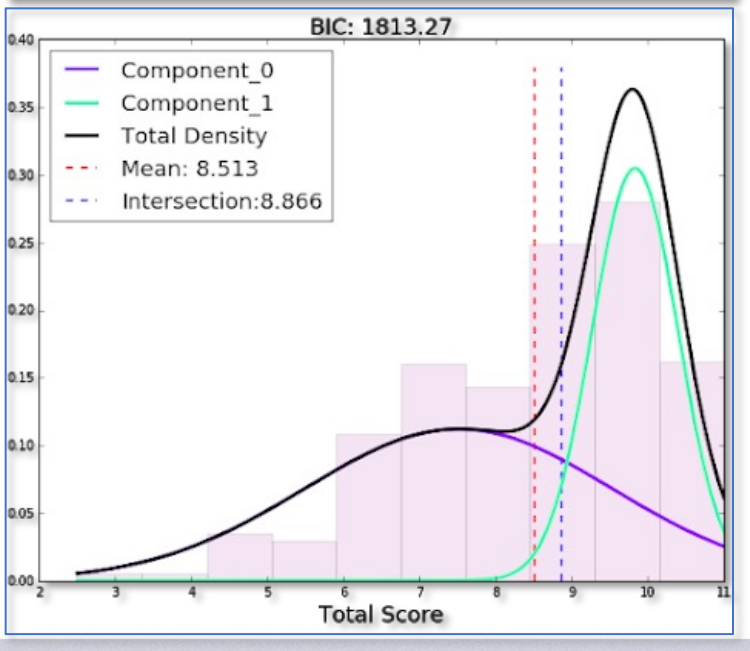
- Need to specify the number of clusters before running the algorithm



Model-based clustering

- Gaussian mixture model: the underlying distribution is approximated as a weighted sum of gaussian distributions
- Using BIC/AIC to determine the number of mixtures





THE ASTROPHYSICAL JOURNAL
SUPPLEMENT SERIES

A GMBCG GALAXY CLUSTER CATALOG OF 55,424 RICH CLUSTERS FROM SDSS DR7

Jiangang Hao[1], Timothy A. McKay[2,3], Benjamin P. Koester[4], Eli S. Rykoff[12,5,6], Eduardo Rozo[13,7], James Annis[1], Risa H. Wechsler[8,9], August Evrard[2,3], Seth R. Siegel[2], Matthew Becker[10]
+ Show full author list

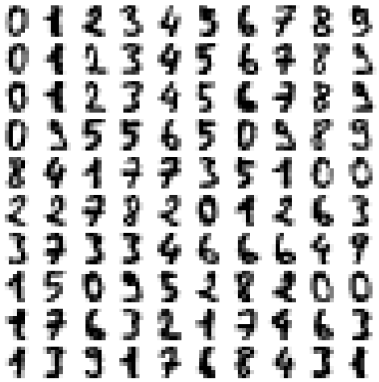
The Astrophysical Journal Supplement Series, Volume 191, Number 2
Citation Jiangang Hao *et al* 2010 *ApJS* **191** 254

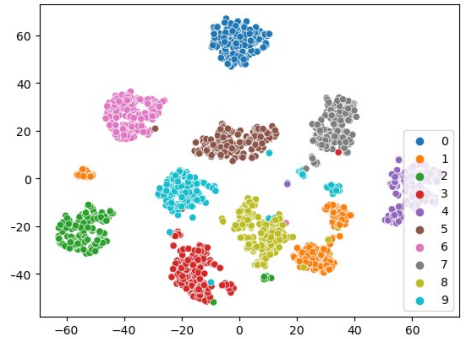
Number of Clusters

- A tricky question
- Overall, clusters should maximize between cluster distance and minimize within cluster distance
- Statistical measures: function(Data, Cluster_label)
  - Calinski-Harabasz index: ratio of the sum of between-cluster dispersion and of within-cluster dispersion. The bigger the better.
  - Silhouette score: measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Best value is 1 and worst is -1.
- Hierarchical clustering
  - Visual check
- K-means
  - Inertia: within-cluster sum of square $\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$

Dimensionality Reduction – t-SNE

- Principal Component Analysis (PCA) does not address non-linear variance
- t-SNE: t-distributed stochastic neighbor embedding
  - N data points x_1, x_2, …, x_n in high dimensional space
  - Look for a low dimensional representation y_1, y_2, …, y_n of the N data points
  - Such that $p_{ij}(X) \sim q_{ij}(Y)$, where p_{ij} and q_{ij} are distribution that characterize how close the data points are in the corresponding space.





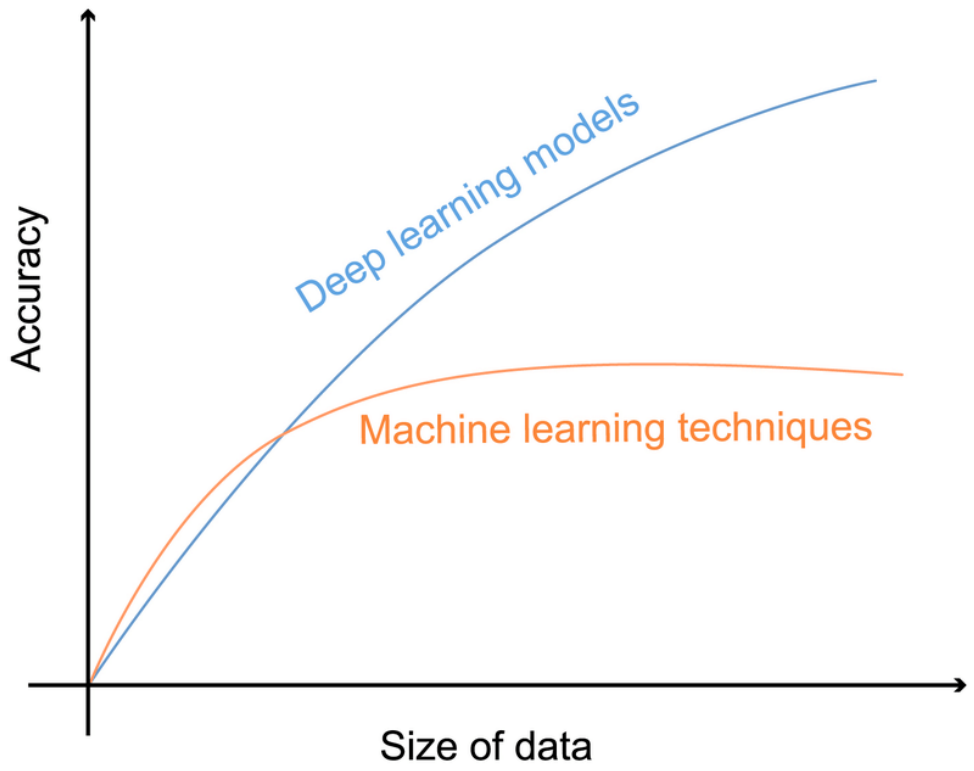
$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma^2\right)} \qquad q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}} \qquad C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE Hyperparameters

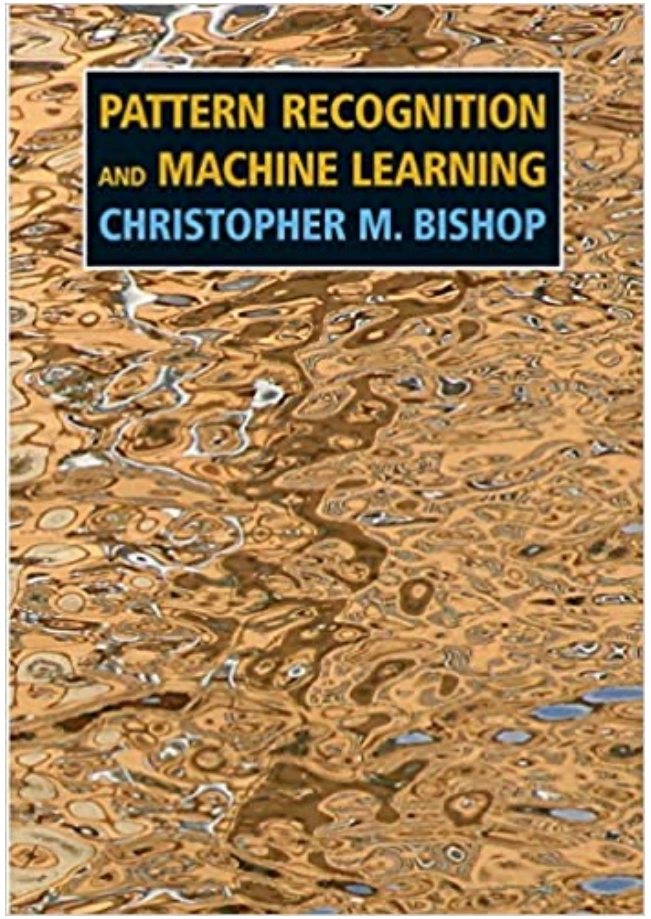
- Perplexity: related to the number of nearest neighbors that are used in other manifold learning algorithms. Larger datasets usually required a larger perplexity
  - Usually set between 5 and 50
- Early exaggeration: controls how tight natural clusters in the original space will be in the embedded space, and how much space will be between them
  - Default = 12 in sklearn
- Parameter playground: https://distill.pub/2016/misread-tsne/

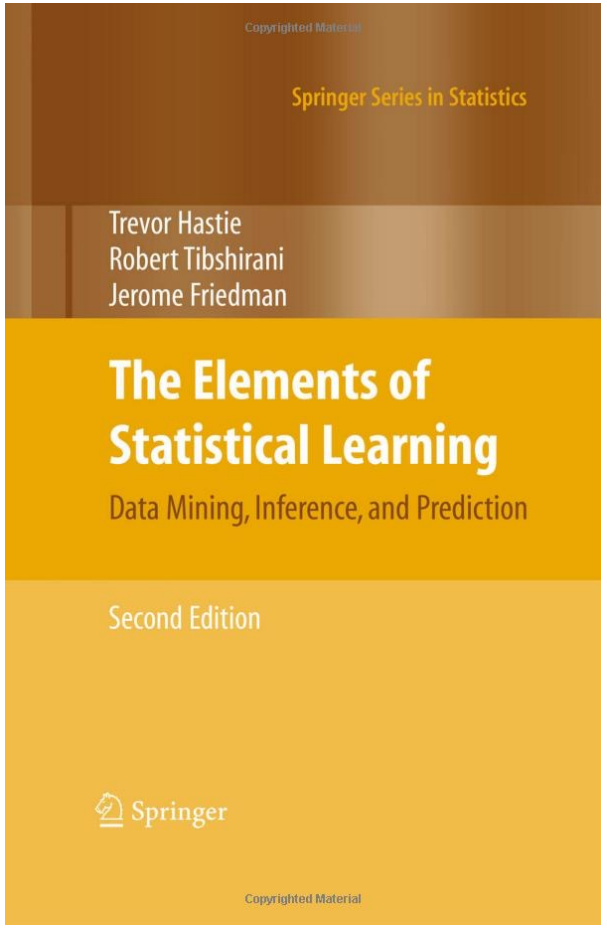
Summary

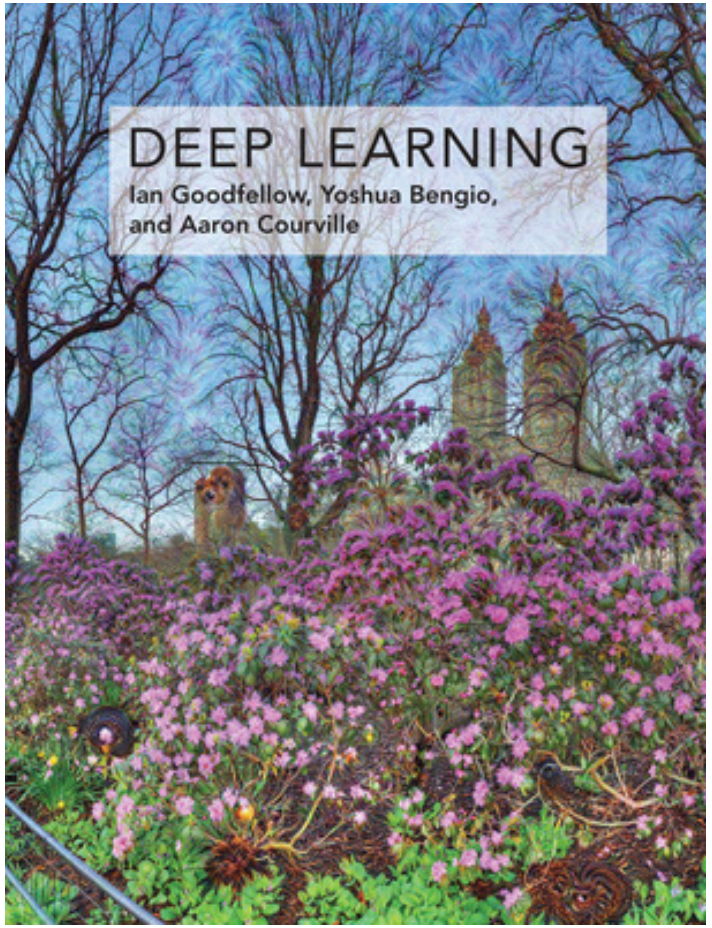
- Machine learning is a very active research field
- We discussed the so-called traditional machine learning methods and did not cover neural network and deep learning
- Deep learning refers to neural network models with multiple layers and its accuracy increases as size of data increases.
- The more you learn, the less you “know”



Recommended Books

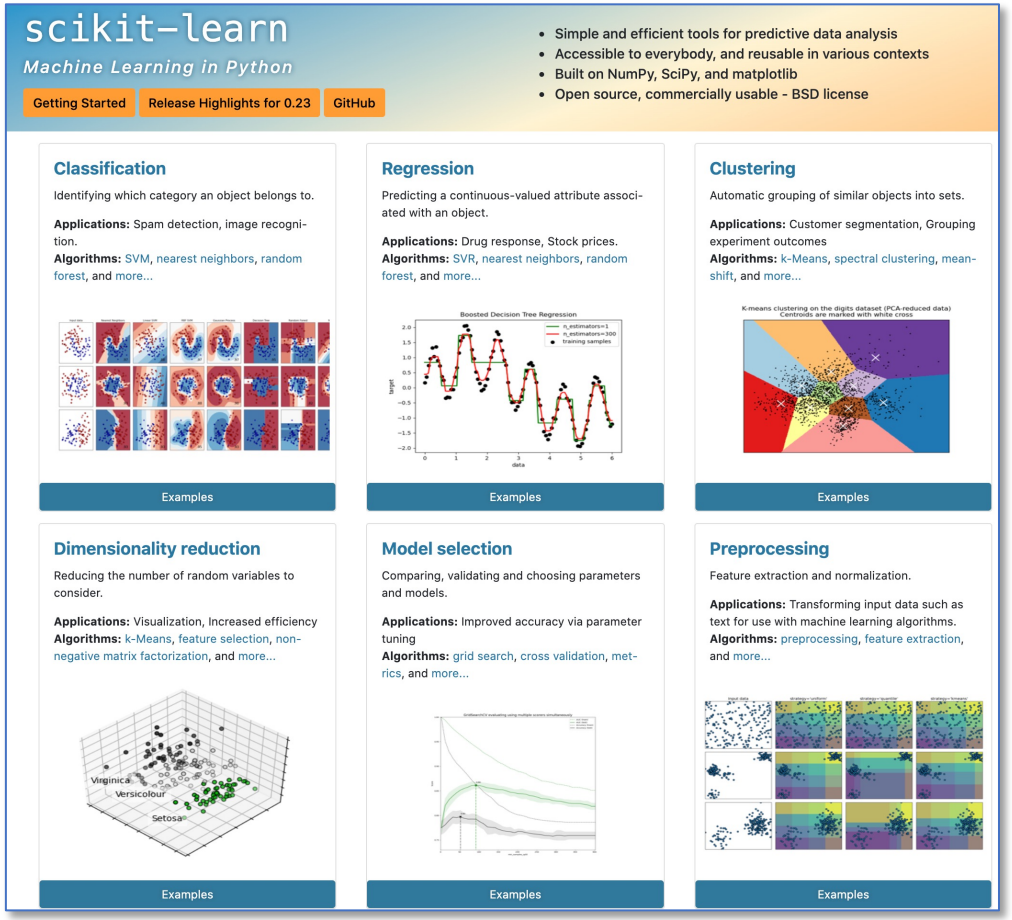




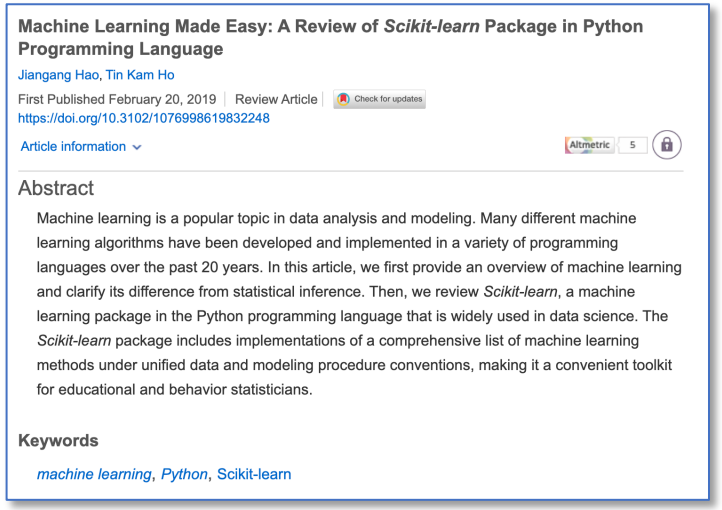


Scikit-learn package in Python

Scikit-learn



One of the main open-source ML package in Python



Machine Learning Made Easy: A Review of *Scikit-learn* Package in Python Programming Language

Jiangang Hao, Tin Kam Ho

First Published February 20, 2019 | Review Article | Check for updates
https://doi.org/10.3102/1076998619832248
Article information
Altmetric 5

Abstract

Machine learning is a popular topic in data analysis and modeling. Many different machine learning algorithms have been developed and implemented in a variety of programming languages over the past 20 years. In this article, we first provide an overview of machine learning and clarify its difference from statistical inference. Then, we review *Scikit-learn*, a machine learning package in the Python programming language that is widely used in data science. The *Scikit-learn* package includes implementations of a comprehensive list of machine learning methods under unified data and modeling procedure conventions, making it a convenient toolkit for educational and behavior statisticians.

Keywords

machine learning, *Python*, Scikit-learn

```
model = classifier(hyperparameters = something)
model.fit(X_train, y_train)
y_test = model.predict(X_test)
```

Setup Environment – from Anaconda to Mamba

- **Step 1: Install mamba**
  - Go to https://github.com/conda-forge/miniforge to download the "Mambaforge" (NOT the other ones). After downloading, install it into your system.
- **Step 2: Create working virtual environment**
  - you want to create an environment for python 3.9, you can do:

    ```
    mamba create --name py39 python=3.9
    ```

  - After the virtual environment is created, you can activate/deactivate it as:

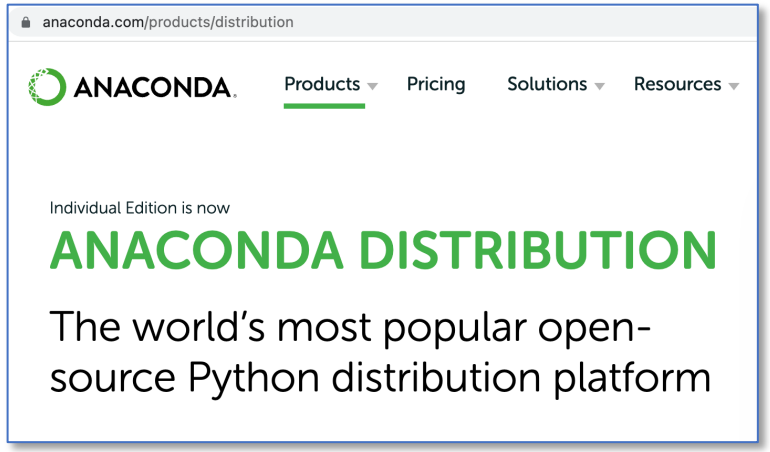
    ```
    mamba activate py39 mamba deactivate
    ```

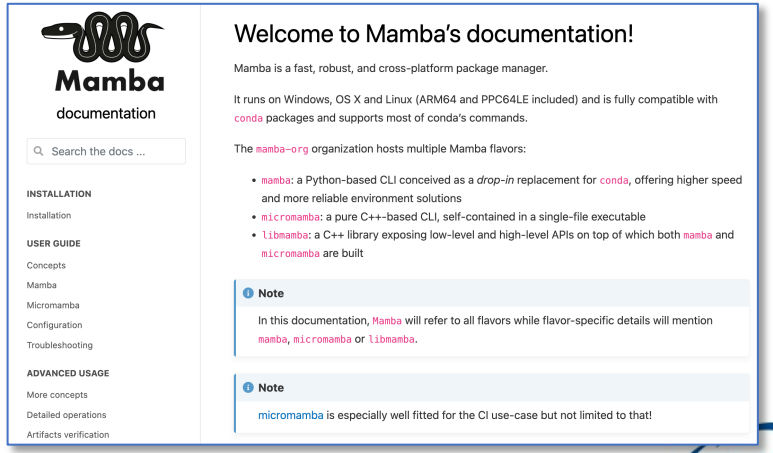
  - If you want to install python packages into your newly created environment, you can do (after you are in that environment), for example,

    ```
    mamba install numpy scipy pandas jupyterlab seaborn
    ```

  - After you installed jupyterlab, you can start it by typing

    ```
    jupyter lab
    ```





Scikit-learn PCA Example

- Iris Data Set: 150 instances x 4 features (Fisher, 1950)
  - sepal length [cm]
  - sepal width [cm]
  - petal length [cm]
  - petal width [cm]
  - Class: Iris Setosa, Iris Versicolour, Iris Virginica

Notebook demo

Assignments

- Read the attached book chapter on supervised learning
- Practice the examples in the demo notebook

Interactive Machine Learning