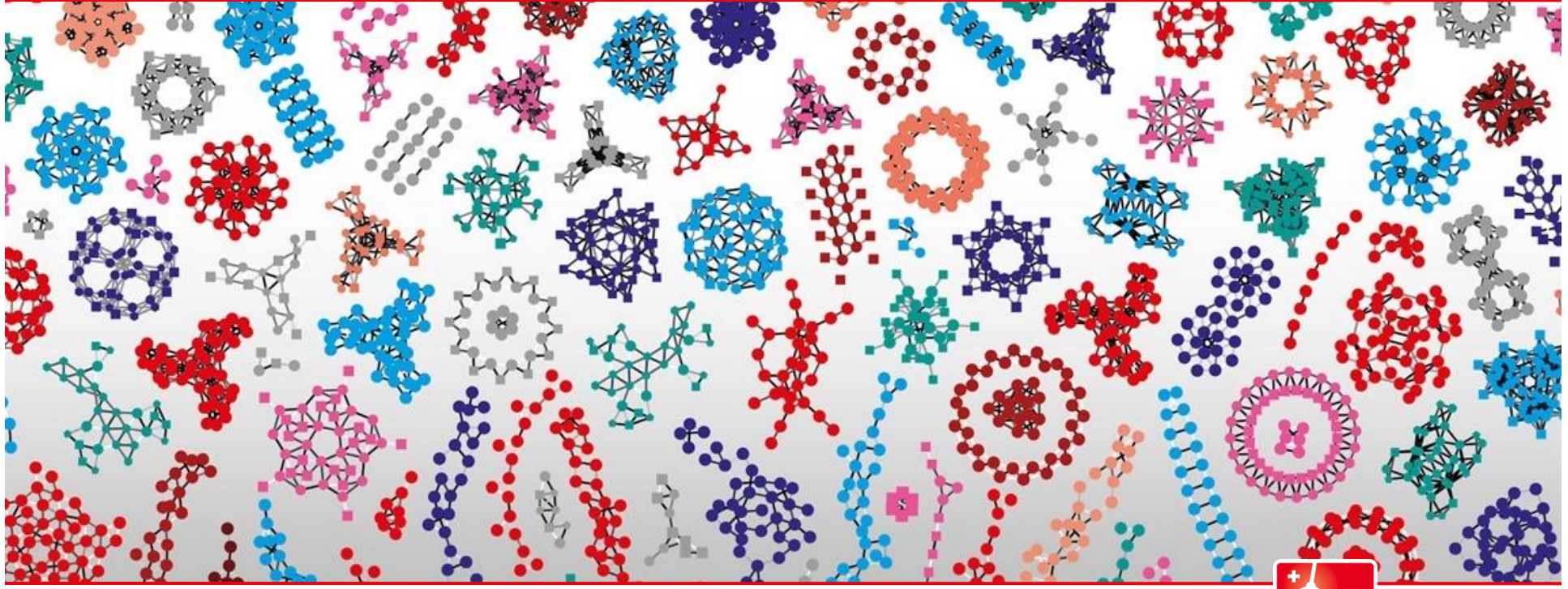


Swiss Institute of Bioinformatics

SIB Swiss Institute of Bioinformatics

Introduction to Machine Learning with Python, 03 - 04 October 2022

Wandrille Duchemin and Markus Müller

Code of conduct

SIB abides by the ELIXIR Code of Conduct. We are all thus expected to abide by the same code. In summary:

We **value** each other's perspectives providing a safe environment for people to be themselves.

We will **maintain** high ethical standards across all ELIXIR events.

We **adopt** a zero-tolerance approach to harassment and discrimination in any form.

We will **apply** honesty and integrity in the dealing of any transgressions against the Code.

We are **committed** to making ELIXIR events a collaborative, supportive and enjoyable experience.

We will **ensure** that our environment allows everyone to feel respected and included.

https://elixir-europe.org/events/code-of-conduct

A quick round of introduction

Swiss Institute of Bioinformatics

Wandrille Duchemin

- Bioinformatics Support and Training
- SIB Swiss Institute of Bioinformatics / sciCORE UNIBAS

Markus Mueller

- Principal Computational Biologist
- Vital-IT, SIB Swiss Institute of Bioinformatics

Fritz Bayer

- Computational Biology Group , SIB Swiss Institute of Bioinformatics

Monique Zahn

- Training Manager and neXtProt Quality Manager
- SIB Swiss Institute of Bioinformatics

Course agenda

Swiss Institute of Bioinformatics

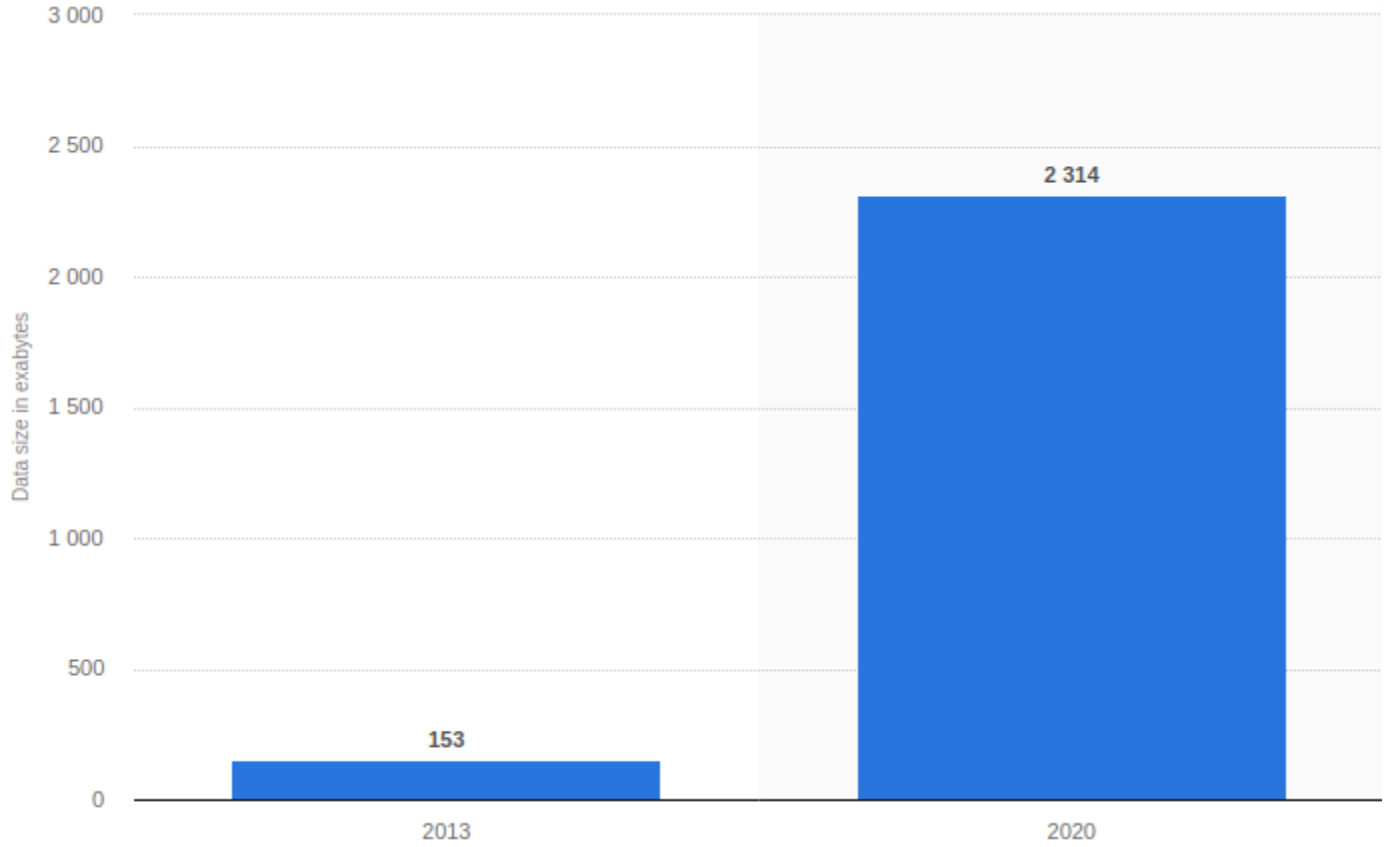
09:00 - 09:30	**Day 1: Course Introduction**
09:30 - 10:00	**Introduction to Machine Learning** (*theory*) (Wandrille)
10:00 - 10:30	**Exploratory Analysis and Unsupervised Learning** (*theory*) - Data preparation for ML (Markus) - Dimensionality reduction/Embedding (Markus)
10:30 - 11:00	*Coffee Break*
11:00 - 12:00	**Exploratory Analysis and Unsupervised Learning** (*practical*) (Markus)
12:00 - 12:30	**Exploratory Analysis and Unsupervised Learning** (*theory*) - Clustering methods (Wandrille)
12:30 - 13:30	*Lunch break*
13:30 - 14:00	**Exploratory Analysis and Unsupervised Learning** (*practical*) (Markus)
14:00 - 14:30	**Supervised Learning** (*theory*) - Introduction (Wandrille) - Performance metrics (Wandrille) K-nearest neighbors (Markus) - Logistic regression (Markus) - Support vector machines (Markus)
14:30 - 15:00	*Coffee Break*
15:00 - 17:00	**Supervised Learning (practical)** (Markus)

09:00 - 09:30	**Welcome Day 2** - Questions from Day 1 - Agenda
09:30 - 10:30	**Supervised Learning** *(practical)* (Markus)
10:30 - 11:00	*Coffee Break*
11:00 - 11:30	**Supervised Learning** (*theory*) - Decision trees (Wandrille) - Random Forests (Wandrille) - Ada Boost (Wandrille) - Gradient boosting (Wandrille)
11:30 - 12:30	Regression(*theory*) (Markus)
12:30 - 13:30	*Lunch break*
13:30 - 14:30	**Regression** (*parctical*) - Linear regression (Markus) - KNN regression (Markus) - Support vector regression (Markus) - Gradient boosting regression (Markus)
14:30 - 15:00	*Coffee Break*
15:00 - 16:30	**Supervised Learning** (*practical*) (Wandrille) - Deep learning (Wandrille) - Neural networks (Wandrille)
16:30 - 17:00	*Closing questions, Discussion*

Isabelle Dupanloup, BCF

Total amount of global healthcare data generated and projections for end 2020 (in exabytes)

Swiss Institute of Bioinformatics

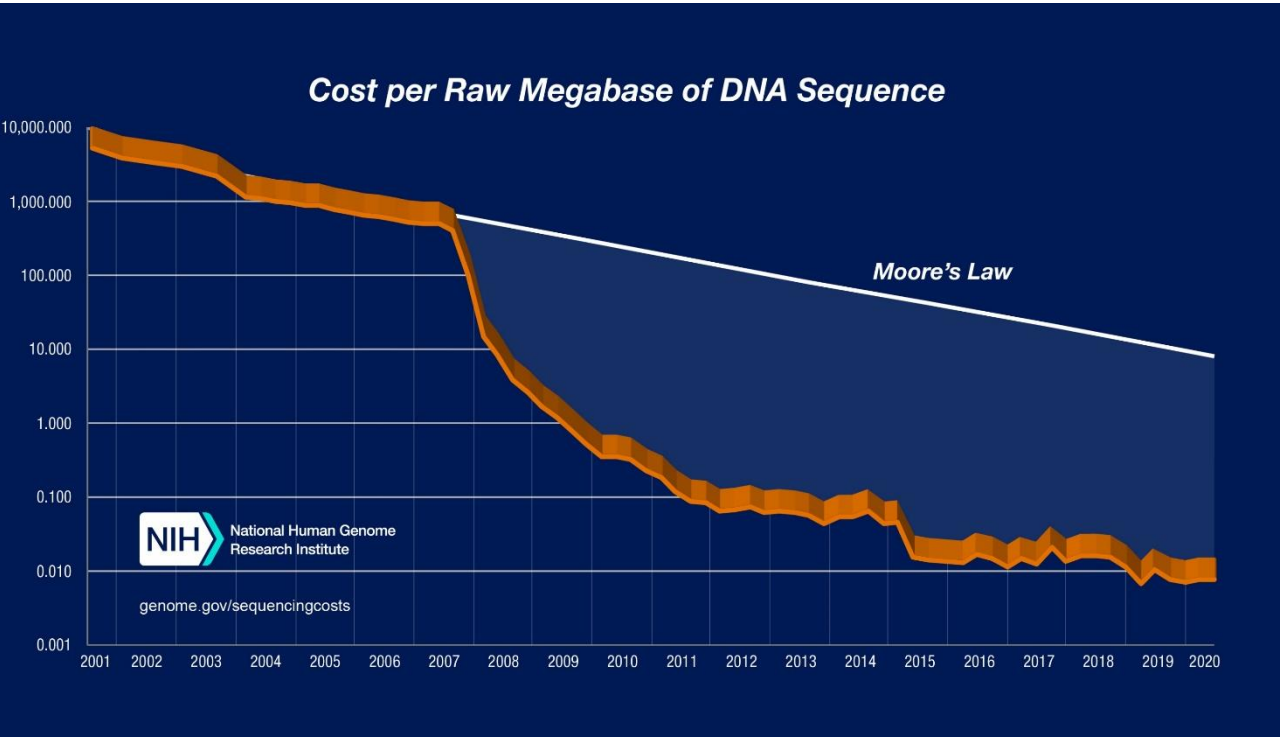


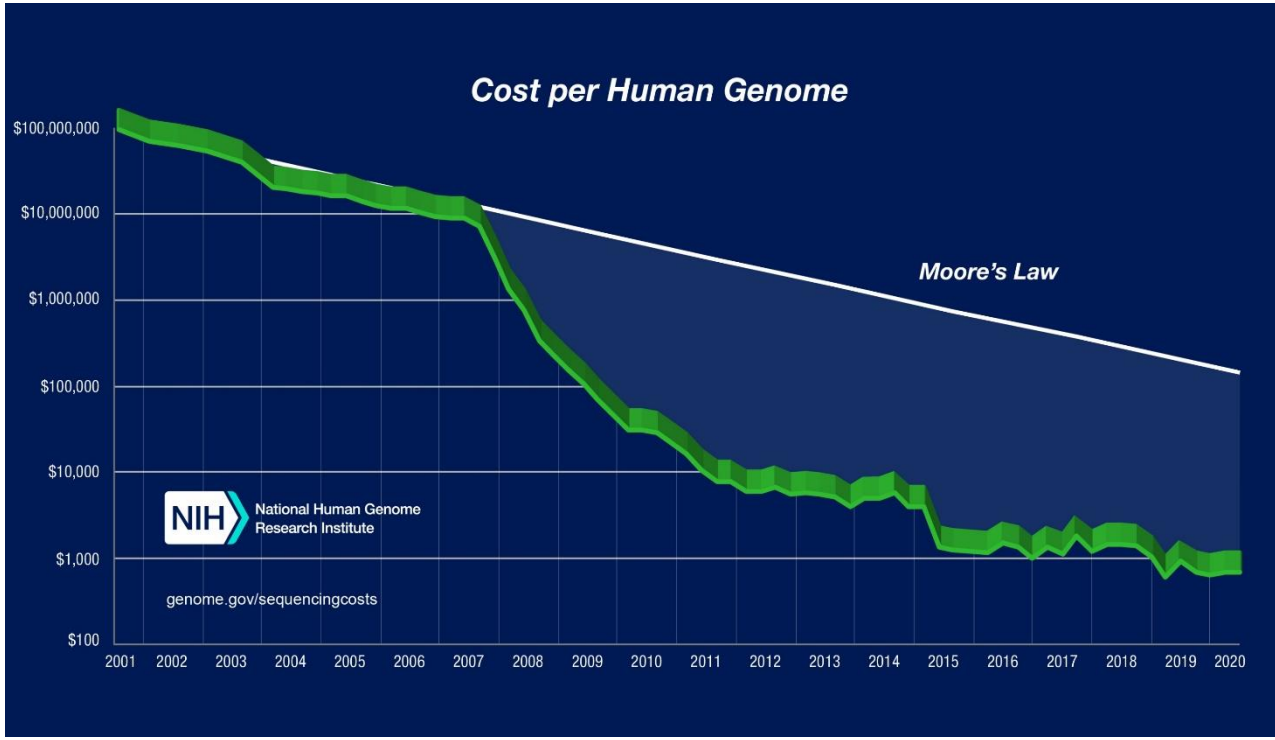
1 exabyte = 1 billion gigabytes

https://www.statista.com/statistics/1037970/global-healthcare-data-volume/

Isabelle Dupanloup, BCF

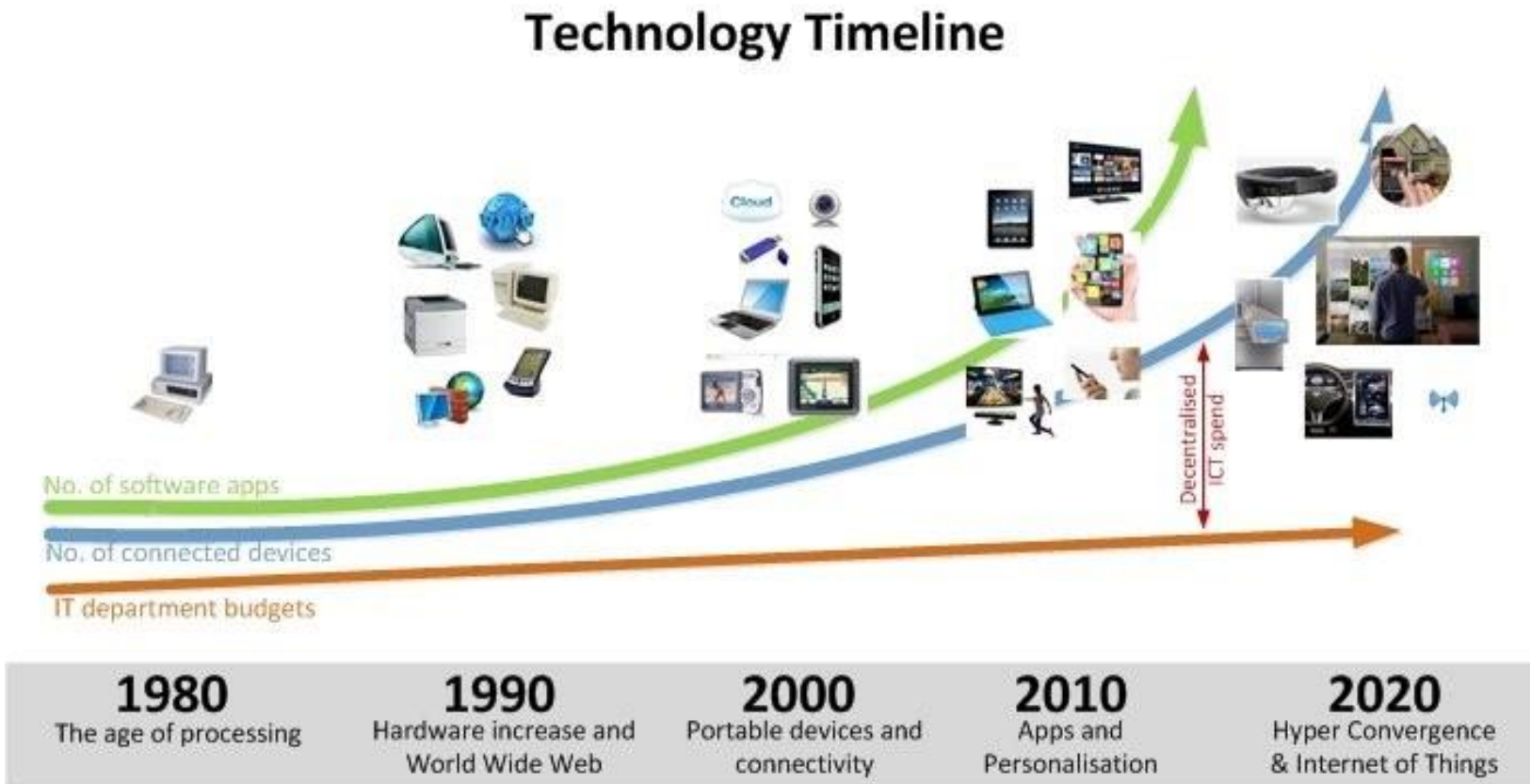
Swiss Institute of Bioinformatics





https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data

Swiss Institute of Bioinformatics



https://www.linkedin.com/pulse/technology-increase-vs-department-budgets-sam-errington/

Isabelle Dupanloup, BCF

From data to knowledge

Swiss Institute of Bioinformatics

Isabelle Dupanloup, BCF

Some definitions

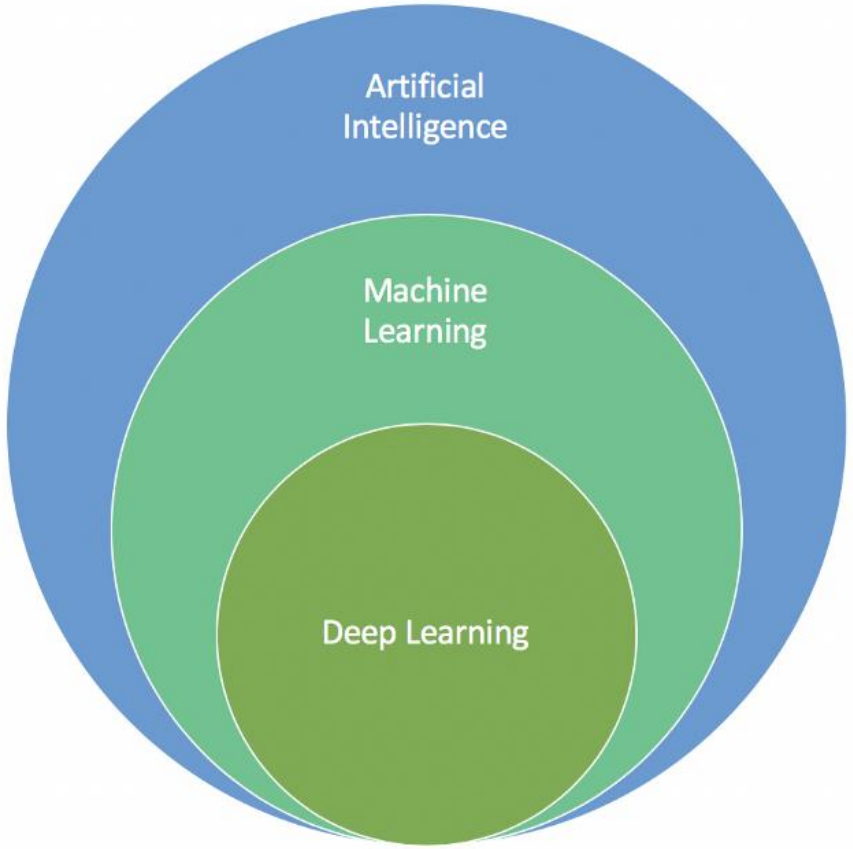
Swiss Institute of Bioinformatics

- Artificial Intelligence
  - programmed by humans to perform human activities
  - computer systems (AI systems): units of “thinking machines”
    - General AI Systems can solve problems intelligently
    - Narrow AI Systems can perform specific tasks very well

- AI systems
  - Intentionality (make decisions from data, predetermined responses)
  - Intelligence (intelligent decision making)
  - Adaptability (refine their decision-making capabilities to improve the outcome)

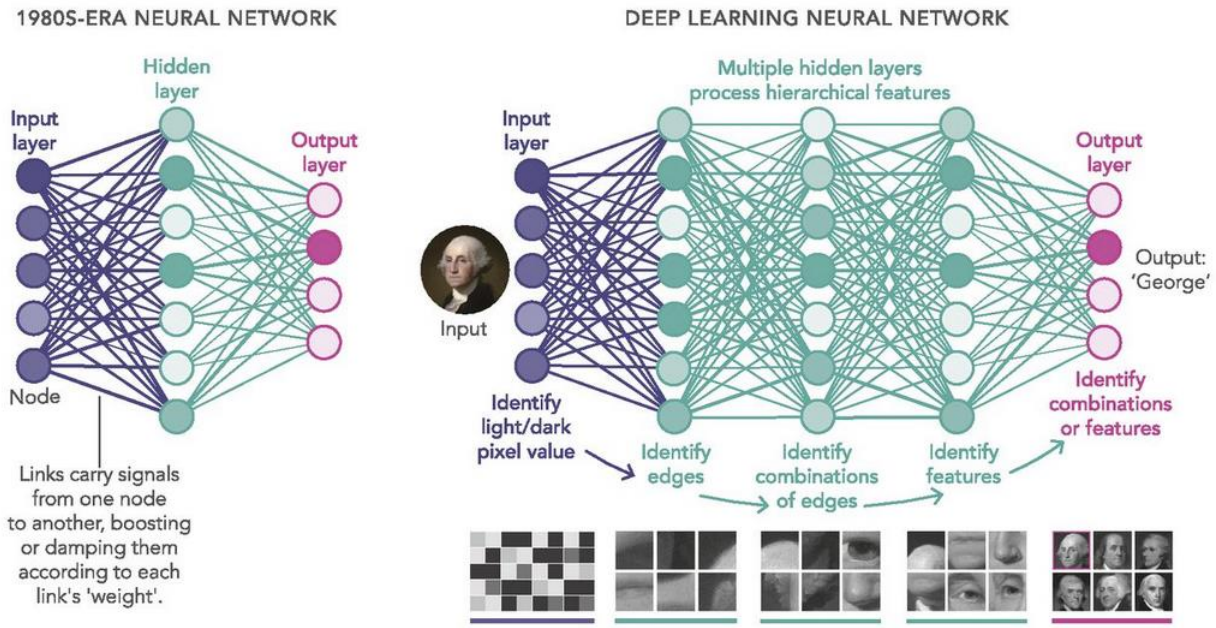
Some definitions

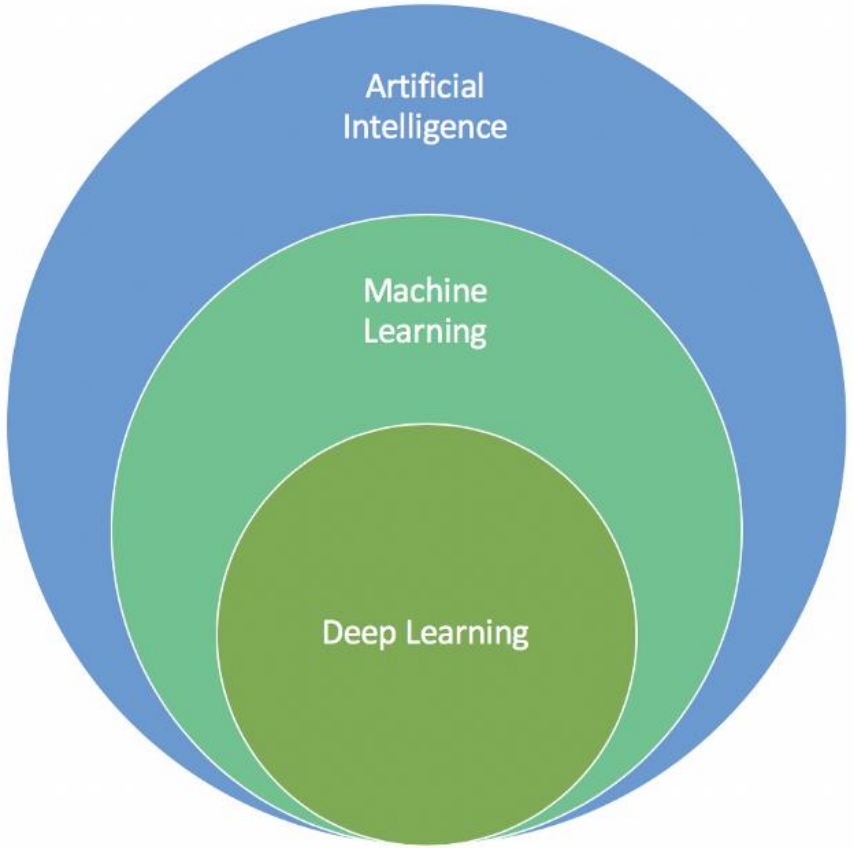
- Machine learning
  - application (or subset) of AI that allows machines to learn from data using general purpose algorithms
  - variety of algorithms to iteratively learn, describe and improve data in order to predict better outcomes
  - extensive use of statistical techniques



Some definitions

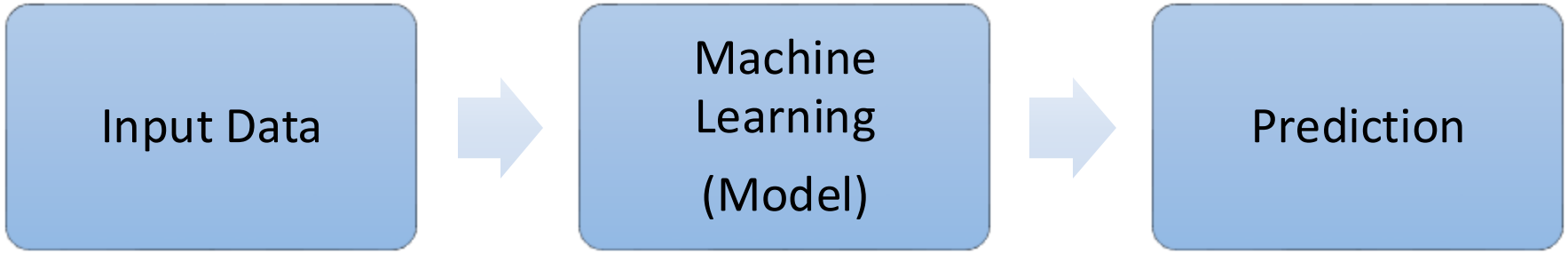
- Deep Learning
  - subset of Machine Learning
  - use artificial neural networks
  - analyzes data with a logical structure similar to how a human would draw conclusions





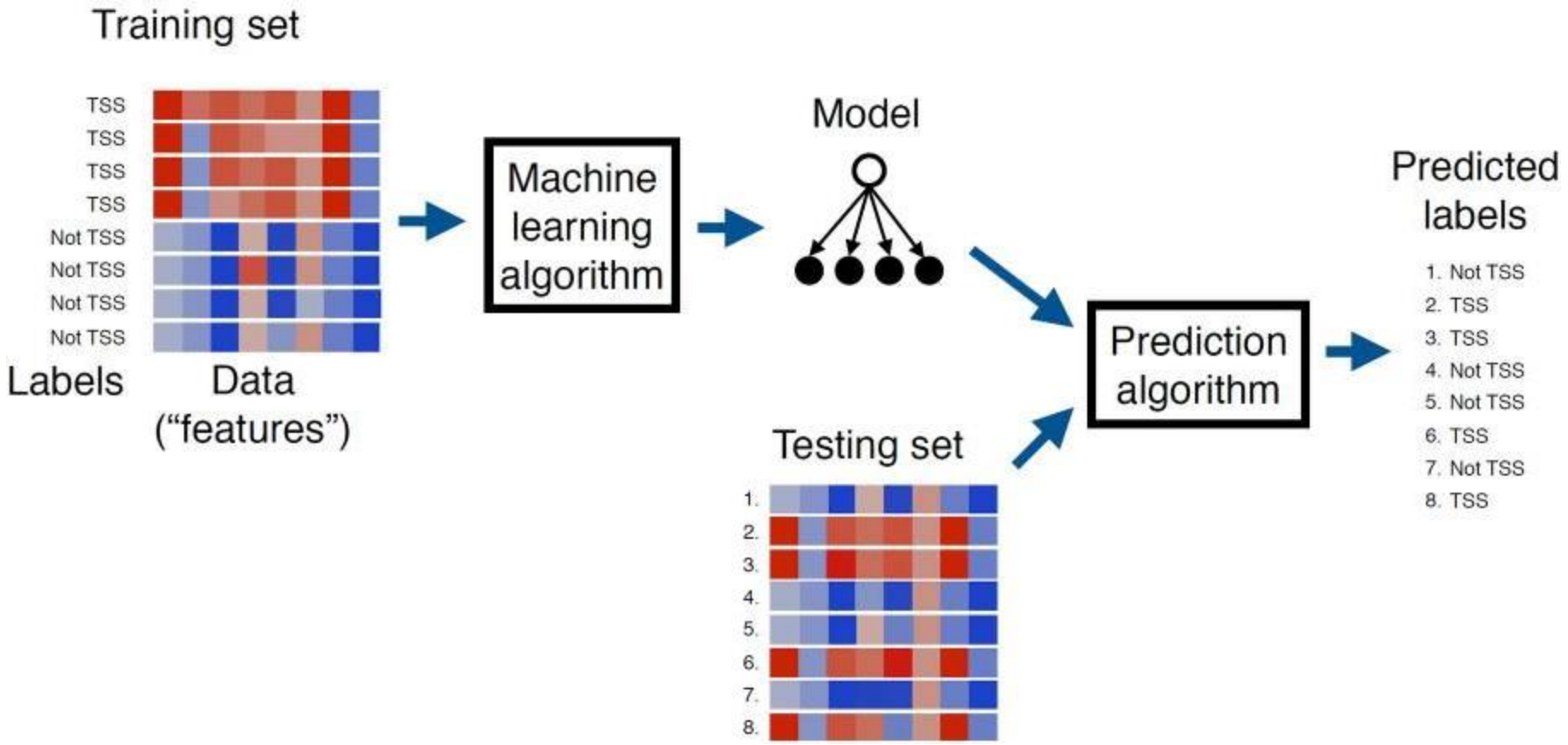
Isabelle Dupanloup, BCF

Machine learning



- Learning begins with observations or data
- The system looks for patterns in data and makes better decisions in the future based on the examples that we provide
- The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly

Machine learning in omics



https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5204302/

Machine learning

Swiss Institute of Bioinformatics

Any machine learning problem can be represented with the following three concepts:

- We will have to learn to solve a task T.
  - For example, perform genome annotation.
- We will need some experience E to learn to perform the task. Usually, experience is represented through a dataset.
  - For the gene prediction, experience comes as a set of DNA sequences provided as input to a learning procedure, along with binary labels indicating whether each sequence is centered on a TSS or not. The learning algorithm produces a model which can then be subsequently used, in conjunction with a prediction algorithm, to assign predicted labels to unlabeled test sequences.
- We will need a measure of performance P to know how well we are solving the task and also to know whether after doing some modifications, our results are improving or getting worse.
  - The percentage of genes that our gene prediction model is correctly classifying as genes could be P for our gene prediction task.

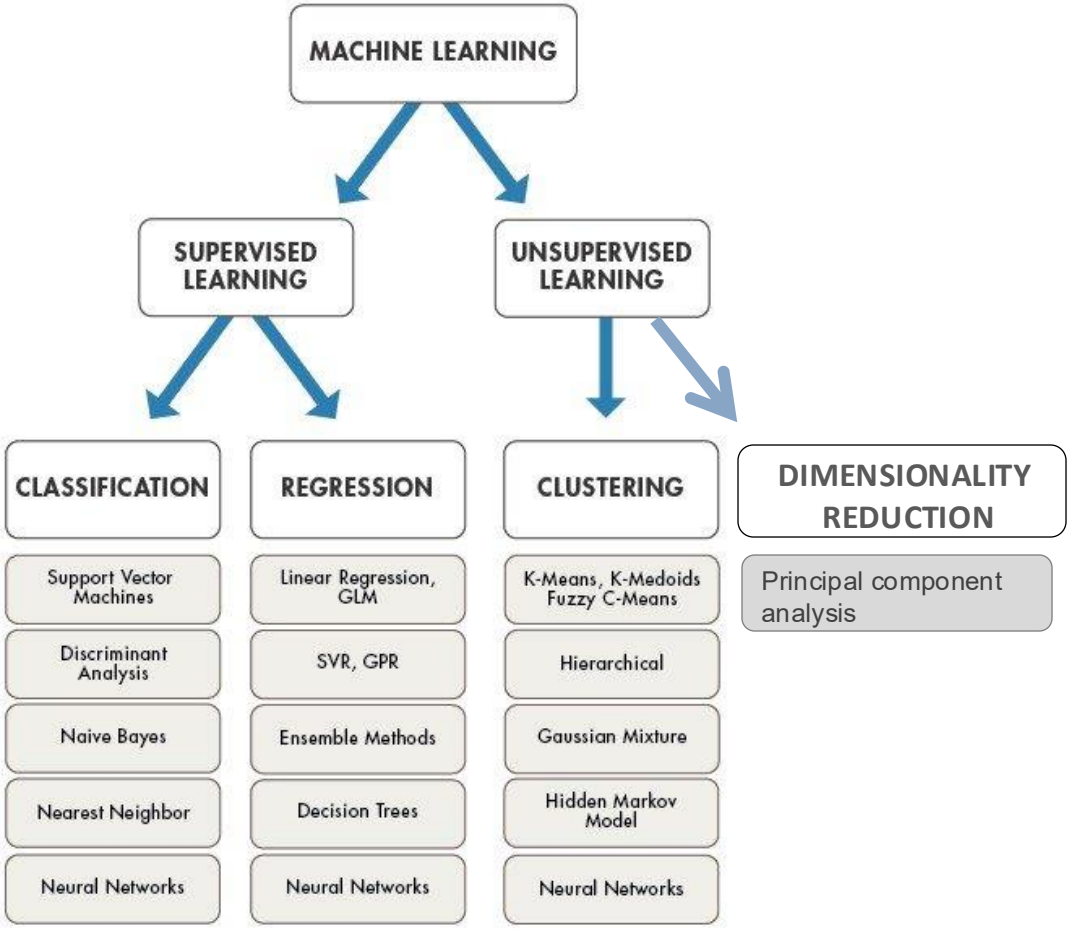
Machine learning in omics

Swiss Institute of Bioinformatics

Input data	Task
DNA sequence	Identify transcription start sites, splice sites, exons, etc.
DNA sequence	Identify TF binding sites
DNA sequence	Identify genes
Gene expression	Predict regulatory relationships
Gene expression data	Identify biomarkers for a disease
Histone and TF ChIP-seq data	Partition and label the genome with chromatin state annotation
DNA sequence + gene expression + …	Predict gene function
DNA sequence + histone mods + …	Predict gene expression
DNA sequence + histone mods + …	Predict variant deleteriousness
Sequence variants + gene expression + …	Predict disease phenotype or prognosis

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5204302/

Machine learning taxonomy



Isabelle Dupanloup, BCF

Supervised versus unsupervised learning

Swiss Institute of Bioinformatics

- Supervised learning
  - Machine learning algorithm that is trained on labeled examples and used to predict the label of unlabeled examples.
  - Prediction limited to input labels
- Unsupervised learning
  - Machine learning algorithm that does not require labels, such as a clustering algorithm.
  - Prediction of new labels
- Semi-supervised learning
- Reinforcement Learning

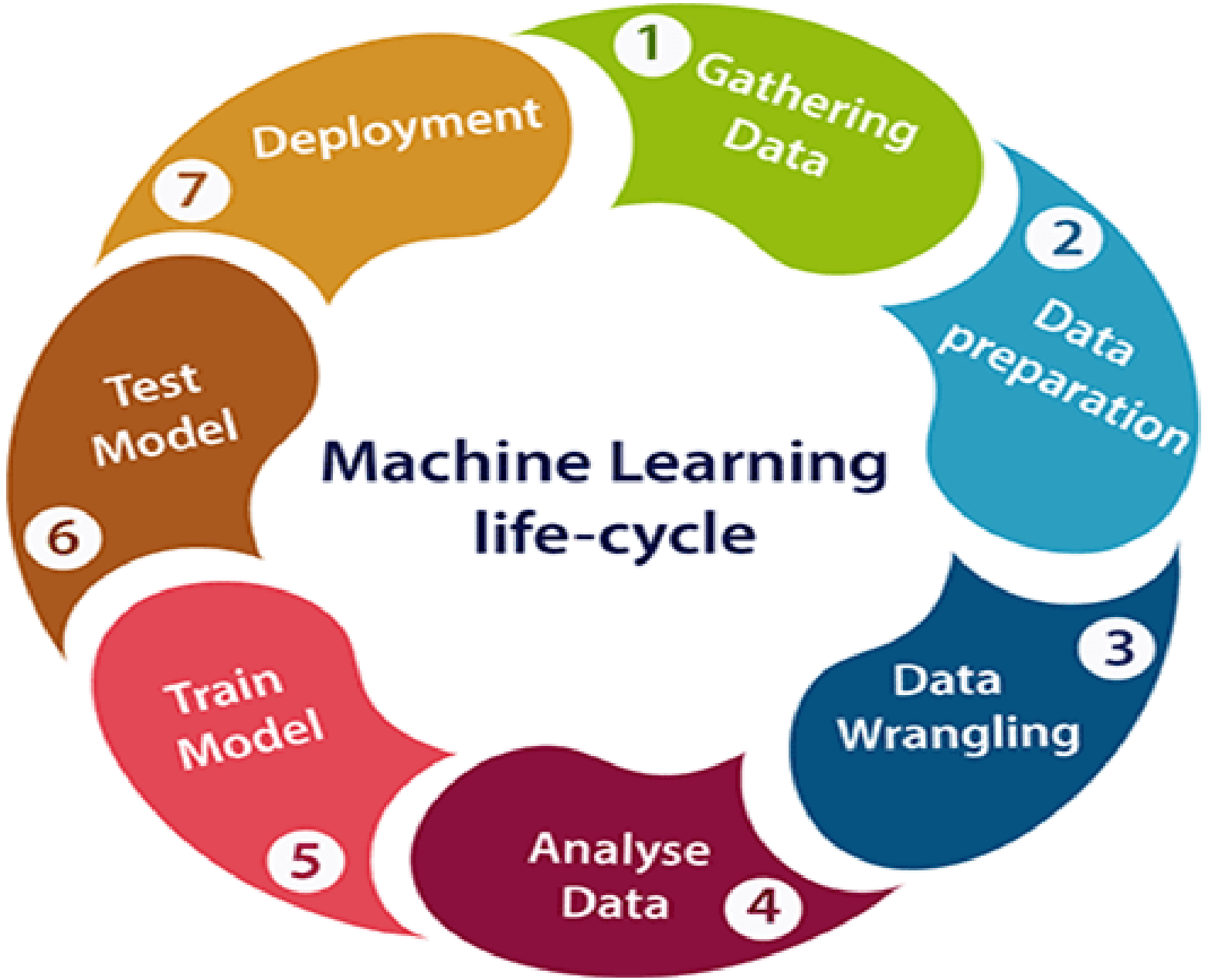
Planning a ML Project

- Understand the problem
- Define the problem and project scope
- Define success metrics
- Find support and manage expectations
- Discuss with colleagues and peers
- Know when to stop project

- Understand the data
- Do Biblio and study existing solutions and algorithms
- Use benchmark models
- Use replicate runs if model contains random component
- Only use models you understand
- Plan for development, testing and production stages

Ezequiel Ortiz Recalde @ Towards Data Science

https://www.javatpoint.com/machine-learning-life-cycle

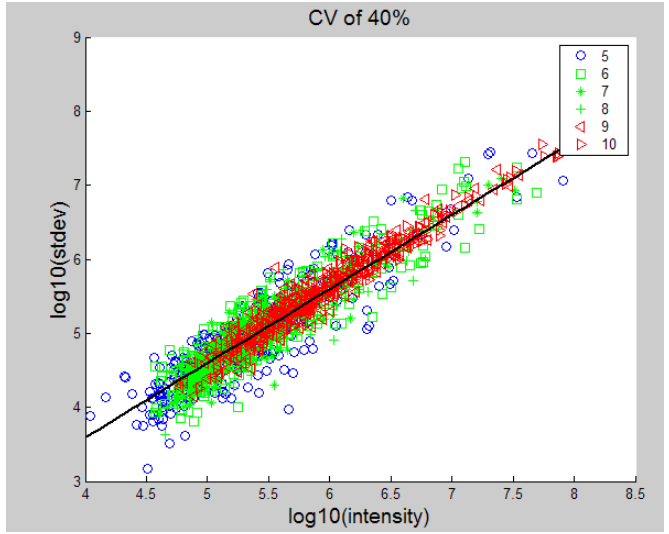


Understand Your Data

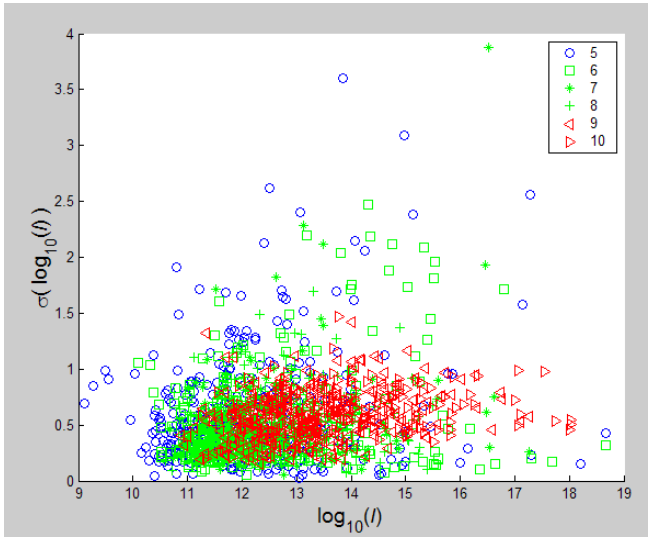
- Are data consistent?
- Are there outliers or wrongly annotated datapoints?
- Are there biases, especially if data comes frome different labs or measurement devices?
- Is there anough data for what I want to do?
- What can I do to handle very large data?

Variance Stabilization

- In Omics data the standard deviation σ of a signal often grows with increasing signal value.
- This can detrimental for some ML techniques that assume that each feature value has the same error (e.g. k-means)
- If the dependency is linear a simple log-transform can stabilize the variance
- For more complex situations check Huber et al. Bioinformatics 2002
- Removing batch effects and confounding factors



$$\sigma = \alpha \cdot y \Rightarrow \log_{10}(\sigma) = \log_{10}(y) + \log_{10}(\alpha)$$



$$\sigma(\log_{10}(y)) = const$$

Missing Values (MV)

Swiss Institute of Bioinformatics

- Omics data (especially proteomics) often have missing values, which can be caused by leaky measurement or data processing
- There are two main strategies to deal with MVs: discard or impute them
- MVs can occur at random or depend on feature values
  - For missing at random we can only replace or impute them by the best guess, the global non-missing mean (Donders et al, J. Clin. Epi., 2006)
  - In the latter case, we can try to predict the MVs using the feature values by EM, KNN, SVD, or regression technique (Troyanskaya et al, Bioinformatics, 2001)

Feature Selection

Swiss Institute of Bioinformatics

- Many ML methods work better when the number of noisy features is small (curse of dimensionality)
- Feature selection is a way to detect uninformative features and remove them before applying the ML method.
- sklearn offers various feature selection methods in sklearn.feature_selection

Turning categorical values into numerical ones

Many ML tools (eg. neural nets, logistic regression) can only handle numerical features and we need to change categorical values to numbers

- Simple encoding: replace categories by numbers
  - Introduces unnatural order in categories
- One hot encoding:
  - Create a new feature f_c for each category c
  - For the category $\hat{c}$ of a data point set: $f_c = \begin{cases} 1 \; if \; c = \hat{c} \\ 0 \; if \; c \neq \hat{c} \end{cases}$
  - Blows up number of features if there are many categories

$$\begin{matrix} & a & b \\ \begin{pmatrix} a \\ a \\ b \end{pmatrix} & \rightarrow & \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix}$$

- Target encoding:
  - $f_c = \mathrm{E}(y|c_i = c)$, y target value
  - For supervised learning or regression

Reproducibility in machine learning

Swiss Institute of Bioinformatics

- Reproducibility: a ML research finding is reproducible if the code and data used to obtain the finding are available and the data is correctly analyzed
- Computational reproducibility: other researchers should be able to replicate the results of a model, given the full details on data, code and conditions.
- Credibility of ML relies on reproducibility
- Irreproducible results can easily arise in ML (Kapoor and Narayanan, arXiv, 2022)
- The most prominent source of error is ‘data leakage’ (see below), when information from the test data is included in the training data.
- See reproducibility workshop from Princeton University

Understanding the data

Swiss Institute of Bioinformatics

- Association
  - Discover correlations between features and target value (histograms, scatterplots)
  - Discover correlations between features (pair plots)
- Visualization in 2D or 3D
  - High dimensional data often lives on low dimensional manifolds. We can try to capture this low dimensional structure by finding suitable ‘projections’ of the data onto 2D or 3D spaces (PCA, ICA, MDS, tSNE, UMAP)
- Clustering
  - Discover groupings inside the input data (example: cluster DNA sequences into functional groups).

Some Notations:

M features or conditions

Data matrix: $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_N)^{\boldsymbol{T}} = \begin{pmatrix} x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NM} \end{pmatrix}$ *N items* $\quad N \times M$

Zero centering: $m_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}\,, x_{ij} \longmapsto x_{ij} - m_j$ *Columns have 0 means*

Principal components analysis (PCA)

Swiss Institute of Bioinformatics

- Mathematical algorithm that reduces the dimensionality of the data while retaining most of the variation in the data set
- Identifies orthogonal directions, called principal components, along which the variation in the data is maximal
- By using a few components, each sample can be represented by relatively few numbers instead of thousands of variables.
- Samples can then be plotted, making it possible to visually assess similarities and differences between samples and determine whether samples can be grouped.

Principal Component Analysis (PCA)

Swiss Institute of Bioinformatics

We try to find a direction $\boldsymbol{w}$ along which the data has the largest variance:

$$\boldsymbol{w} = \underset{\|\boldsymbol{w}\|=1}{\operatorname{argmax}} \sum_{i=1}^{N} (\boldsymbol{x}_i \cdot \boldsymbol{w})^2$$

$$\sum_{i=1}^{N} (\boldsymbol{x}_i \cdot \boldsymbol{w})^2 = \|\boldsymbol{X}\boldsymbol{w}\|^2 = \boldsymbol{w}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$

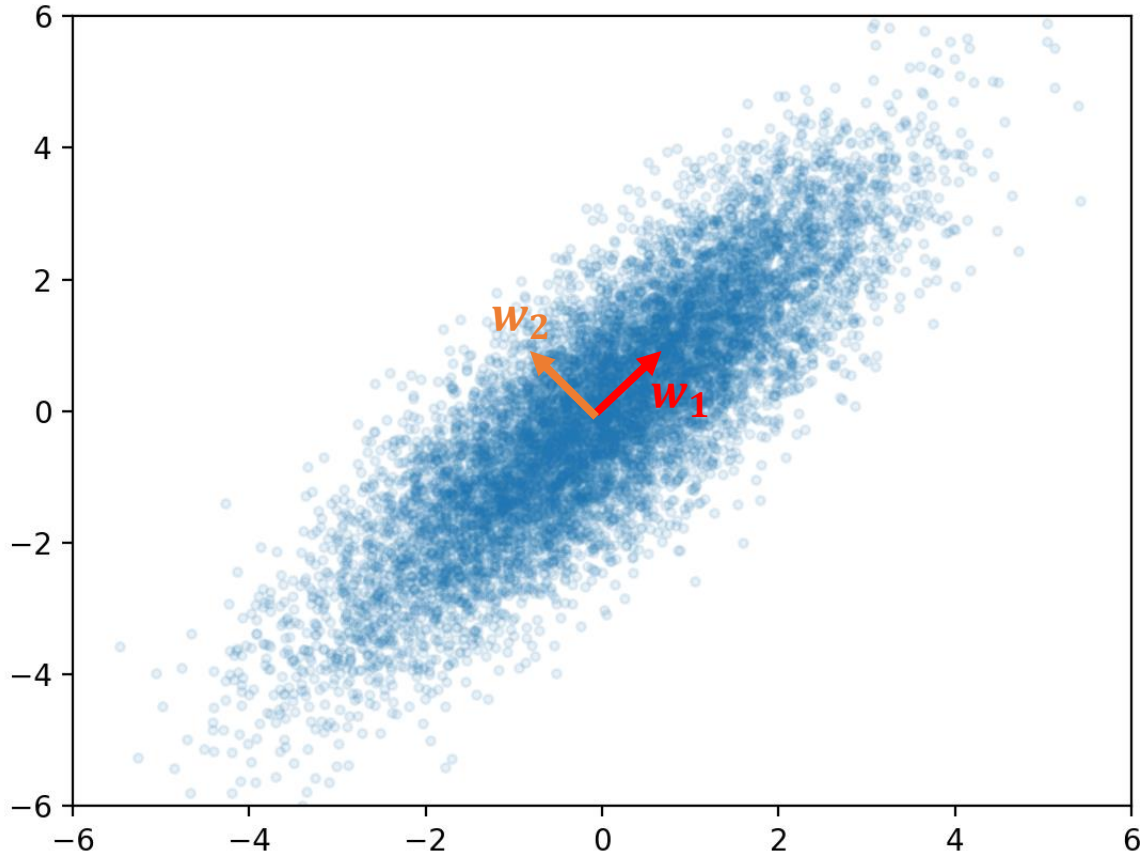
$$\frac{\partial}{\partial \boldsymbol{w}}\left(\boldsymbol{w}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} + \lambda \boldsymbol{w} \cdot \boldsymbol{w}\right) = 0 \Longrightarrow \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \lambda \boldsymbol{w}$$

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{W}\boldsymbol{D}\boldsymbol{W}^T, \boldsymbol{W} = (\boldsymbol{w_1} \cdots \boldsymbol{w_M})$$

$$\{\boldsymbol{w_1} \cdots \boldsymbol{w_M}\} = \text{Eigenvectors or PCs of cov. matrix } \boldsymbol{X}^T\boldsymbol{X}$$

$$\lambda_j = \sigma_j^2,\ \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_M, D = \begin{pmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \sigma_j^2 & \vdots \\ 0 & \cdots & \sigma_M^2 \end{pmatrix}$$

Zero centered $\boldsymbol{X}$



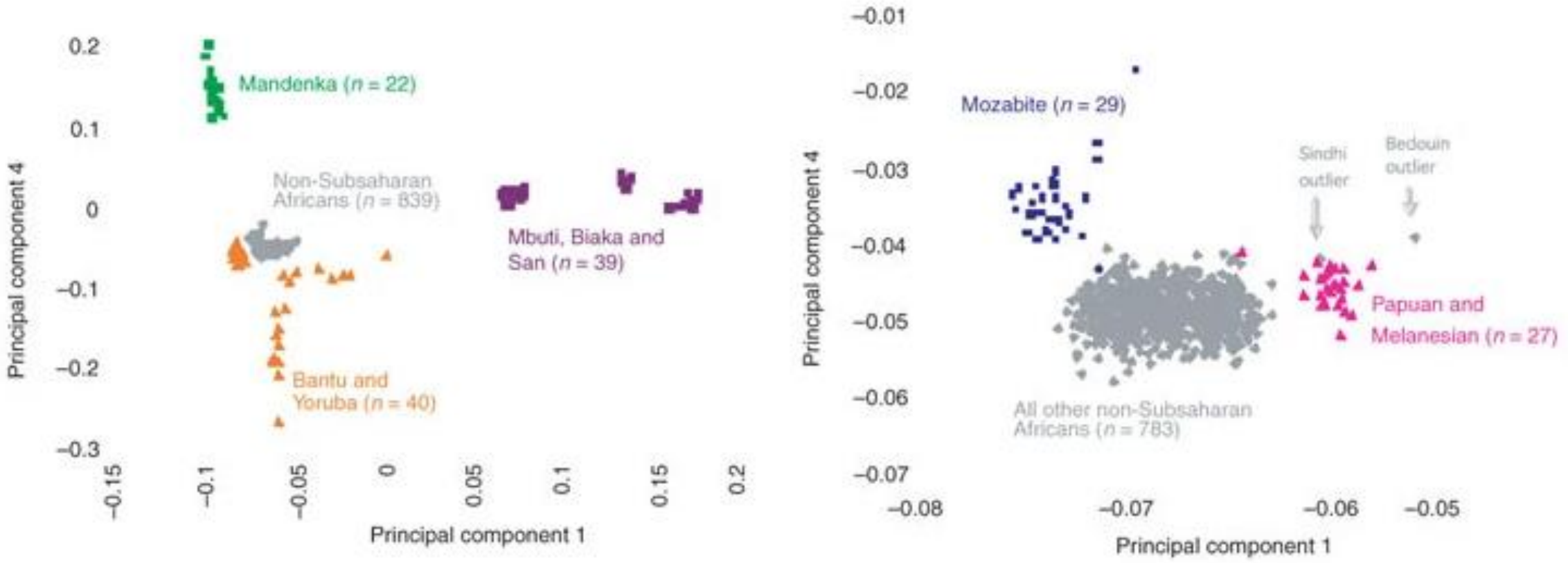
Principal Component Analysis (PCA)

Swiss Institute of Bioinformatics

- Dimension reduction and visualization
- Decorrelate or whiten signals
- Data compression

Principal components analysis (PCA)

Swiss Institute of Bioinformatics



Reich et al. (2008) Nature Genetics.

Independent Component Analysis (ICA)

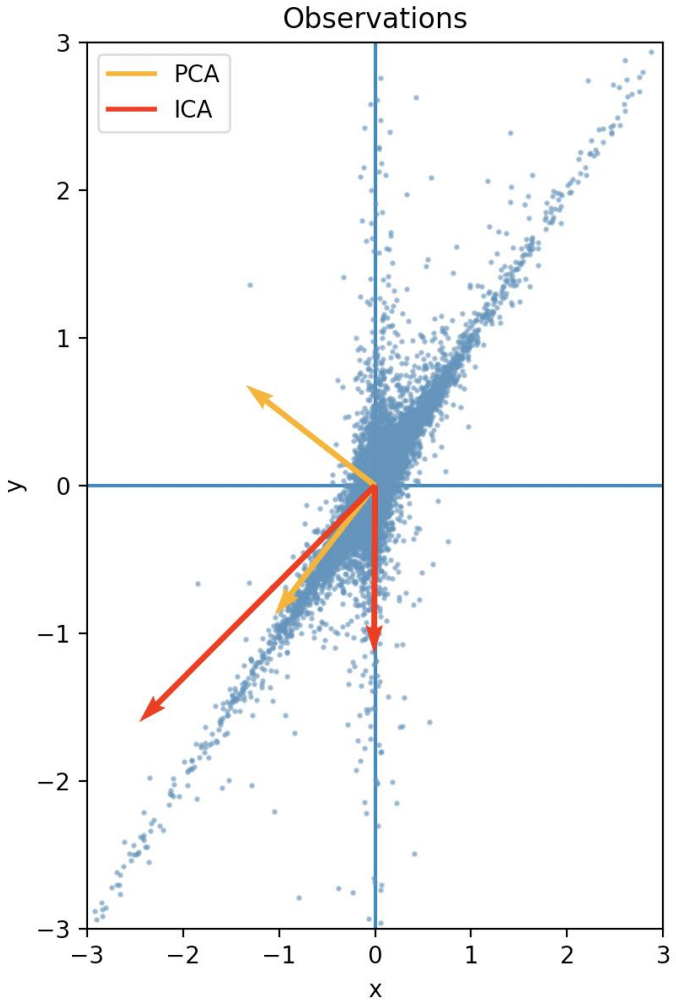
Swiss Institute of Bioinformatics

We try to find direction $\boldsymbol{w}$ along which the data deviates the most from a Gaussian distribution:

$$\boldsymbol{w} = \underset{\|\boldsymbol{w}\|=1}{\operatorname{argmax}} DG(t_1, t_2, \cdots, t_N), t_i = \boldsymbol{x}_i \cdot \boldsymbol{w}$$

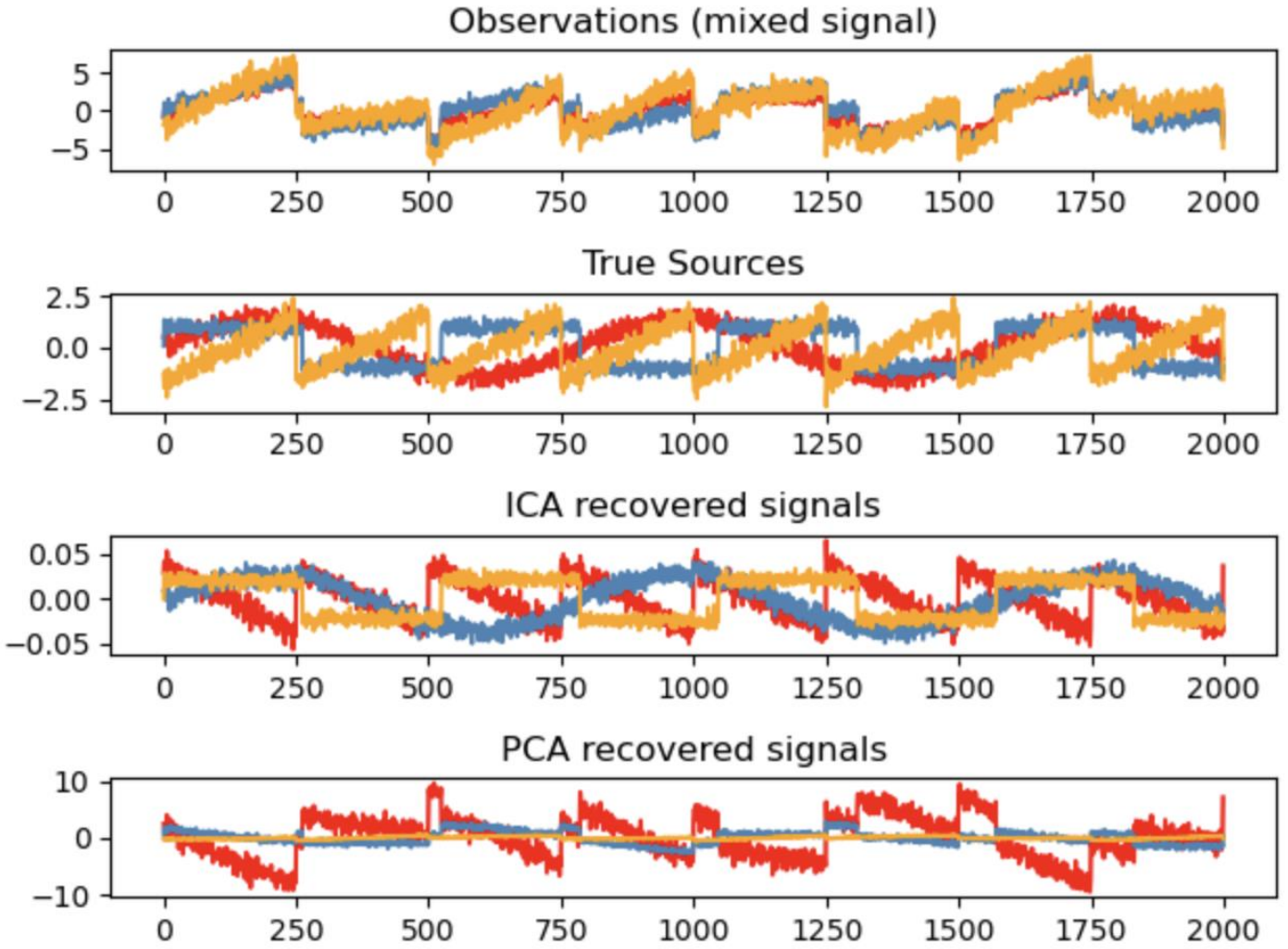
where $DG(t_1, t_2, \cdots, t_N)$ is a function that measures how far away $\{t_1, t_2, \cdots, t_N\}$ are from a Gaussian distribution.

Empirical functions DG were found that work well in practice and are fast to calculate (Hyvärinen & Oja, Neural Networks, 2000).



Independent Component Analysis (ICA)

Swiss Institute of Bioinformatics



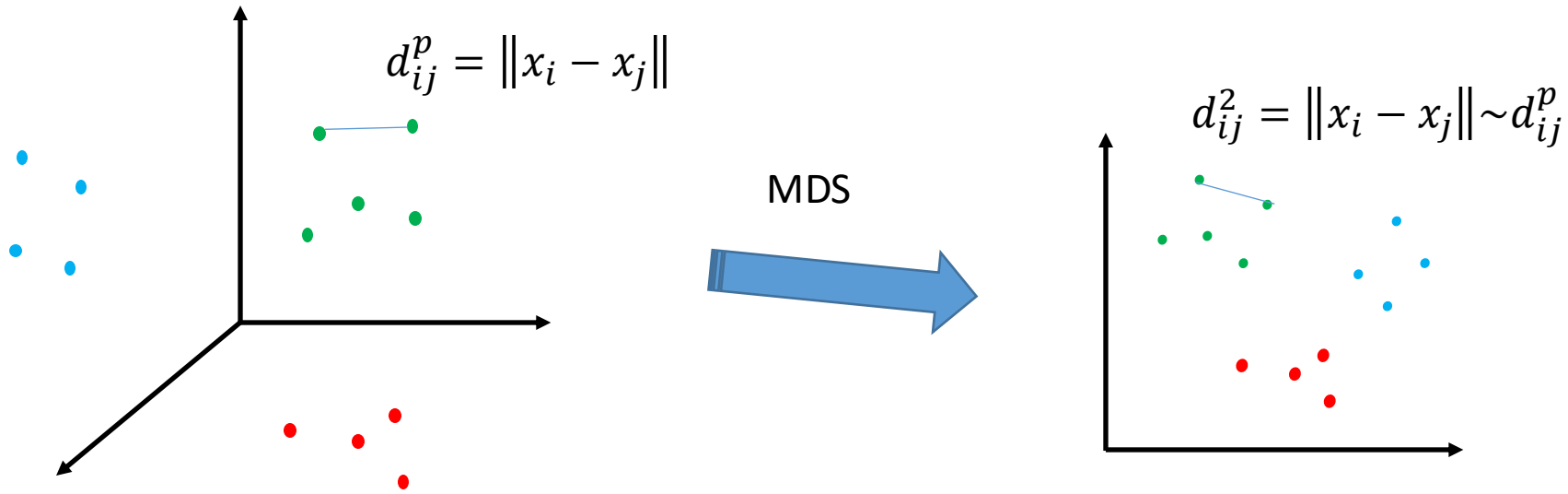
Multidimensional scaling (MDS)

Swiss Institute of Bioinformatics

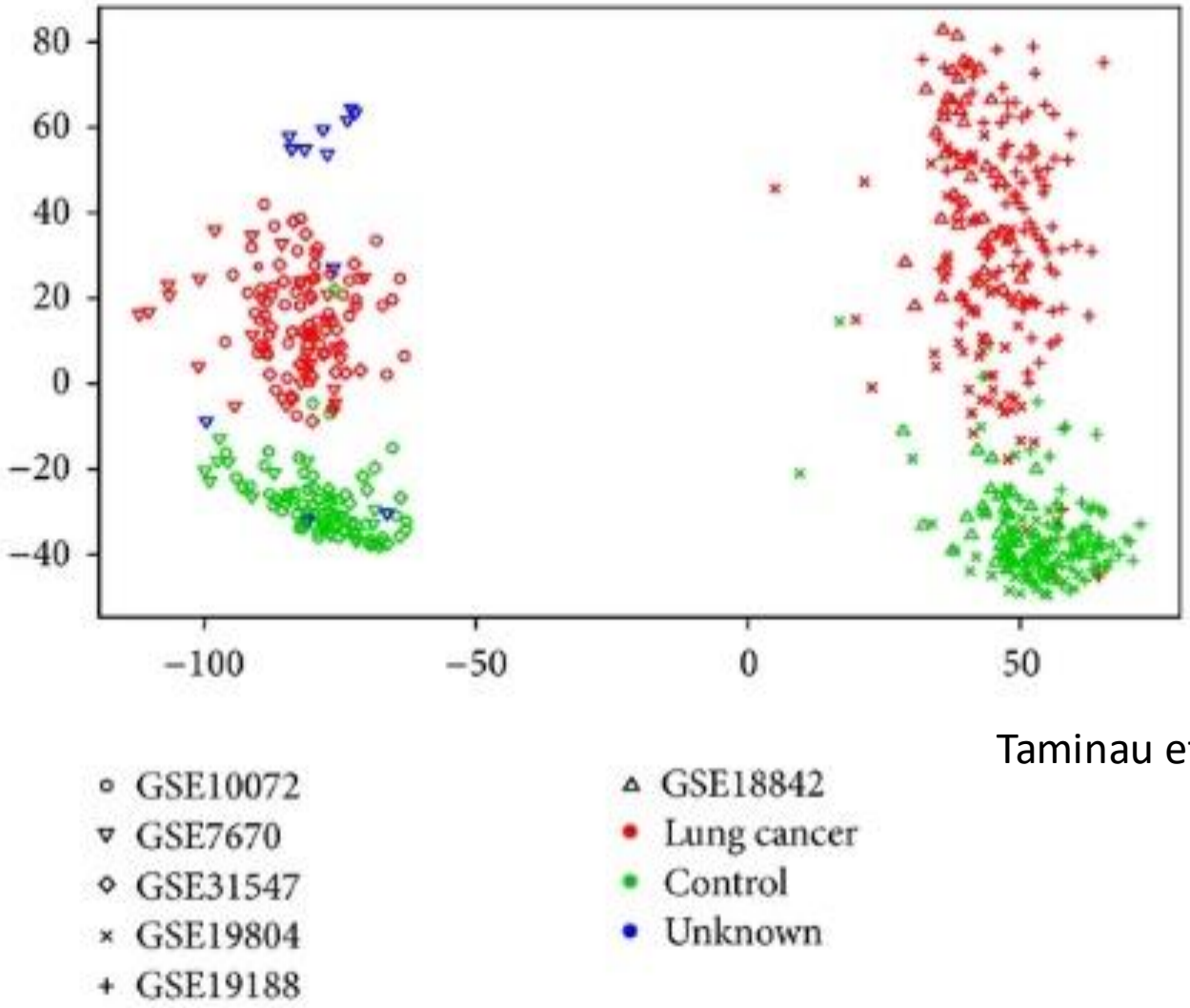
- visual representation of distances between sets of objects
- objects that are more similar (or have smaller distances) are closer together on the MDS than objects that are less similar (or have larger distances)
- MDS can also serve as a dimension reduction technique for high-dimensional data

Multidimensional scaling (MDS)

- Step by step
  1. Assign points to arbitrary coordinates in p-dimensional space.
  2. Compute euclidean distances among all pairs of points
  3. Compare the similarity matrix with the original input matrix by evaluating the stress function. Stress is a goodness-of-fit measure, based on differences between predicted and actual distances.
  4. Adjust coordinates of each point in the direction that reduces stress.
  5. Repeat steps 2 through 4 until stress can't be minimized further.



Multidimensional scaling (MDS)



Taminau et al. (2014) ISRN Bioinform.

t-distributed Stochastic Neighbor Embedding (t-SNE)

Swiss Institute of Bioinformatics

- unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data
- In concept similar to MDS, but with improved distance metrics
- t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance
- maps a set of high-dimensional points to two dimensions, such that close neighbours remain close and distant points remain distant

t-distributed Stochastic Neighbor Embedding (t-SNE)

Swiss Institute of Bioinformatics

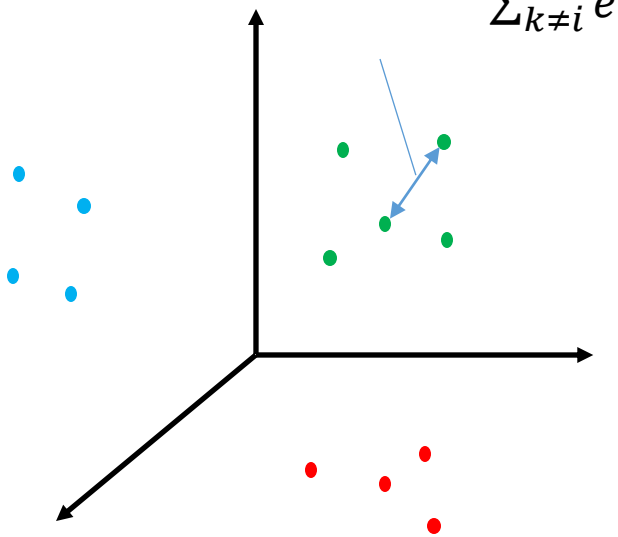
- Informally, the algorithm places all points on the 2D plane, initially at random positions, and lets them interact as if they were physical particles.
- The interaction is governed by two laws: first, all points are repelled from each other; second, each point is attracted to its nearest neighbours
- The most important parameter of t-SNE, called perplexity, determines how many of its nearest neighbours each point is attracted to.

t-distributed Stochastic Neighborhood Embedding (t-SNE)

Swiss Institute of Bioinformatics

Choose σ_i to have user defined perplexity $P_i = 2^{-\sum_j p_{j|i} \log_2(p_{j|i})}$

$$p_{i|j} = \frac{e^{-\|x_i - x_j\|^2 / 2\sigma_i}}{\sum_{k \neq i} e^{-\|x_i - x_k\|^2 / 2\sigma_i}}$$

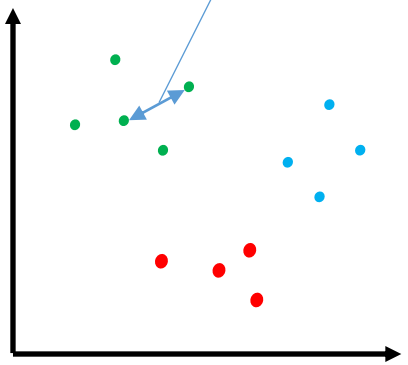


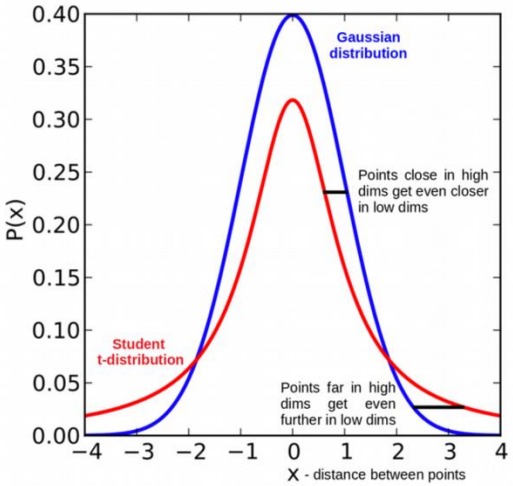
t-SNE

optimize

$$C = \sum_i \sum_j p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}$$

$$q_{i|j} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq i} (1 + \|y_i - y_k\|^2)^{-1}}$$



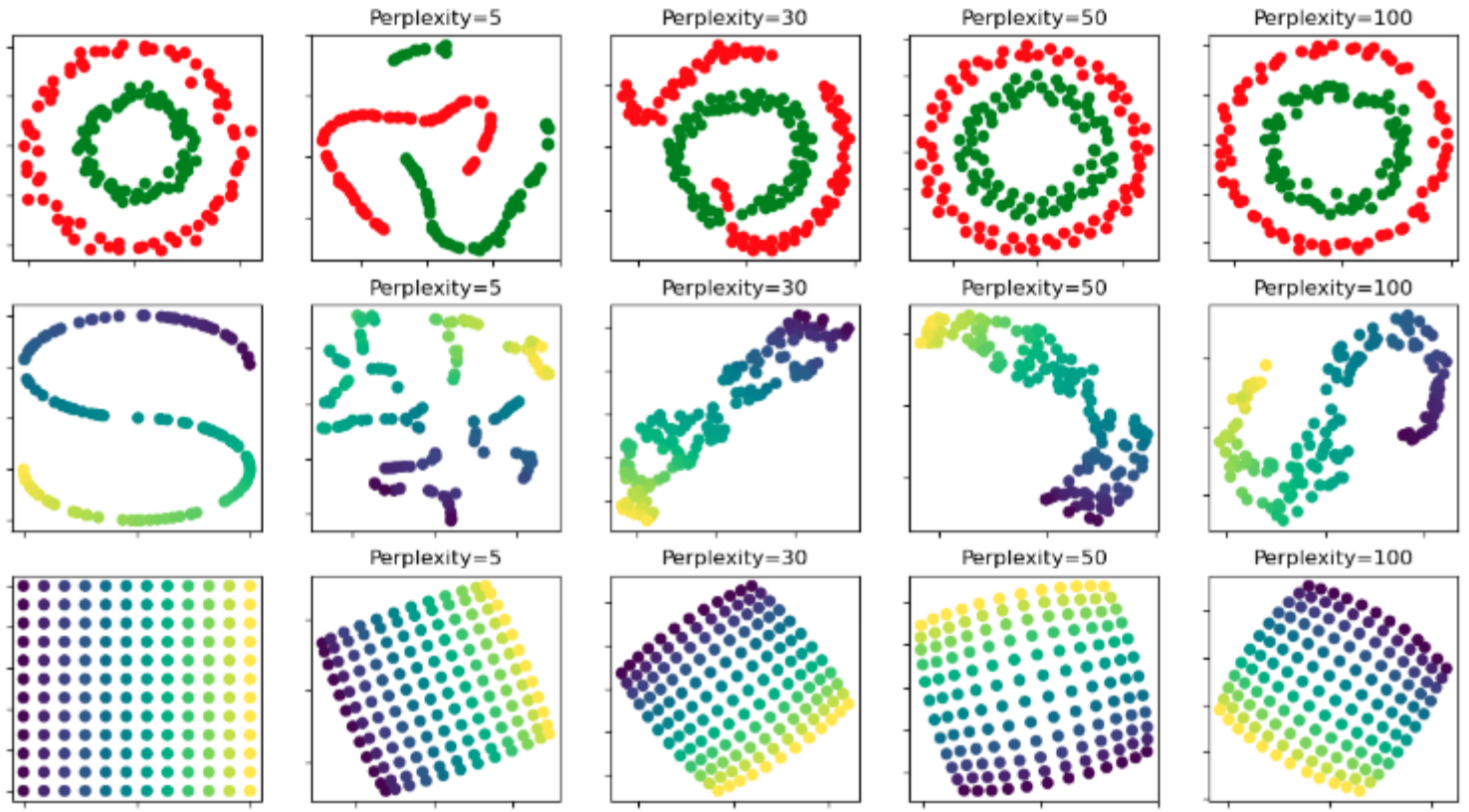


Maaten & Hinton, JMLR, 2008

t-SNE Perplexity

Swiss Institute of Bioinformatics

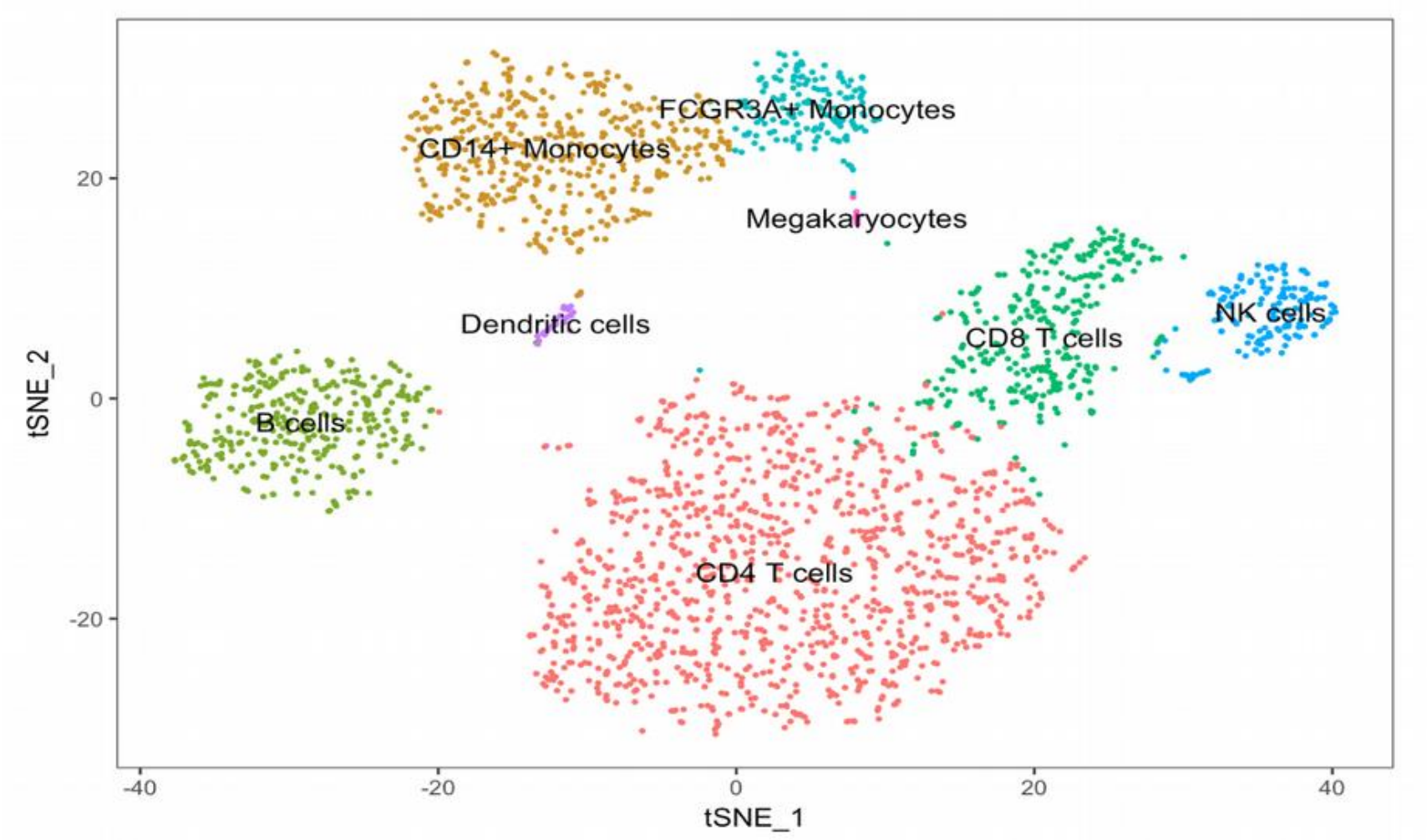
https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html



The larger the perplexity, the more neighbours are considered and the more the global structure is preserved

t-distributed Stochastic Neighbor Embedding (t-SNE)

Swiss Institute of Bioinformatics

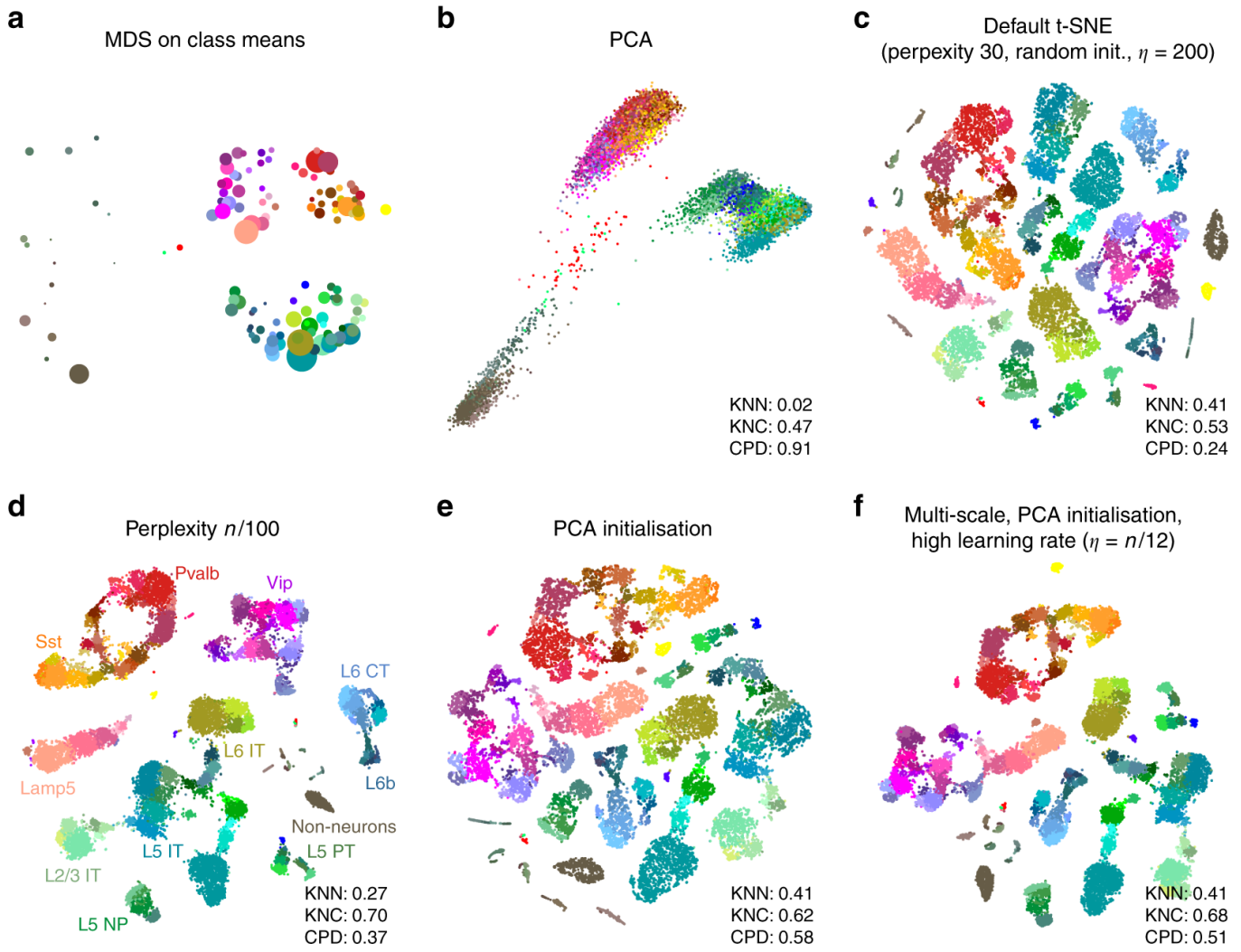


https://towardsdatascience.com/how-to-tune-hyperparameters-of-tsne-7c0596a18868

Isabelle Dupanloup, BCF

t-distributed Stochastic Neighborhood Embedding (t-SNE)

Swiss Institute of Bioinformatics



Single cell analysis

[Kobab & Berens, Nature Comm., 2019]

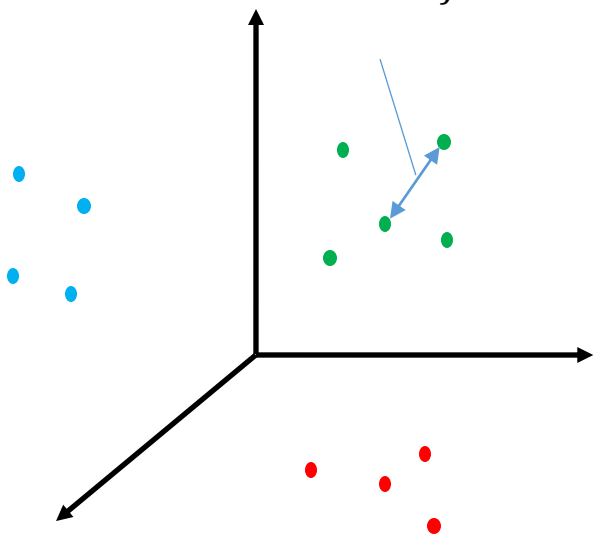
Uniform manifold approximation and projection (UMAP)

Choose σ_i :

$$\sum_{j \in NN(i)} \mu_{ij} = \log_2 n$$

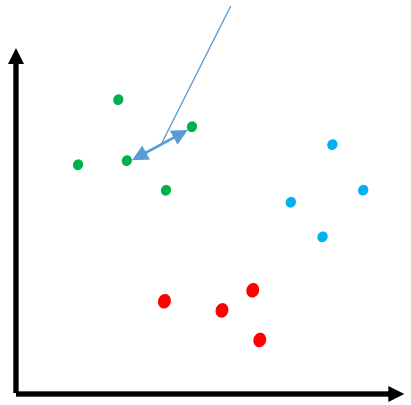
$$\mu_{ij} = e^{-\frac{d(x_i, x_j) - \rho_i}{\sigma_i}}$$

$$v_{ij} = \left(1 + a\|y_i - y_j\|^{2b}\right)^{-1}$$

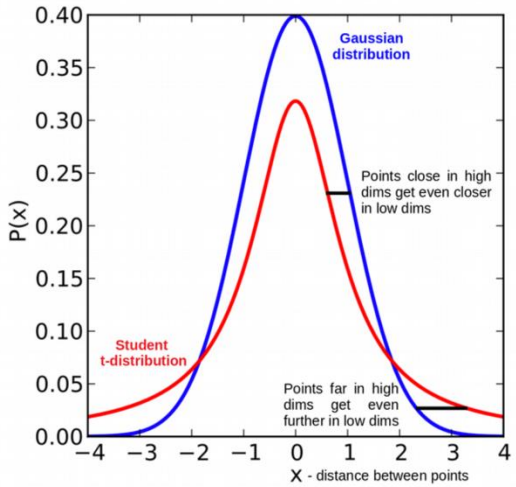


UMAP

Initialize with SpecClust & optimize



$$C = \sum_i \sum_j \mu_{ij} \log \frac{\mu_{ij}}{v_{ij}} + (1 - \mu_{ij}) \log \frac{1 - \mu_{ij}}{1 - v_{ij}}$$

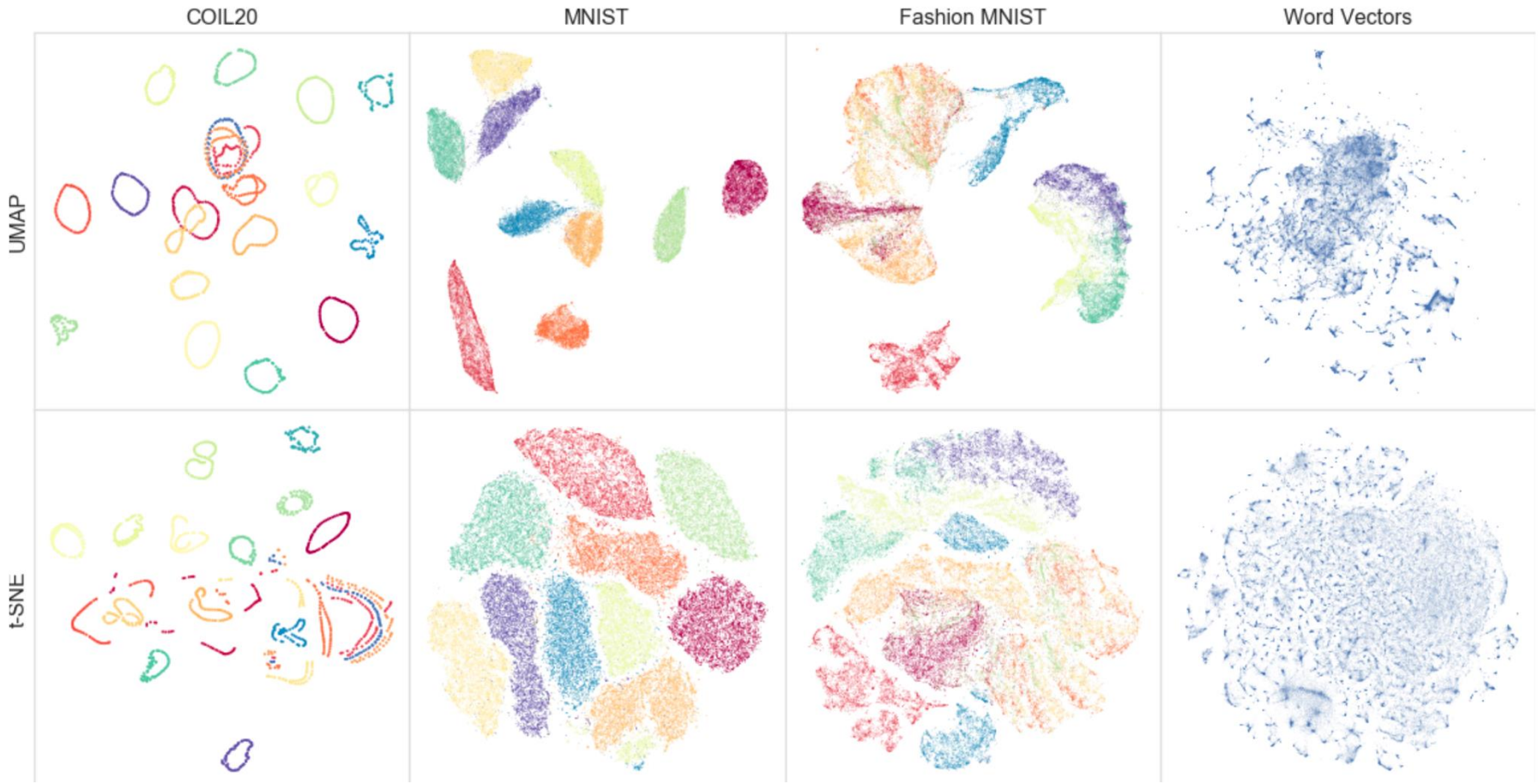


McInnes et al.,arxiv, 2018

SIB Swiss Institute of Bioinformatics

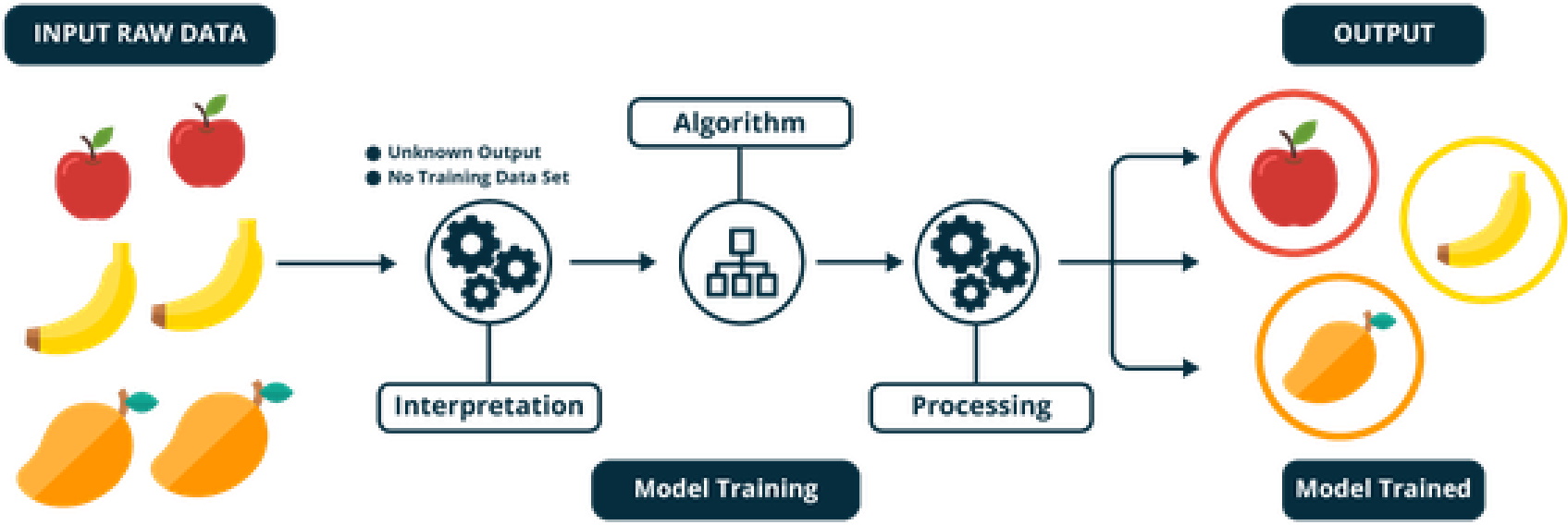
T-SNE versus UMAP

McInnes et al., 2018



[Nikolay Oskolkov's blog]

Unsupervised learning



Unsupervised machine learning algorithms

Swiss Institute of Bioinformatics

- In contrast to supervised machine learning algorithms, they:
  - are applied when the information used to train is neither classified nor labeled.
  - can infer a function to describe a hidden structure from unlabeled data.
  - do not figure out the right output, but explore the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

Clustering

Swiss Institute of Bioinformatics

- Pattern detection: discover hidden patterns in your data
- Compression: replace all cluster members by one representative
- Data augmentation: replace noisy feature vectors by the cluster consensus vector
- Dimensionality reduction: replace a feature vector by cluster similarities
- Outlier detection: detect feature vectors far away from cluster centers
- Semisupervised learning: assign cluster labels to unlabeled cluster members

K-means Clustering

Swiss Institute of Bioinformatics

1. Choose K points from $(\boldsymbol{x}_1, \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_N)$ as initial values for the cluster centers $\boldsymbol{\mu}_k$. These initial points should be spread out over the dataset.

2. Assignment step: assign each $\boldsymbol{x}_i, i = 1 \ldots N$, to the cluster with the closest center:

$$c_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

3. Update step: recalculate new $\boldsymbol{\mu}_k$ based on cluster assignments $C_k = \{i | c_i = k\}$:

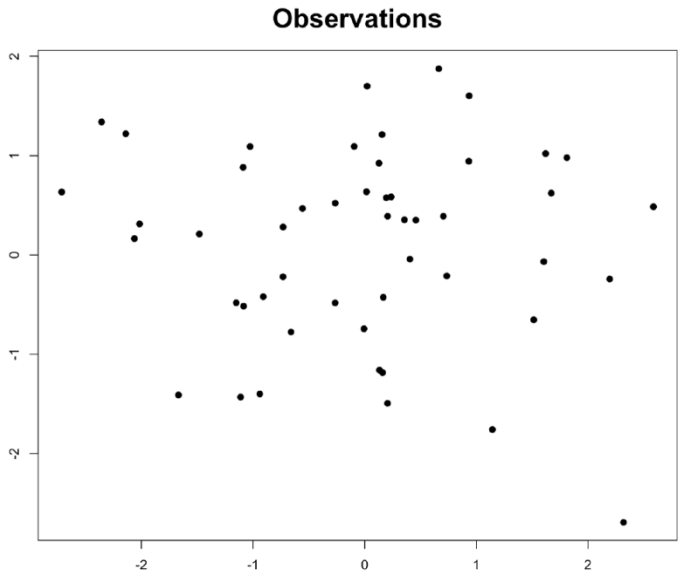
$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \boldsymbol{x}_i$$

4. Iterate steps 2 & 3 until convergence

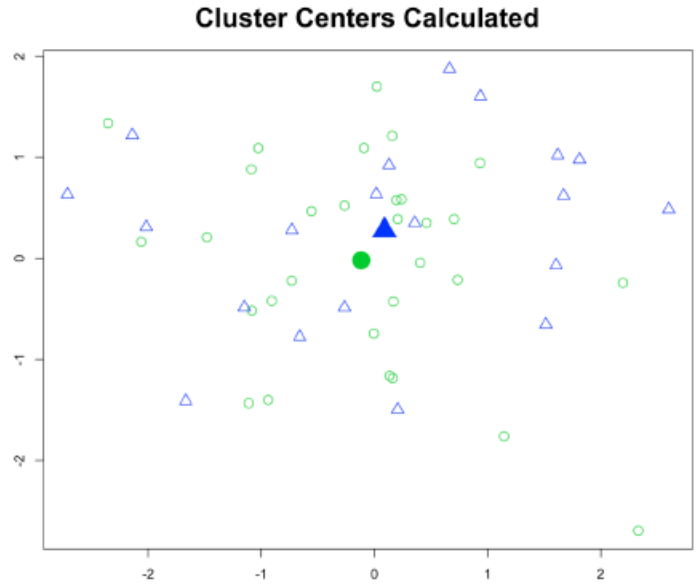
K-means at work

K-means

Swiss Institute of Bioinformatics



K-means initially assigns each point to one of two clusters.

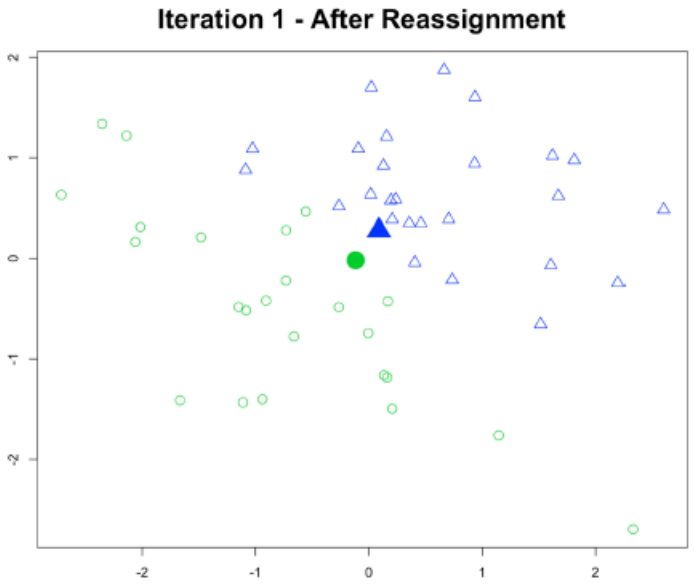


The centers of each subgroup is calculated as the average position of all the points in that group.

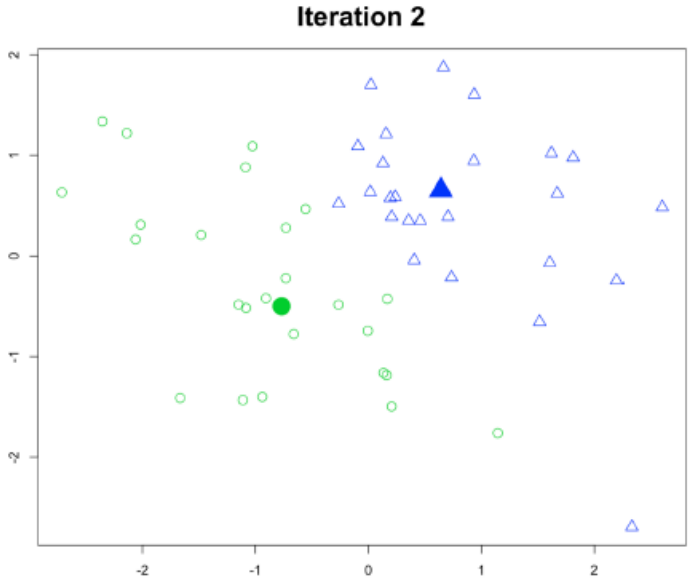
Isabelle Dupanloup, BCF

K-means

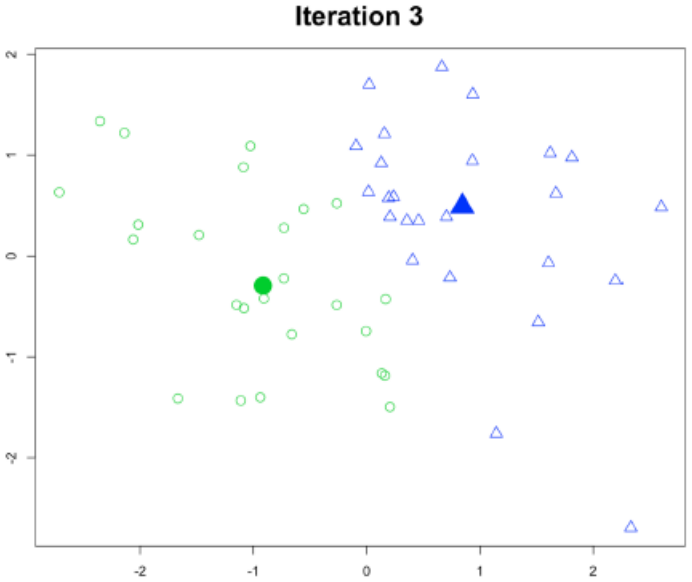
Swiss Institute of Bioinformatics



All the points closest to the solid blue triangle are assigned to the corresponding cluster while those close to the solid green circle are assigned to the other cluster.

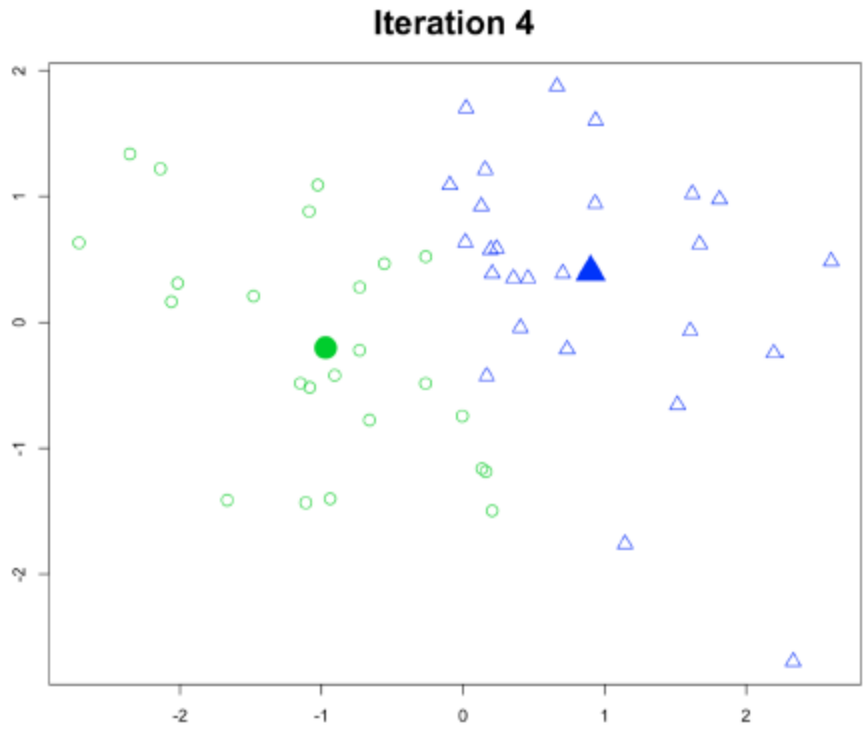


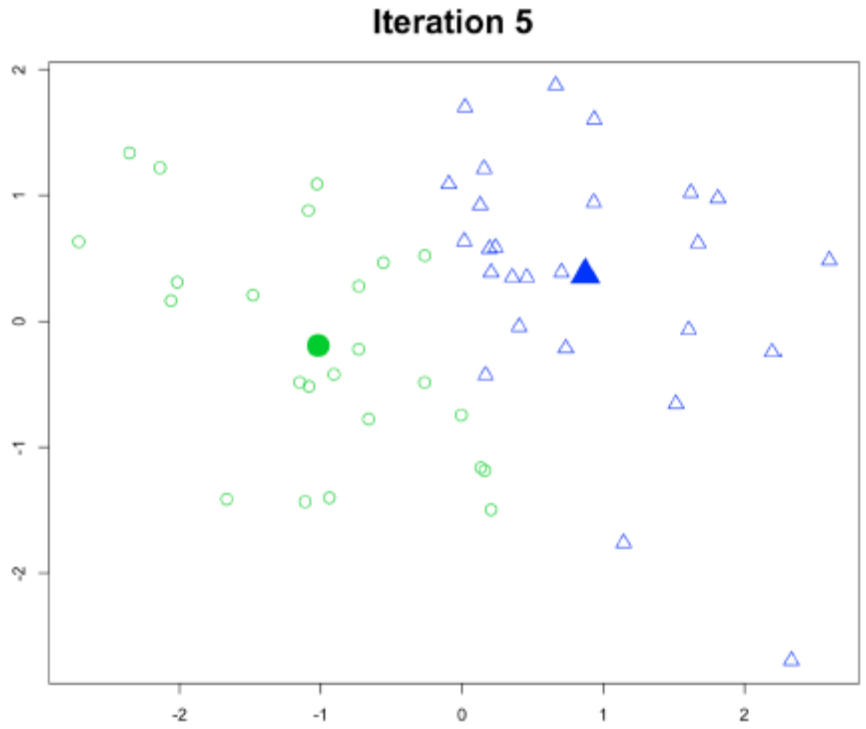
New cluster centers are calculated and each observation reassigned to the cluster of the nearest center.



Isabelle Dupanloup, BCF

K-means





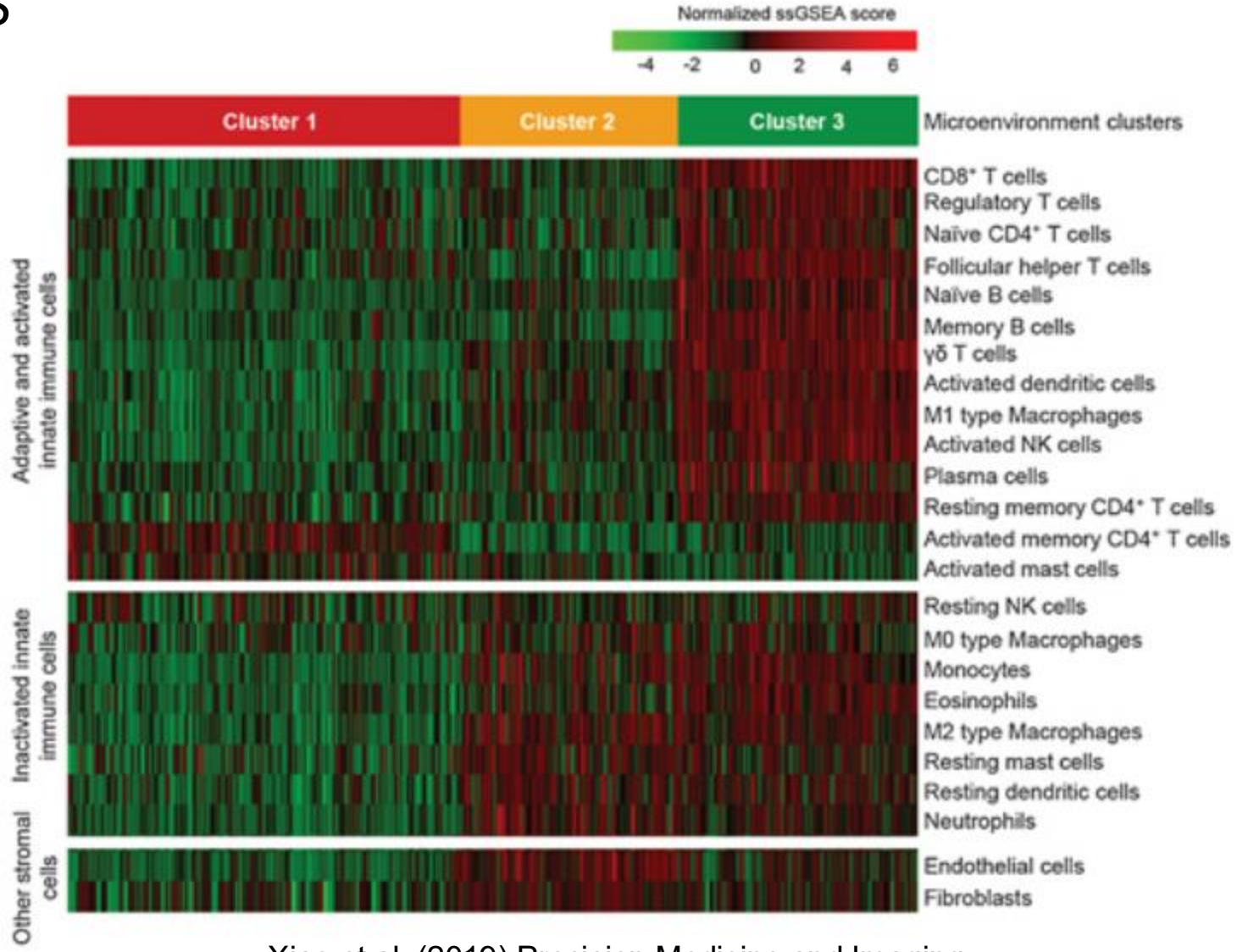
The iterative process stops after a given number of iterations or when the assignment of data points to clusters isn't changing.

K-means

Swiss Institute of Bioinformatics

- Advantage: easy to implement and fast and efficient in terms of computational cost
- Disadvantages:
  - Choice of K
  - Initial seeds have a strong impact on final results
  - Clustering data of varying sizes and density
  - Clustering outliers

K-means



Xiao et al. (2019) Precision Medicine and Imaging

Gaussian Mixtures

Swiss Institute of Bioinformatics

- The data is modelled as a mixtures of n Gaussian distributions:

$$p(\boldsymbol{x}|\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, k = 1 \dots K) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- The parameters $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are determined by the EM algorithm:
  - Expectation step: calculate for each $\boldsymbol{x}_i$ the probability $z_{ik}$to belong to cluster k

$$z_{ik} = \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \Big/ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

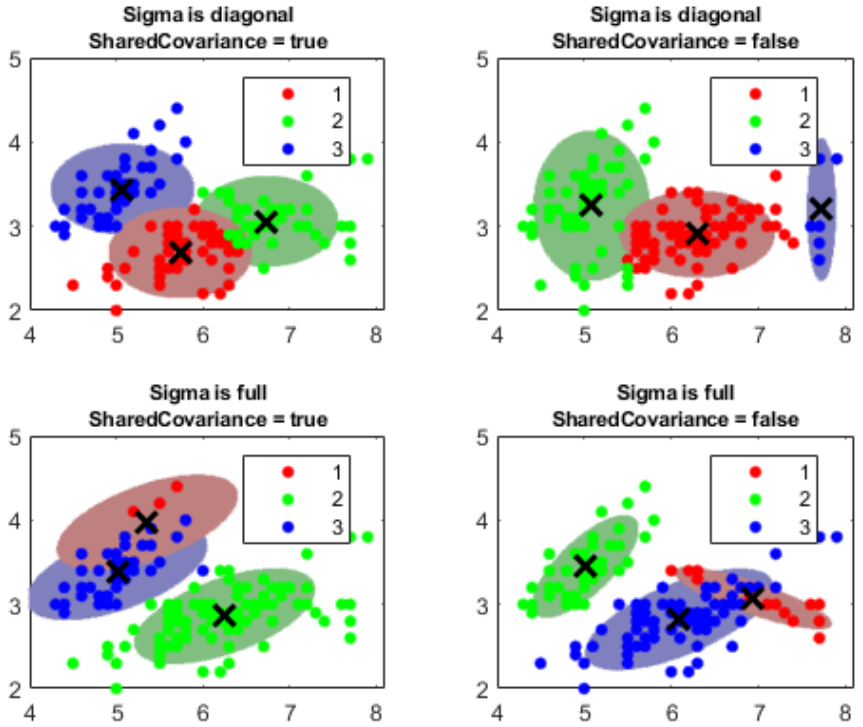
  - Maximization step: recalculate new $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ based on z_{ik} :

$$n_k = \sum_{i=1}^{N} z_{ik}, \pi_k = \frac{n_k}{N}, \boldsymbol{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{N} z_{ik} \boldsymbol{x}_i,$$
$$\boldsymbol{\Sigma}_k = \frac{1}{n_k} \sum_{i=1}^{N} z_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T$$

  - Iterate steps 2 & 3 until convergence
  - $Z_i = \underset{k}{\operatorname{argmax}}\, z_{ik}$: cluster assignement for $\boldsymbol{x}_i$

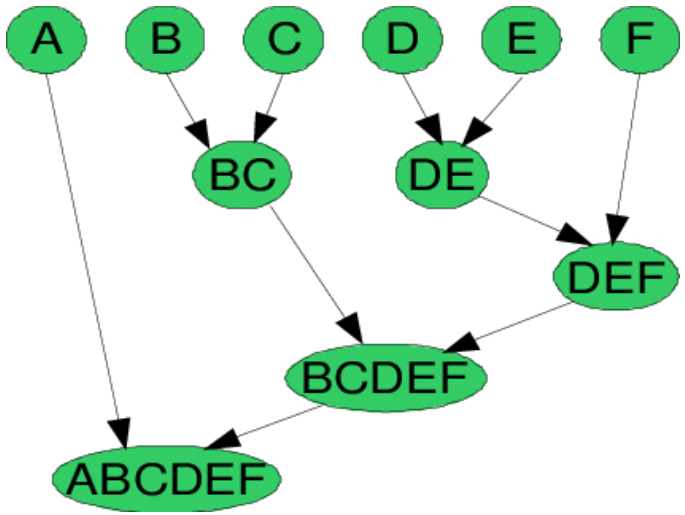
[Frailey & Raftery, The Computer Journal, 1998]

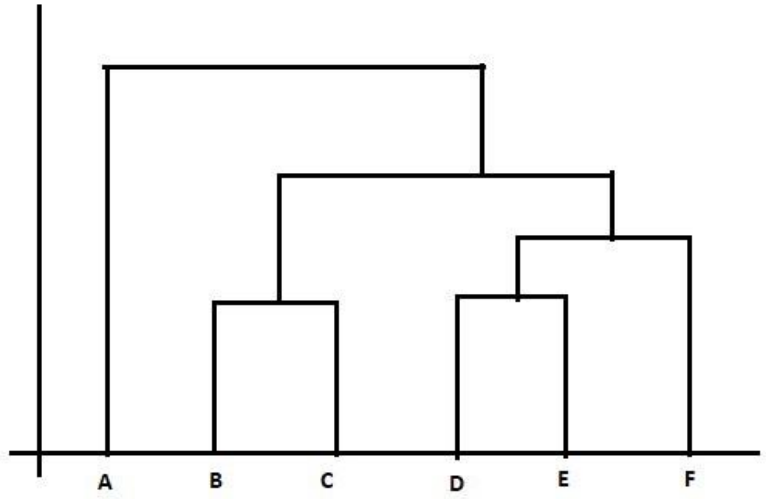
Models $\mathcal{M}$



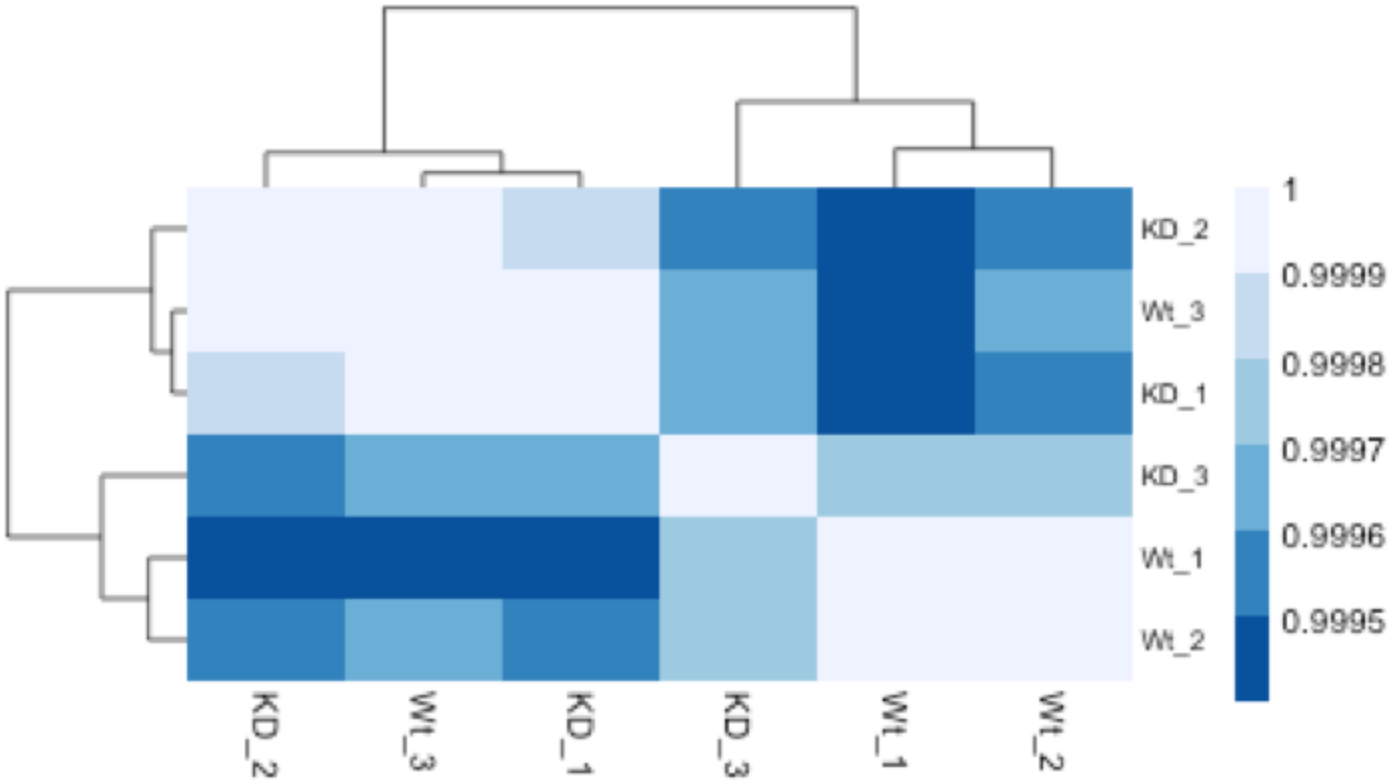
Hierarchical clustering

- 2 types: agglomerative or divisive
- Agglomerative:
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. Repeat: Merge the two closest clusters and update the proximity matrix
  4. Until only a single cluster remains
- Results can be visualized using a dendrogram





Hierarchical clustering



https://hbctraining.github.io/DGE_workshop_salmon/lessons/03_DGE_QC_analysis.html

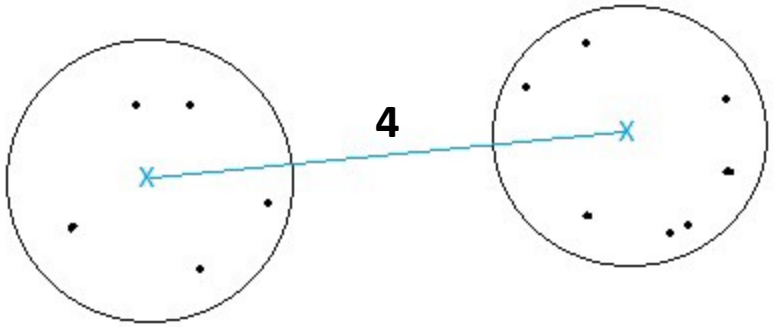
Agglomerative Hierarchical Clustering

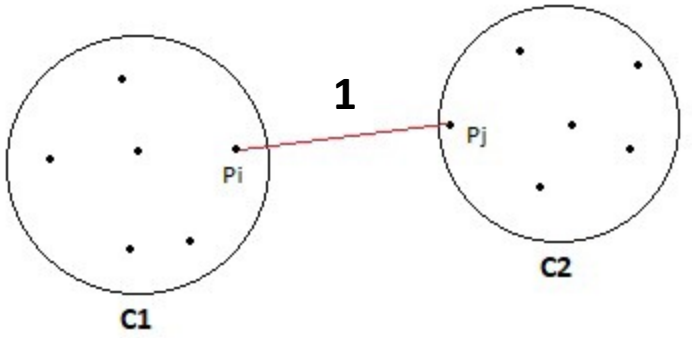
Swiss Institute of Bioinformatics

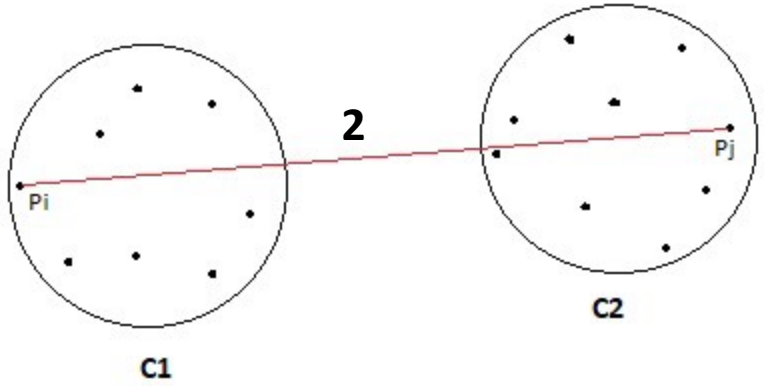
1. Initialize clusters $C_i = \{\boldsymbol{x}_i\},\ i = 1 \dots N, id = \{1,2, \dots N\}$ and calculate distances $d_{ij} = d(C_i, C_j)$
2. Merge closest clusters: $C_i = C_i \cup C_j$, if $d_{ij} = \min_{(l,k)} d_{lk}$, drop j: $id = id \setminus j$ and recalculate $d_{ik} = d(C_i, C_k), k \in id \setminus i$
3. Repeat 2 until $|id| = K$ or $d_{ik} \geq \varepsilon, \forall i, k \in id$

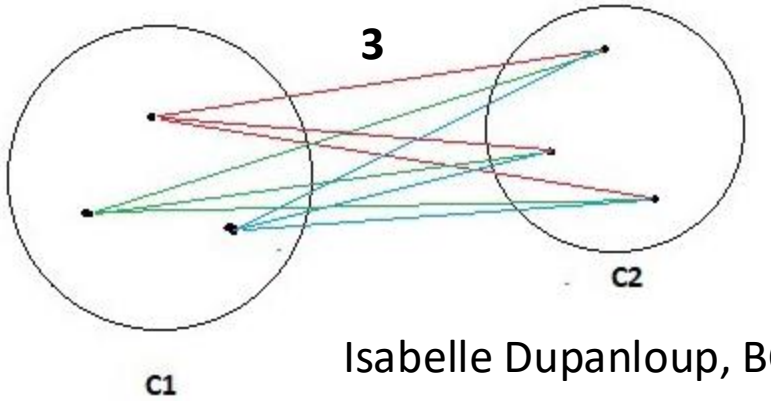
Hierarchical clustering

- Divisive: start with all the data points as a single cluster and in each iteration, we separate the data points from the cluster which are not similar
- Similarity between clusters (choice of linkage)
  1. Min
  2. Max
  3. Group Average
  4. Distance Between Centroids
  5. Ward's Method (similar to Group Average)









Isabelle Dupanloup, BCF

Agglomerative Hierarchical Clustering

Swiss Institute of Bioinformatics

Names	Formula
Maximum or complete-linkage clustering	$\max\{\, d(a,b) : a \in A,\, b \in B \,\}.$
Minimum or single-linkage clustering	$\min\{\, d(a,b) : a \in A,\, b \in B \,\}.$
Unweighted average linkage clustering (or UPGMA)	$\frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum_{a \in A} \sum_{b \in B} d(a,b).$
Weighted average linkage clustering (or WPGMA)	$d(i \cup j, k) = \frac{d(i,k) + d(j,k)}{2}.$
Centroid linkage clustering, or UPGMC	$\lVert c_s - c_t \rVert$ where c_s and c_t are the centroids of clusters *s* and *t*, respectively.
Minimum energy clustering	$\frac{2}{nm} \sum_{i,j=1}^{n,m} \lVert a_i - b_j \rVert_2 - \frac{1}{n^2} \sum_{i,j=1}^{n} \lVert a_i - a_j \rVert_2 - \frac{1}{m^2} \sum_{i,j=1}^{m} \lVert b_i - b_j \rVert_2$

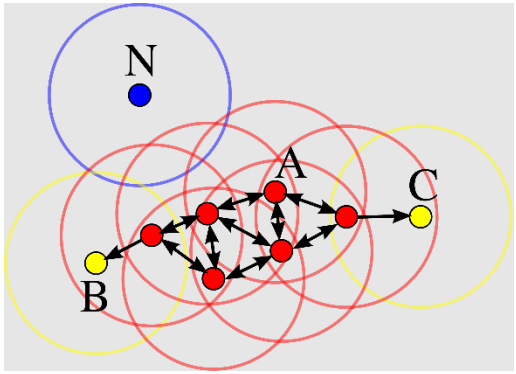
where *d* is the chosen metric. Other linkage criteria include:

- The sum of all intra-cluster variance.
- The increase in variance for the cluster being merged (Ward's criterion).[8]
- The probability that candidate clusters spawn from the same distribution function (V-linkage).
- The product of in-degree and out-degree on a k-nearest-neighbour graph (graph degree linkage).[9]
- The increment of some cluster descriptor (i.e., a quantity defined for measuring the quality of a cluster) after merging two clusters.[

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

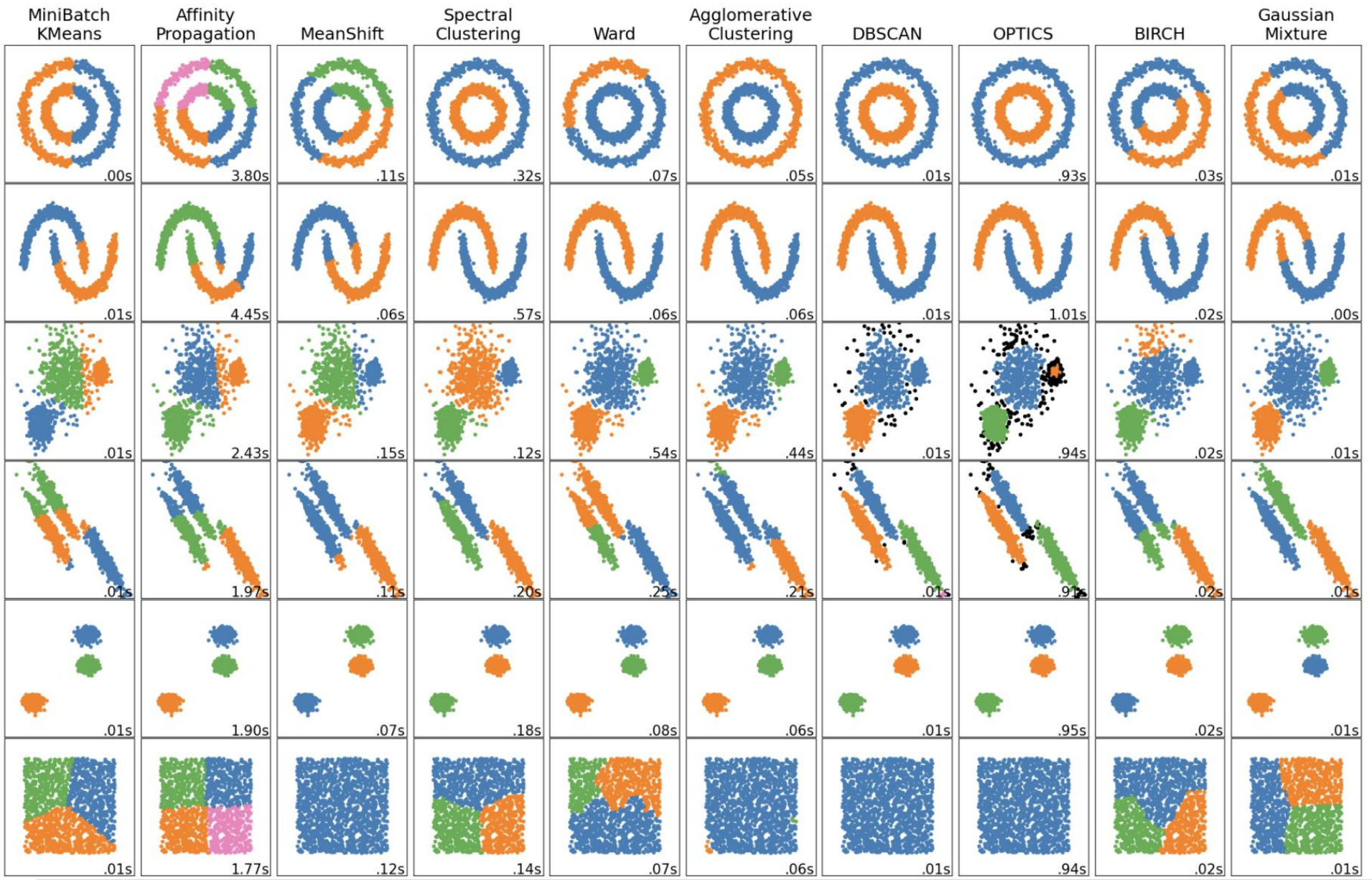
DBSCAN

- DBSCAN uses a minimal distance ε and a minimal number of cluster seed points n as input
- It finds dense regions in the data and uses them as cluster seeds. Then it expands these clusters and adds points if they are within ε distance to a cluster. All points not within reach of a cluster are marked as outliers.
- DBSCAN can deal with clusters of arbitrary shapes.
- DBSCAN requires clusters of similar density
- Distance based clustering are difficult in high dimensions (curse of dimensionality)



Ester et al., Proc KDD, 1996

DBSCAN

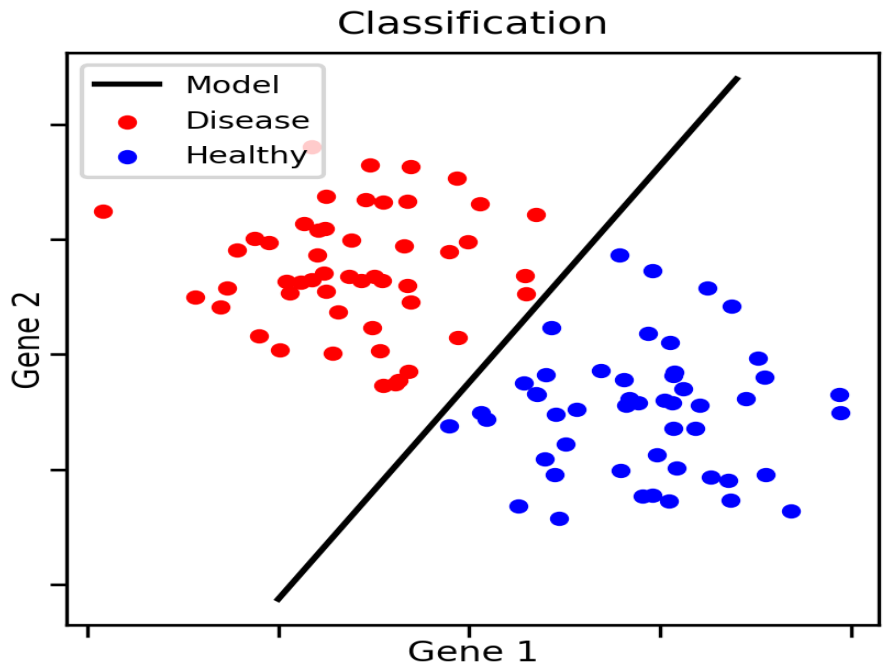


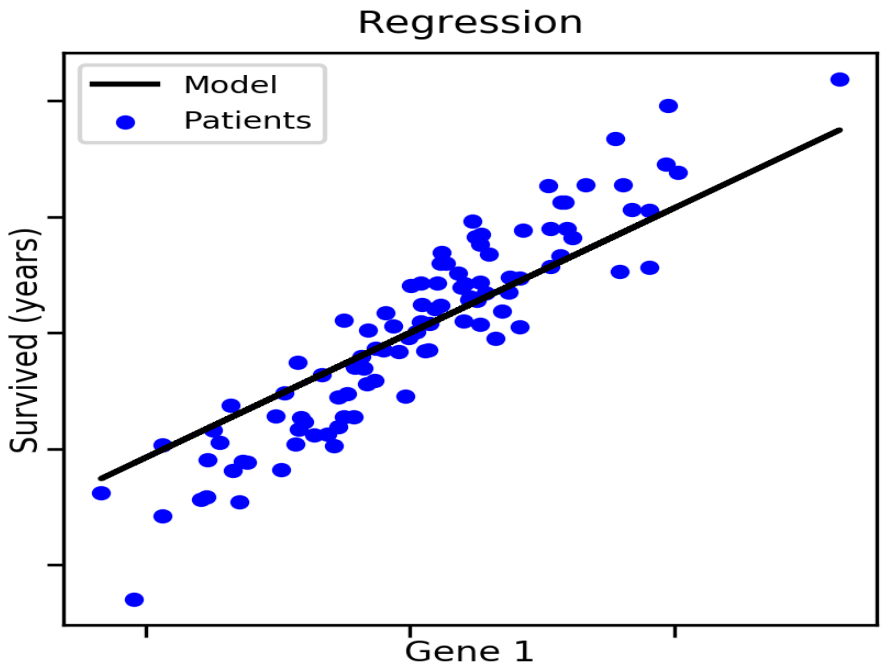
[sklearn docu]

Supervised learning

Swiss Institute of Bioinformatics

Fundamentally, classification is about predicting a label and regression is about predicting a quantity.

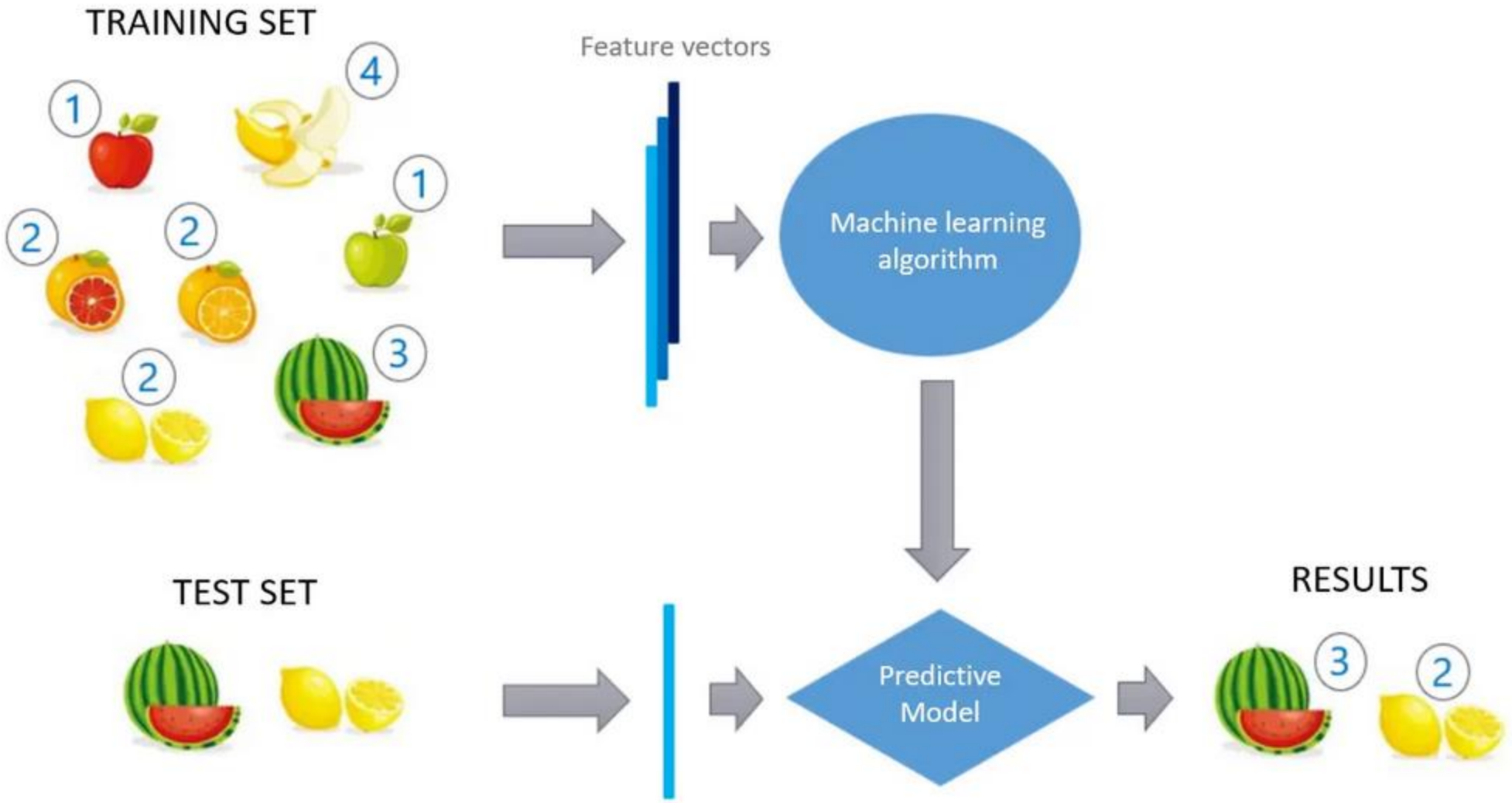




Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).

Supervised learning

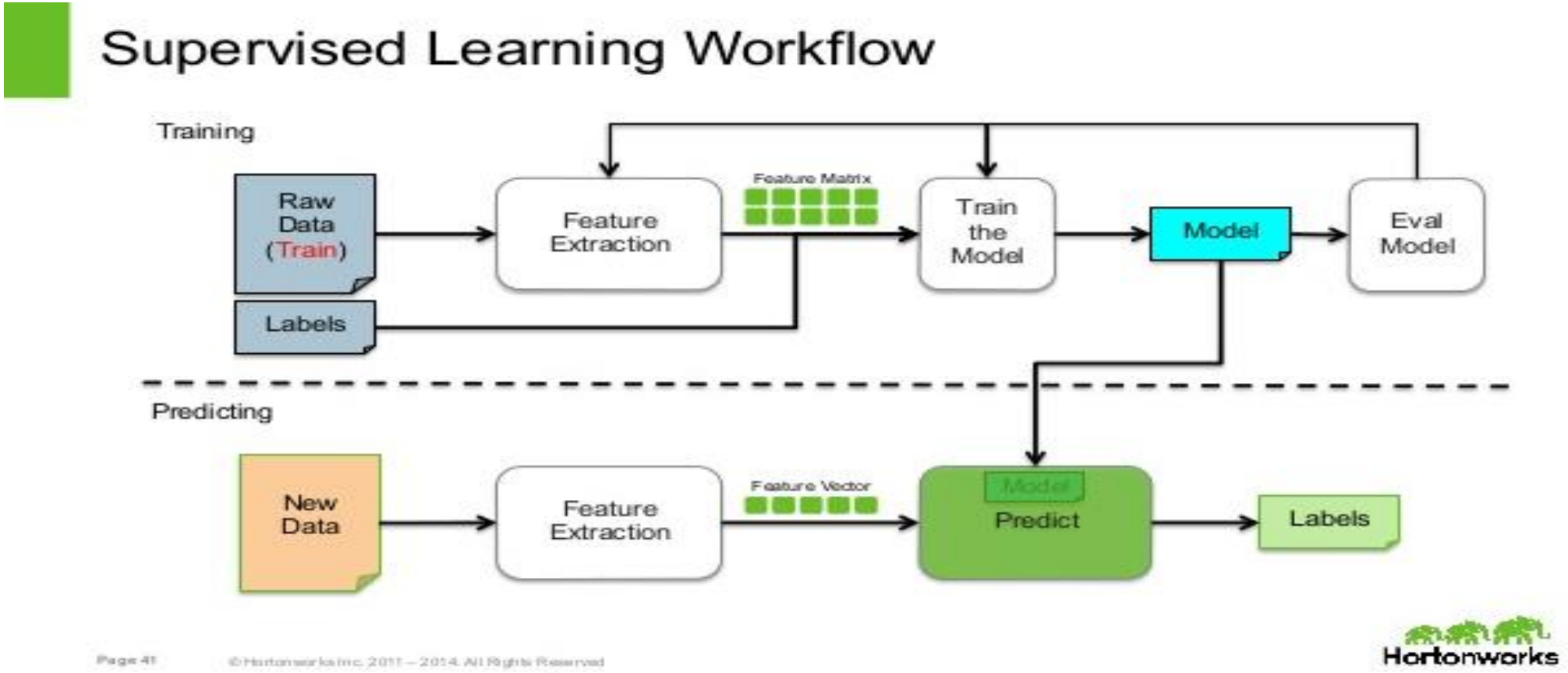


Swiss Institute of Bioinformatics

Isabelle Dupanloup, BCF

Supervised learning workflow

Swiss Institute of Bioinformatics



Supervised learning datasets

Swiss Institute of Bioinformatics

- **Training dataset**

The subset of the dataset provided to the algorithm for learning is called the training set.

- **Validation dataset**

A set of examples used to tune the parameters of a classifier/regressor.

Hybrid dataset: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing

- **Test dataset**

A set of examples used only to assess the performance of a fully-specified classifier/regressor

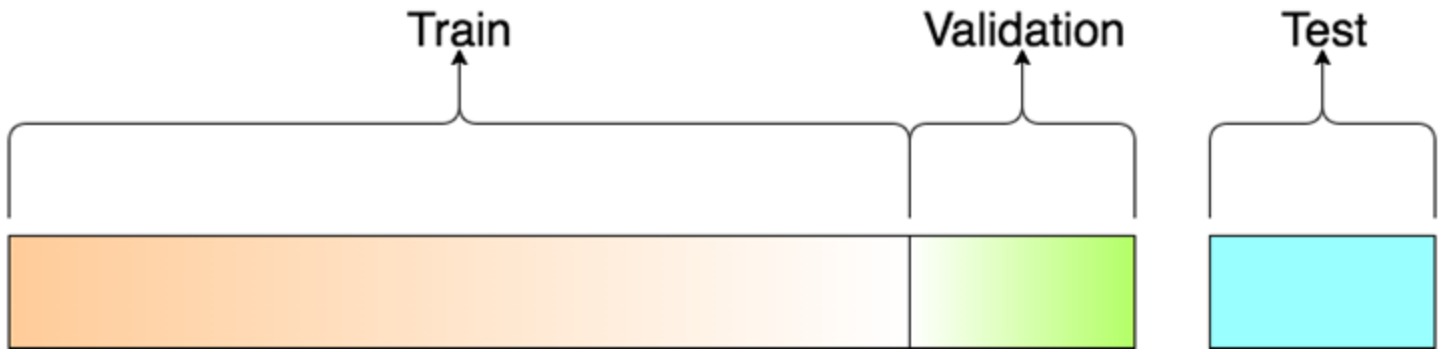
Training, Validation and Test Sets

Swiss Institute of Bioinformatics

- Given a dataset, we can always find a classifier that perfectly predicts the outcome (for example the 1-NN classifier) for this dataset.
- However, the purpose of training a classifier is to give us accurate predictions on a new and independent test dataset. The performance on the training data may be too optimistic and cannot be reproduced on the test data. We call this "*overfitting the training data*".
- When choosing a classifier, we will always select the one that performs best on the test or validation data.

Supervised learning datasets

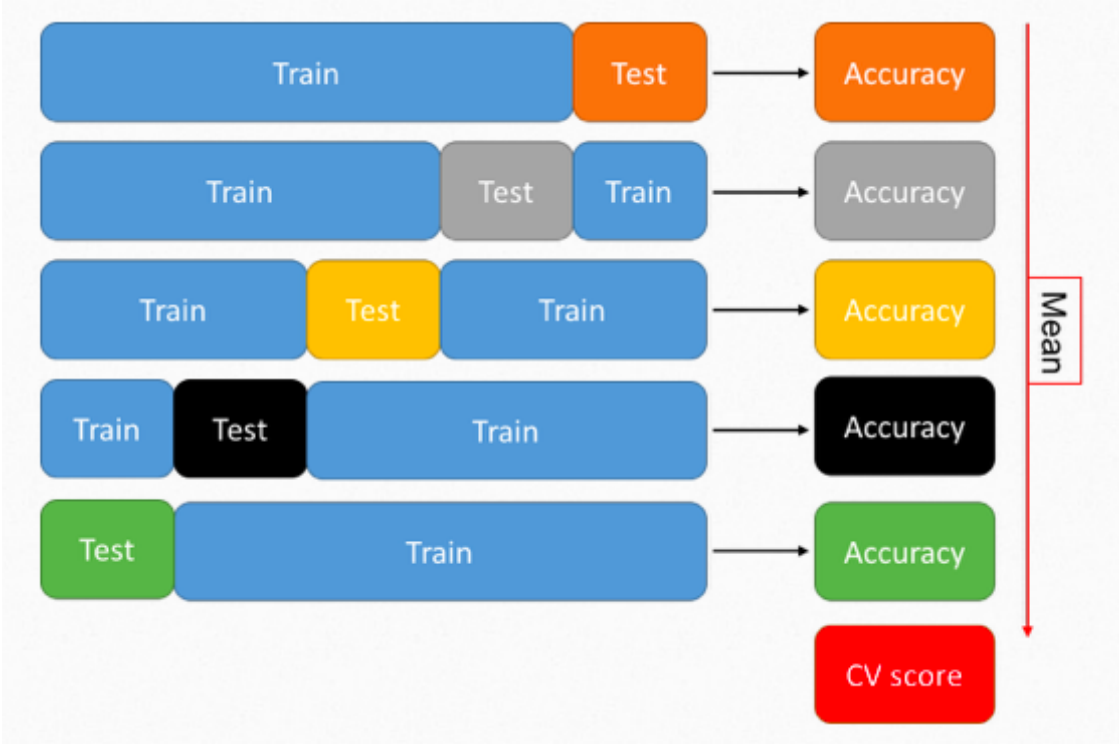
- **Train-test-validation split ratio**



Supervised learning datasets

Swiss Institute of Bioinformatics

- **K-fold cross validation**



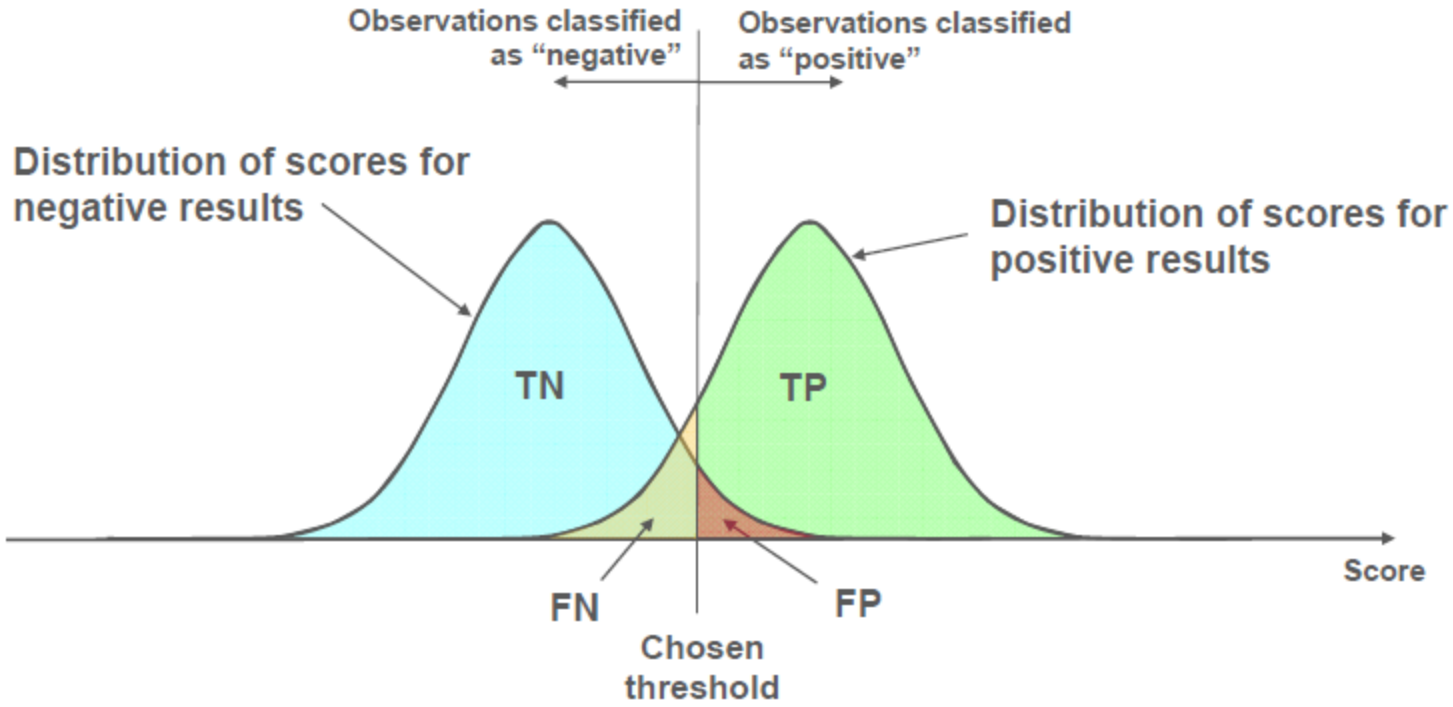
Assessing performance

Swiss Institute of Bioinformatics

- **TP (True Positives)**: number of positive predictions which are correct
- **FP (False Positives)**: number of positive predictions which are incorrect
- **TN (True Negatives)**: number of negative predictions which are correct
- **FN (False Negatives)**: number of negative predictions which are incorrect

confusion matrix

LABEL / PREDICTED	P	N
P	TP	FN
N	FP	TN



Isabelle Dupanloup, BCF

Assessing performance

Swiss Institute of Bioinformatics

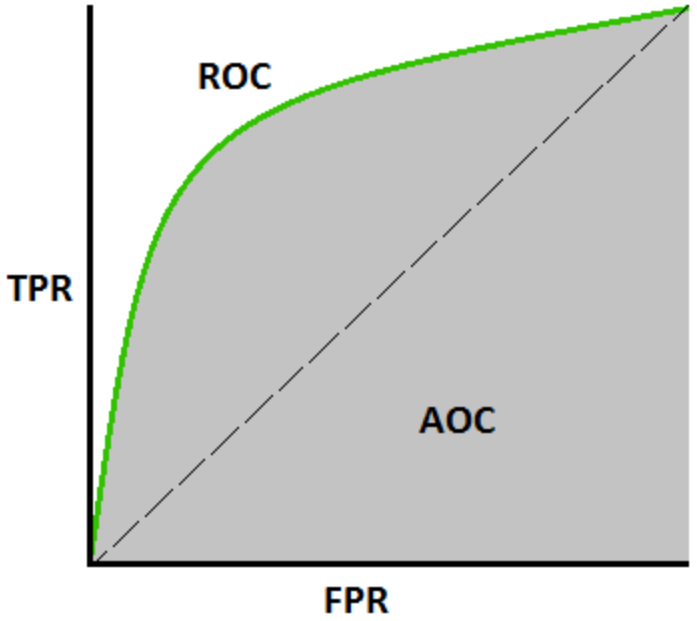
- **Precision**: fraction of the positive predictions that are actually positive: $\frac{TP}{TP + FP}$ ⟸ Total of positive predictions
- **Specificity**: fraction of the negative predictions that are correctly identified: $\frac{TN}{TN + FP}$ ⟸ Total of negative labels
- **Recall / TPR**: fraction of the positive predictions that are correctly identified: $\frac{TP}{TP + FN}$ ⟸ Total of positive labels
  Sensitivity
- **FPR**: 1 – Precision = $\frac{FP}{TN + FP}$ ⟸ Total of negative labels
- **F-measure**: harmonic mean of precision and recall: $2 * \frac{precision * recall}{precision + recall}$
- **Accuracy**: fraction of the total predictions that are correctly identified: $\frac{TP + TN}{TN + TN + FP + FN}$

Assessing performance: AUC - ROC curve

Swiss Institute of Bioinformatics

- **ROC = Receiver Operating Characteristic**
- **AUC = Area Under the Curve**

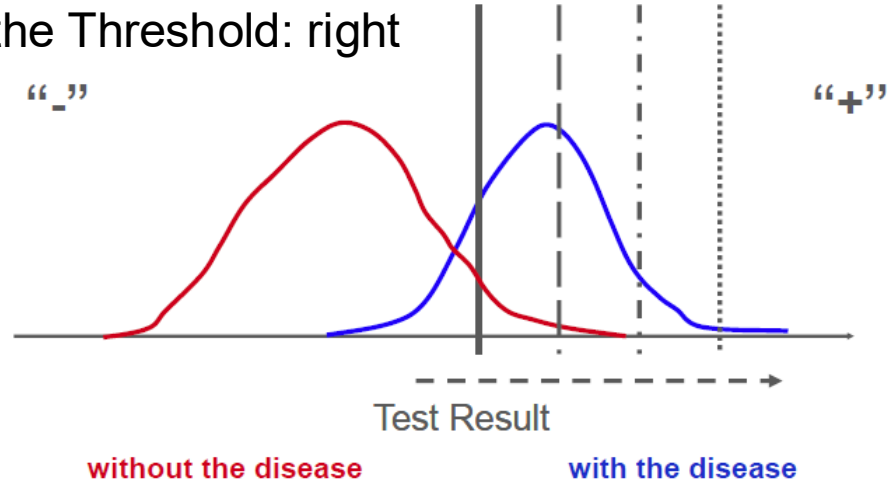
- AUC - ROC curve is a performance measurement for the classification problems at various threshold settings
- ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.



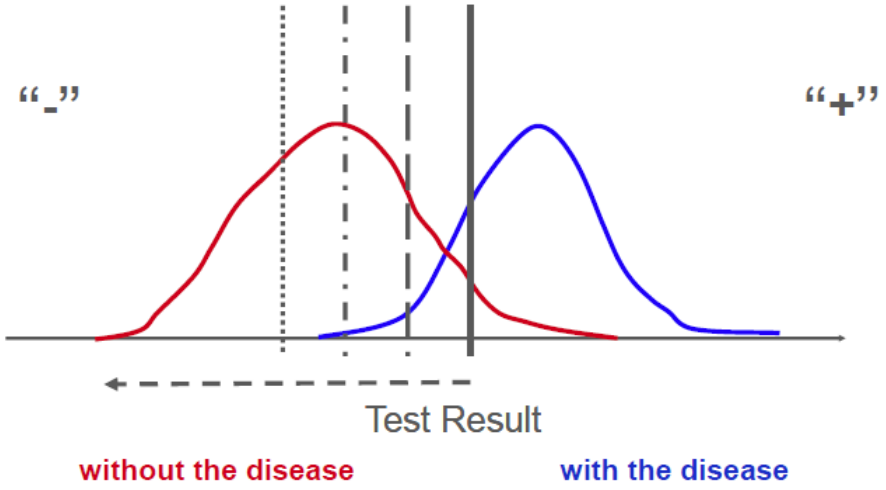
Assessing performance: AUC - ROC curve

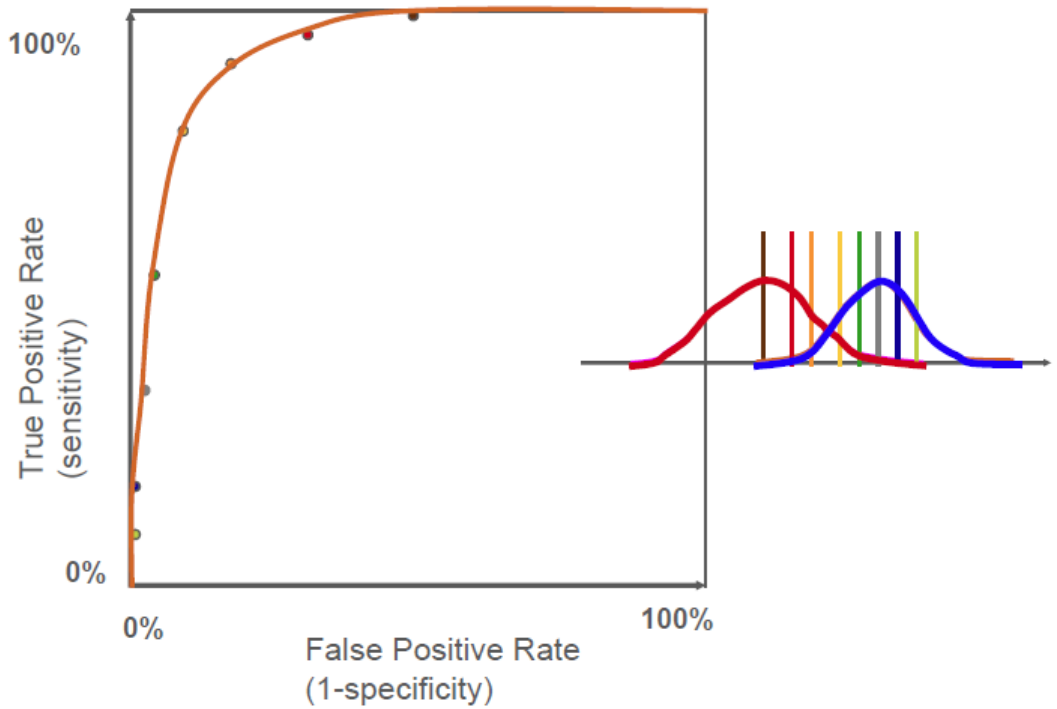
Swiss Institute of Bioinformatics

Moving the Threshold: right



Moving the Threshold: left

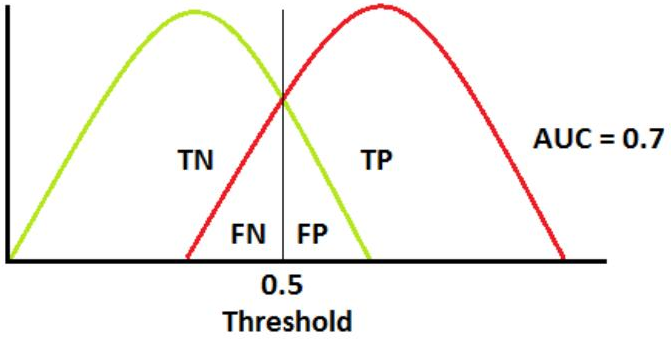


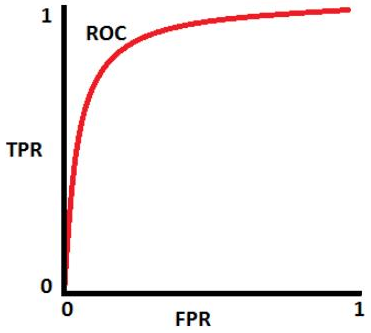


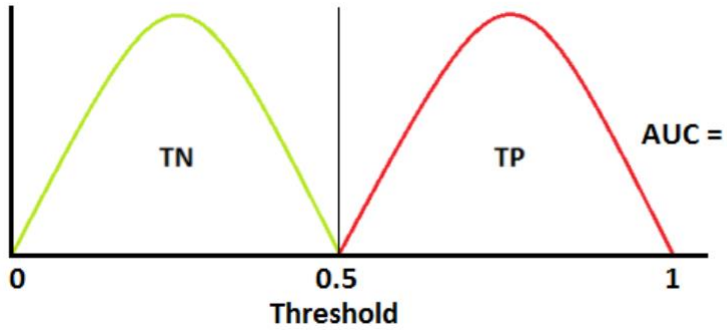
Isabelle Dupanloup, BCF

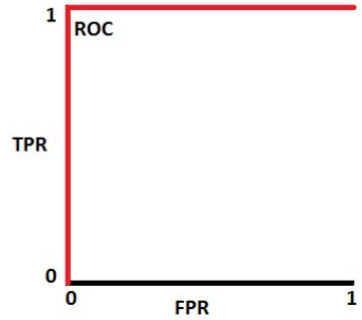
Assessing performance: AUC - ROC curve

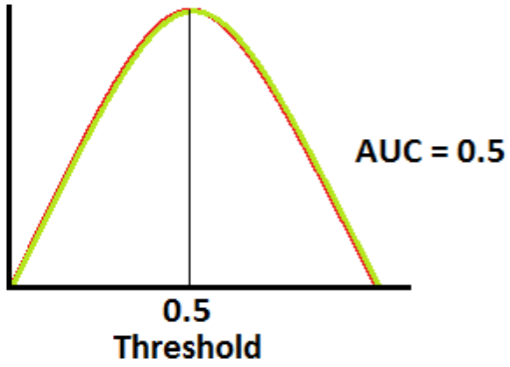
Swiss Institute of Bioinformatics

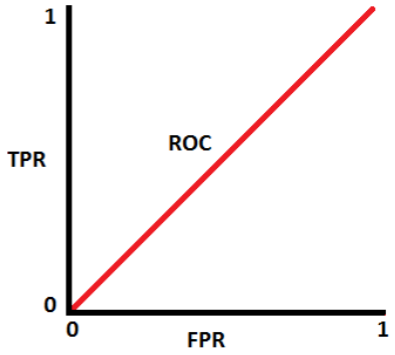


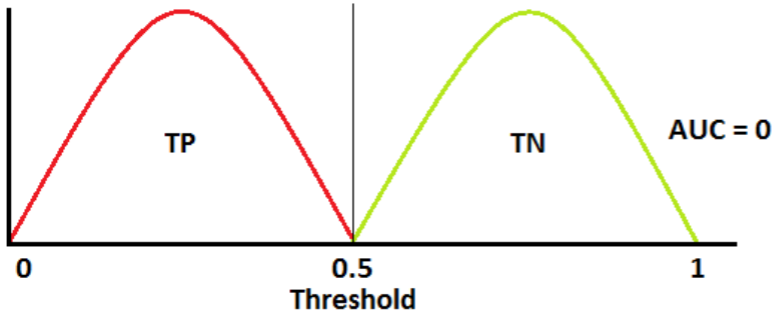


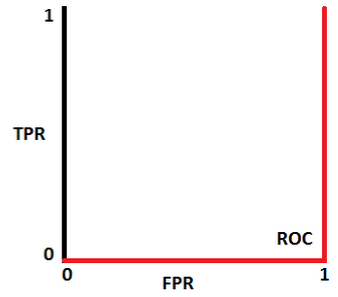












Isabelle Dupanloup, BCF

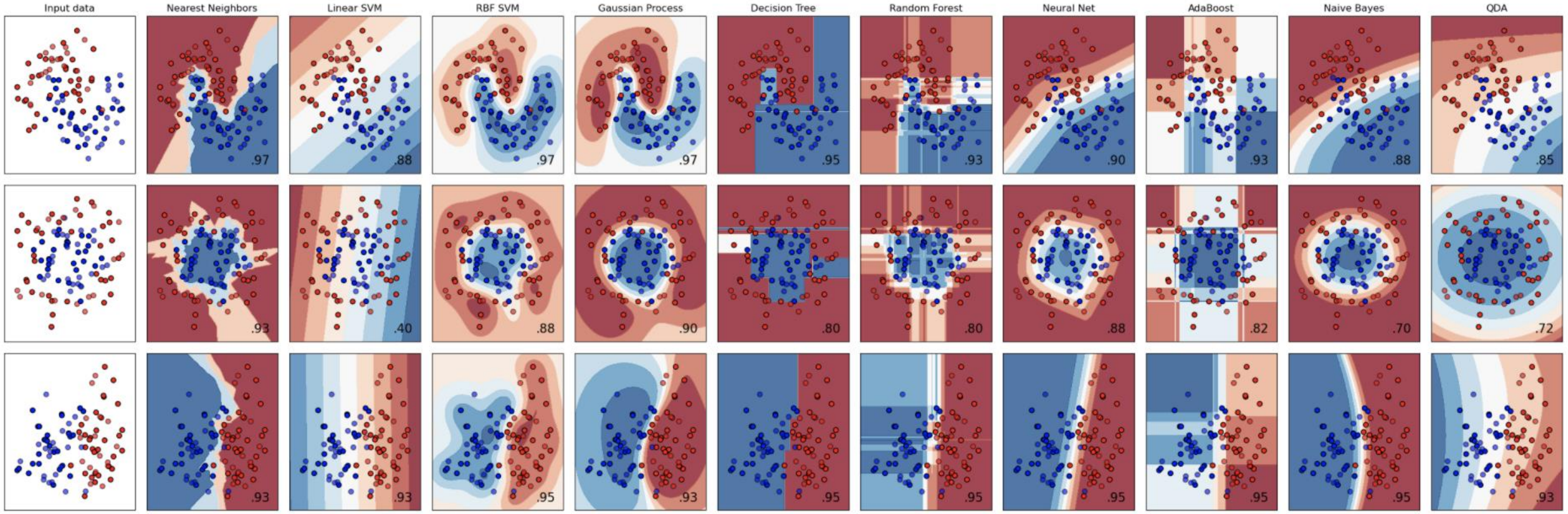
Which Classifier Should We Choose?

Swiss Institute of Bioinformatics

- There is a mathematical theorem that says that there is no universally best classifier. It really depends on the data!
- One needs to test several classifiers for each dataset and compare the results.
- Best is to have a few favorite classifiers (e.g. kNN, LR, SVM), which you know well, and which are somehow complementary (local density based, linear, or inbetween)

Which Classifier Should We Choose?

Swiss Institute of Bioinformatics



[sklearn classfier comparison]

Some More Notations:

Swiss Institute of Bioinformatics

- In the classification task, we would like to predict the outcome $\boldsymbol{y}$ from the features $\boldsymbol{X}$.
- We choose a classifier $\mathcal{C}_{\boldsymbol{\phi}}$ with *hyper parameters* $\boldsymbol{\phi}$ and *internal parameters* $\boldsymbol{\theta}$.
- Hyper parameters (e.g k in $k-NN$, kernel in SVM, learning rate in LR) have to be specified by the user or learned in CV loop outside the classifier.
- The internal parameters $\boldsymbol{\theta}$ are learned by the classifier (e.g. the coefficients in LR, or support vector coefficients in SVM).
- The classifier learns those *internal parameters* $\boldsymbol{\theta}$ that are able to predict $\boldsymbol{y}$ from the features $\boldsymbol{X}$ with the lowest error:

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}\left(\mathcal{C}_{\boldsymbol{\phi}}(\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{y}\right)$$

where the loss function $\mathcal{L}$ measures the error between predicted ($\mathcal{C}_{\boldsymbol{\phi}}(\boldsymbol{X}, \boldsymbol{\theta})$) and known ($\boldsymbol{y}$) outcomes.

Bias versus Variance Tradeoff

Swiss Institute of Bioinformatics

- All classifiers make errors and wrongly predict class labels for certain instances. The error can be split into two part: *bias* and *variance*.
- *Variance*: the classifier will depend on the subset of the data we choose for training. The more a classifier depends on the training data selection, the higher its variance.
- *Bias*: the variance usually decreases with the size of the training set. However, even for very large training sets, there will be an error left due to the bias of the classifier. For example, a linear classifier will never be perfectly able to separate classes with non-linear boundaries even if large datasets are available.
- Classifiers with low bias fit the training data well, but may have a large variance and low performance on the test set. Others have a larger bias, but are more stable and predictive for the test set.

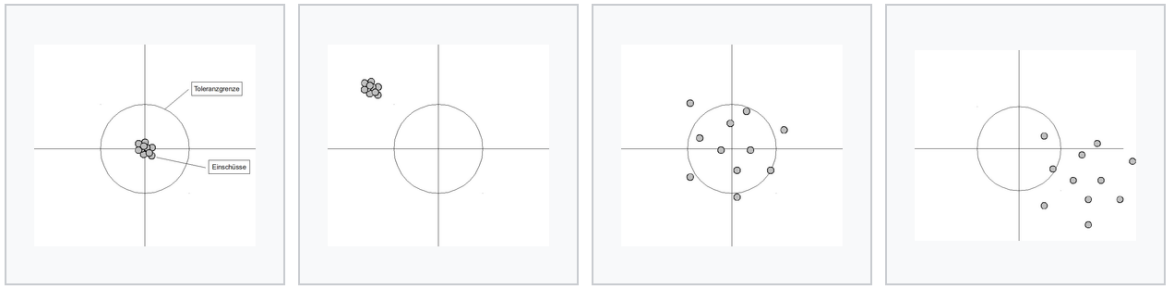
Bias versus Variance Tradeoff

Deep learning

Linear SVM
Regularized
Logistic Regr.

1-NN
RBF SVM

Linear
Perceptron



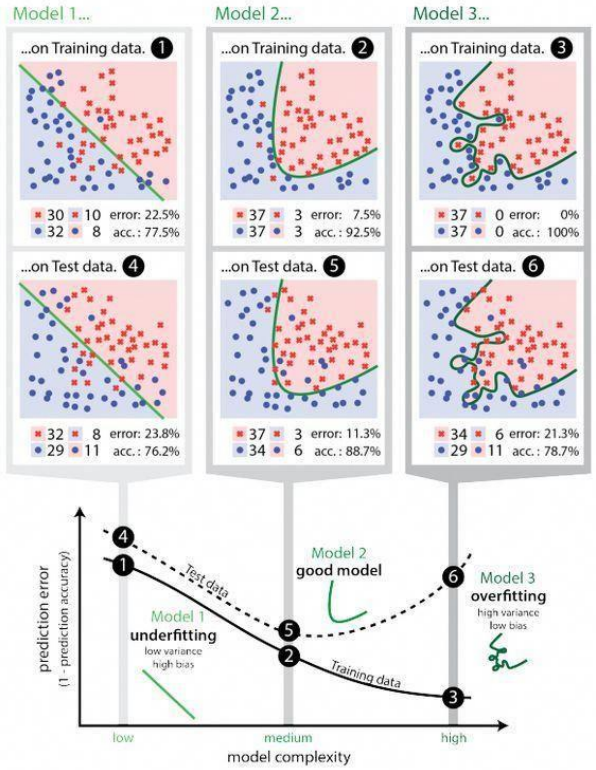
bias low,
variance low:
precision high

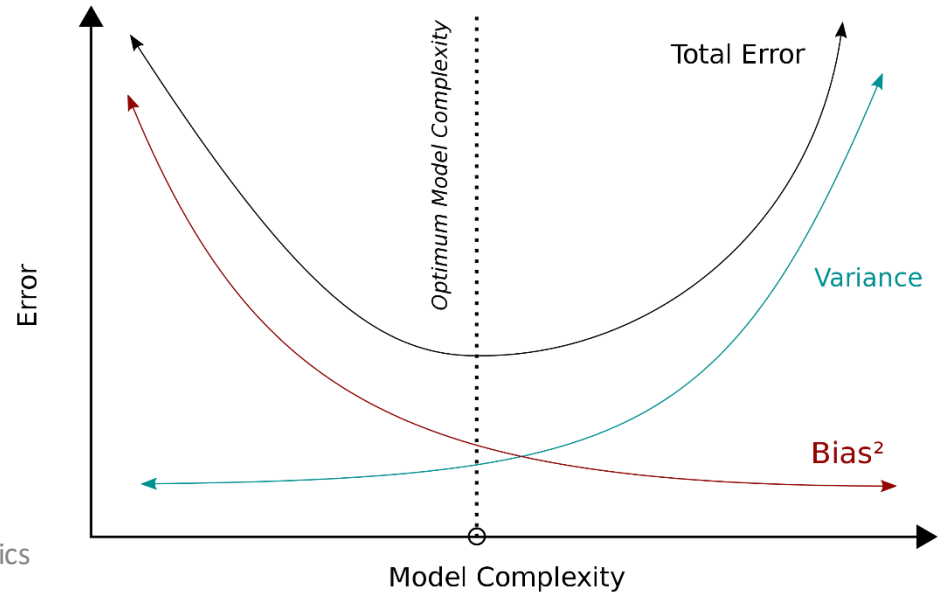
bias high,
variance low:
precision high

bias low,
variance high:
precision low

bias high,
variance high:
precision low

https://en.wikipedia.org/wiki/Bias-variance_tradeoff

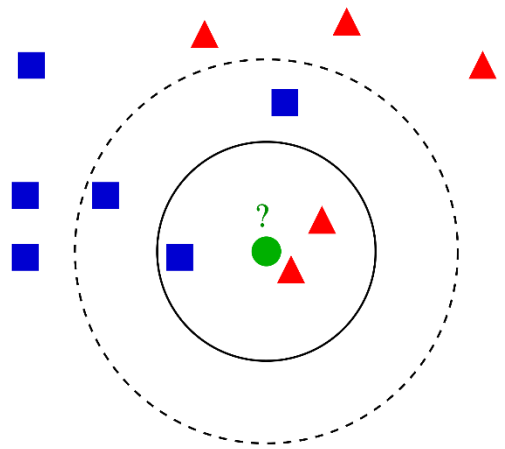




K-Nearest Neighbor Classifier

Swiss Institute of Bioinformatics

- The k-Nearest neighbor (k-NN) classifier is probably the simplest non-trivial classifier. Given a training dataset $\boldsymbol{X}^{train}$, $\boldsymbol{y}^{train}$ we predict the class of an instance $\boldsymbol{x}$ by taking the k nearest instances $\boldsymbol{x}_i^{train}, i \in kNN(\boldsymbol{x})$ according to some distance metric $d\left(\boldsymbol{x}, \boldsymbol{x}_i^{train}\right)$ and predict the majority class in $y_i^{train}, i \in kNN(\boldsymbol{x})$
- Instead of simply predicting the majority class we can also weight the classes by $^{1}/_{d\left(\boldsymbol{x},\boldsymbol{x}_i^{train}\right)}$.
- The number of neighbors k we will determine with cross validation.



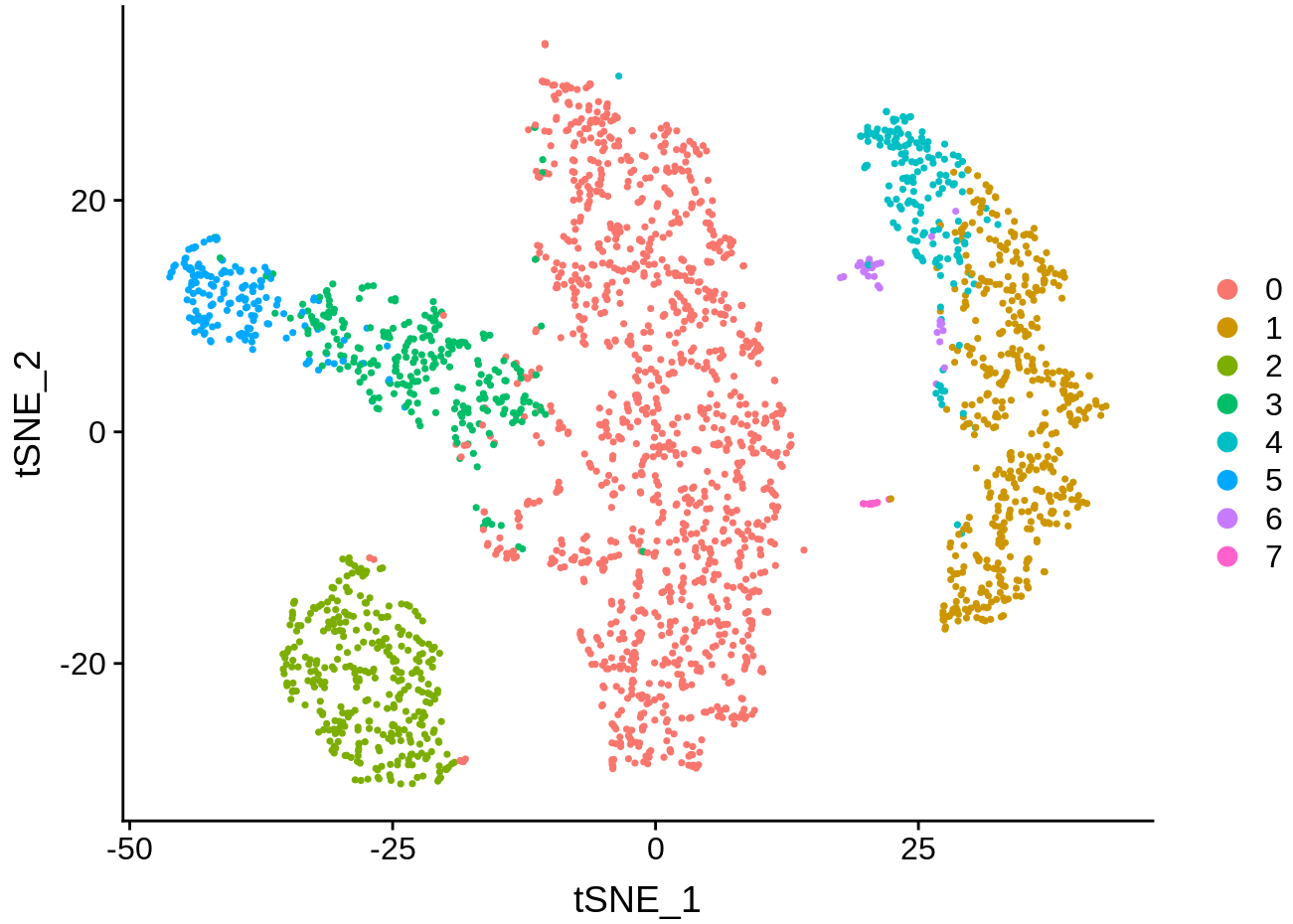
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

k-Nearest Neighbors (KNN)

Swiss Institute of Bioinformatics

- Pros
  - Training phase much faster compared to other classification algorithms
  - No need to train a model for generalization
  - Can be useful in case of nonlinear data
  - Can be used for regression: the output value for the object is computed by the average of k closest neighbors value.
- Cons
  - Testing phase is slower and costlier in terms of time and memory
  - It requires large memory for storing the entire training dataset for prediction
  - It requires scaling of data because KNN uses the Euclidean distance between two data points to find nearest neighbors. Euclidean distance is sensitive to magnitudes. The features with high magnitudes will weight more than features with low magnitudes
  - It is not suitable for large dimensional data.

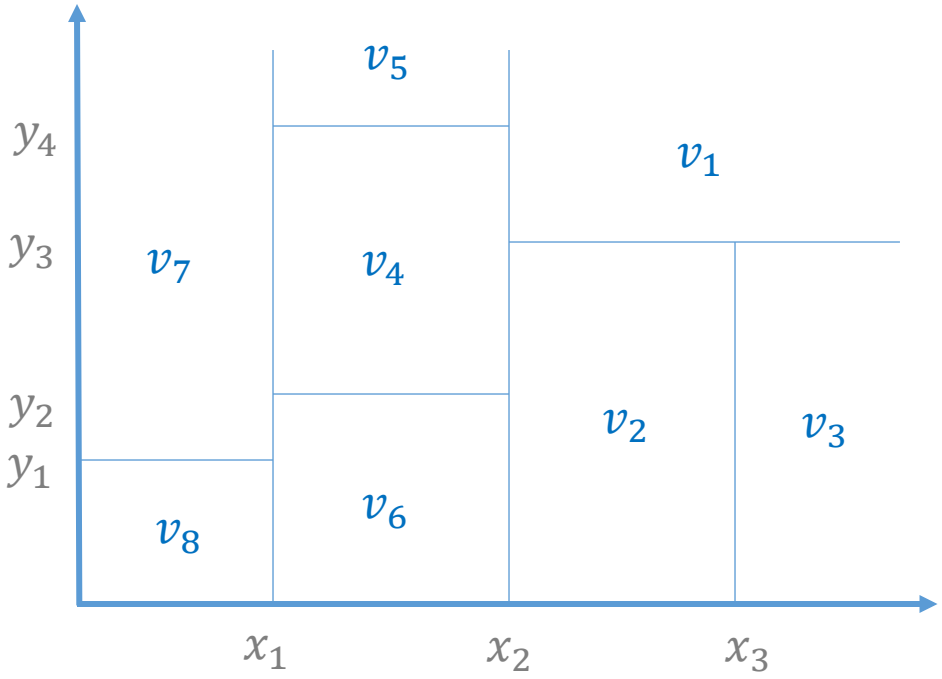
k-Nearest Neighbors (KNN)

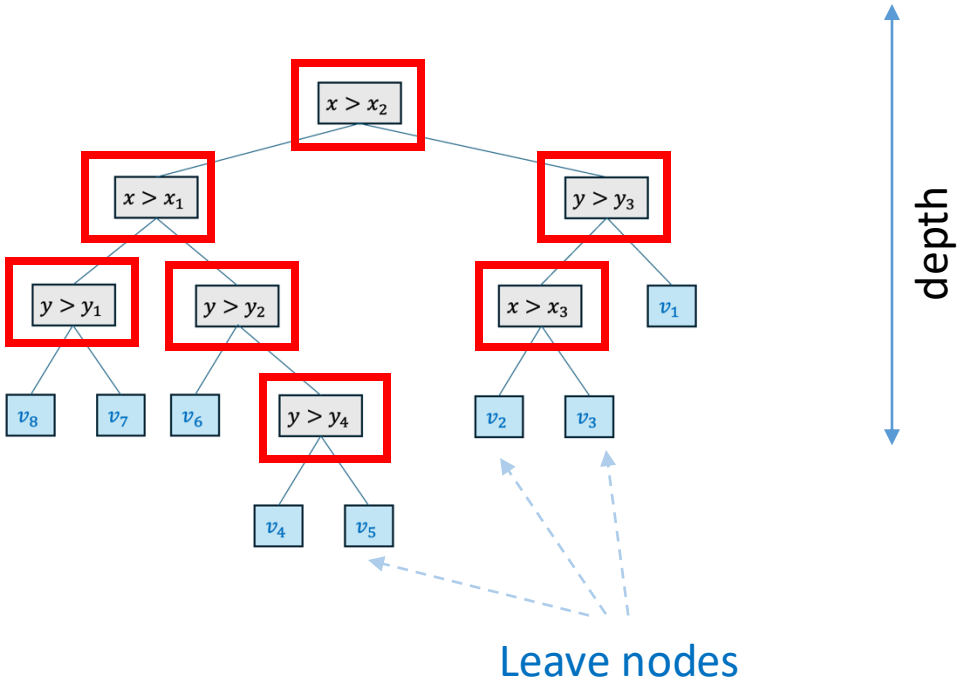


https://scrnaseq-course.cog.sanger.ac.uk/

Swiss Institute of Bioinformatics

Trees

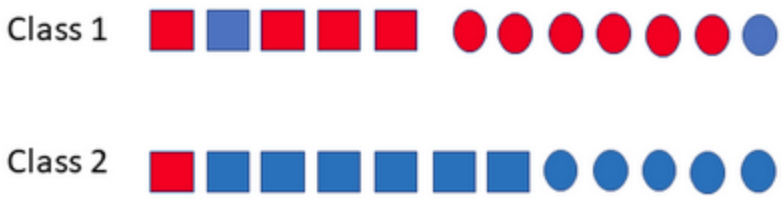




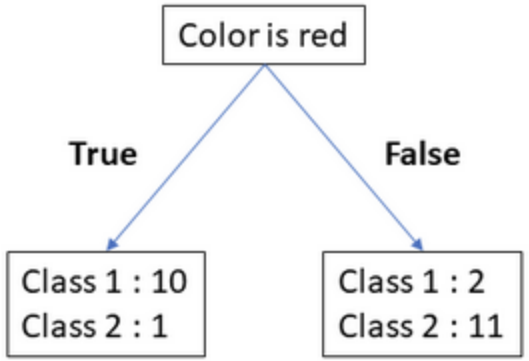
Decision trees

Swiss Institute of Bioinformatics

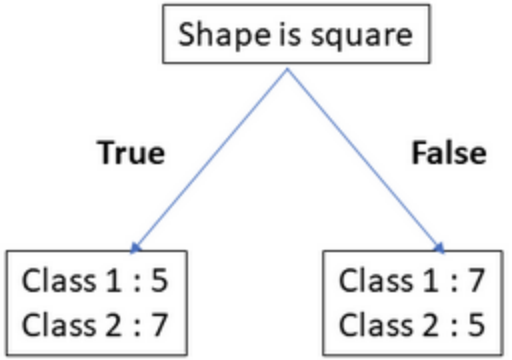
- Algorithm:
  - hierarchical sequence of questions on your features that can be answered by yes or no and that subdivides the data into subgroups on which a new question is asked, and so on and so on
  - tests all the features and chooses the most discriminative feature (in terms of the labels) to divide the data into subsets (impurity: Shannon entropy, Gini index)



Features:
Color: blue or red
Shape: square or circle

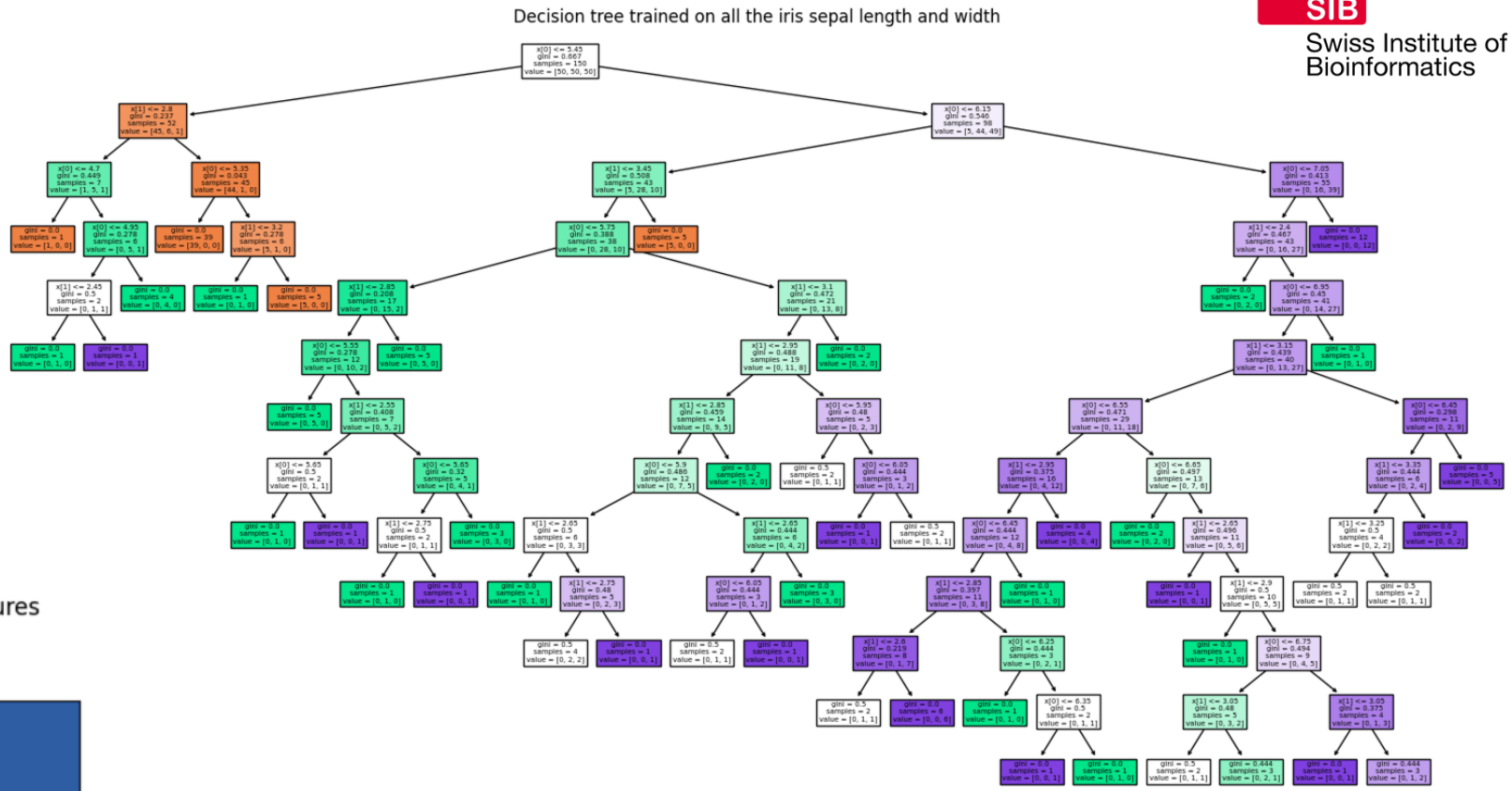


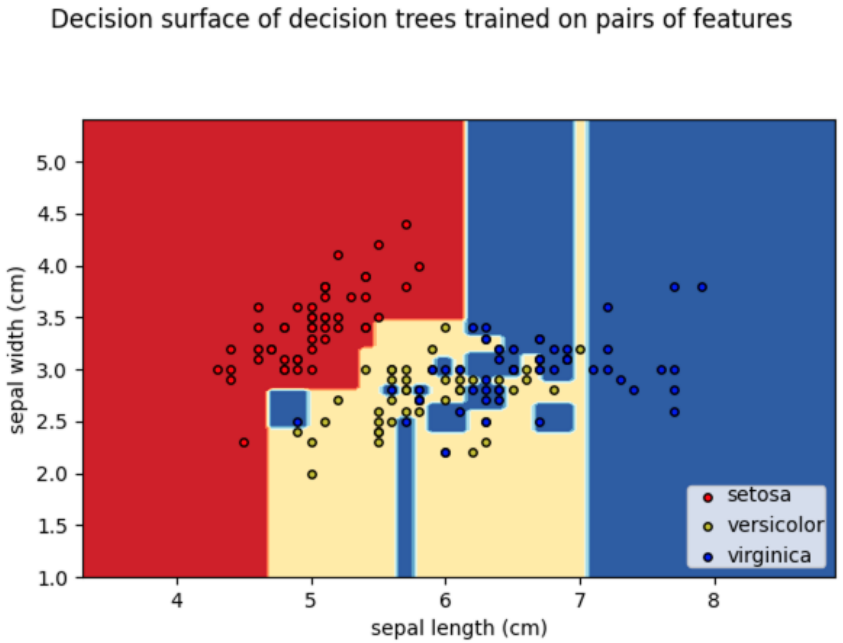
Those groups are highly skewed toward certain class



Those groups are well mixed

Trees





[Code on sklearn page](#) !

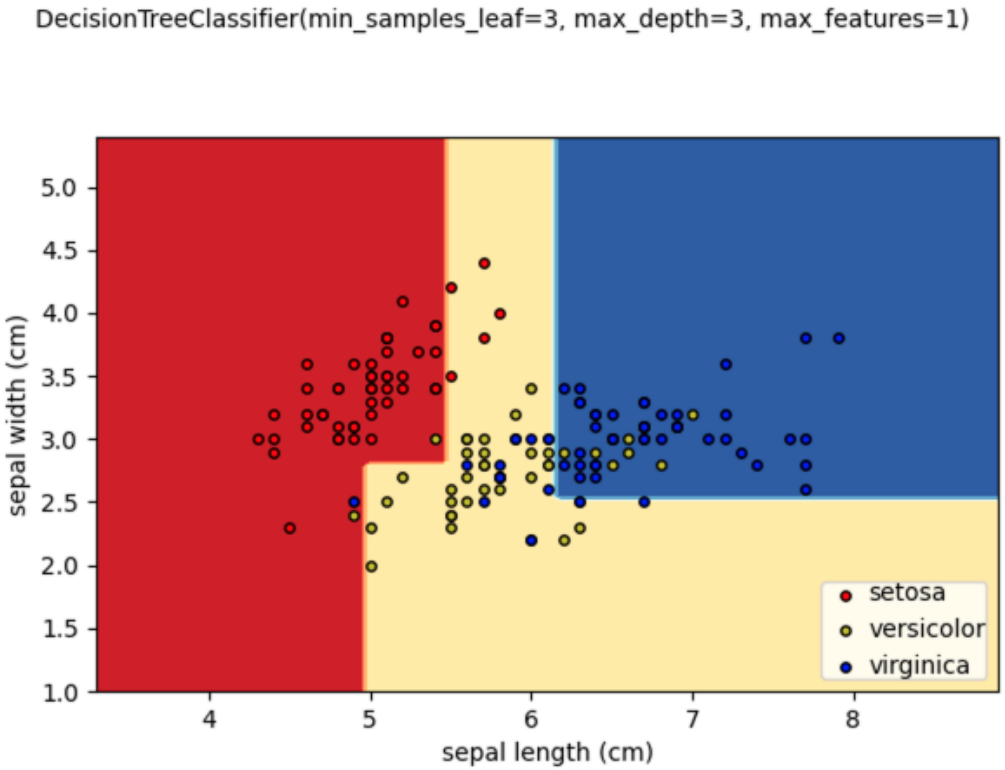
Markus Müller
SIB Swiss Institute of Bioinformatics

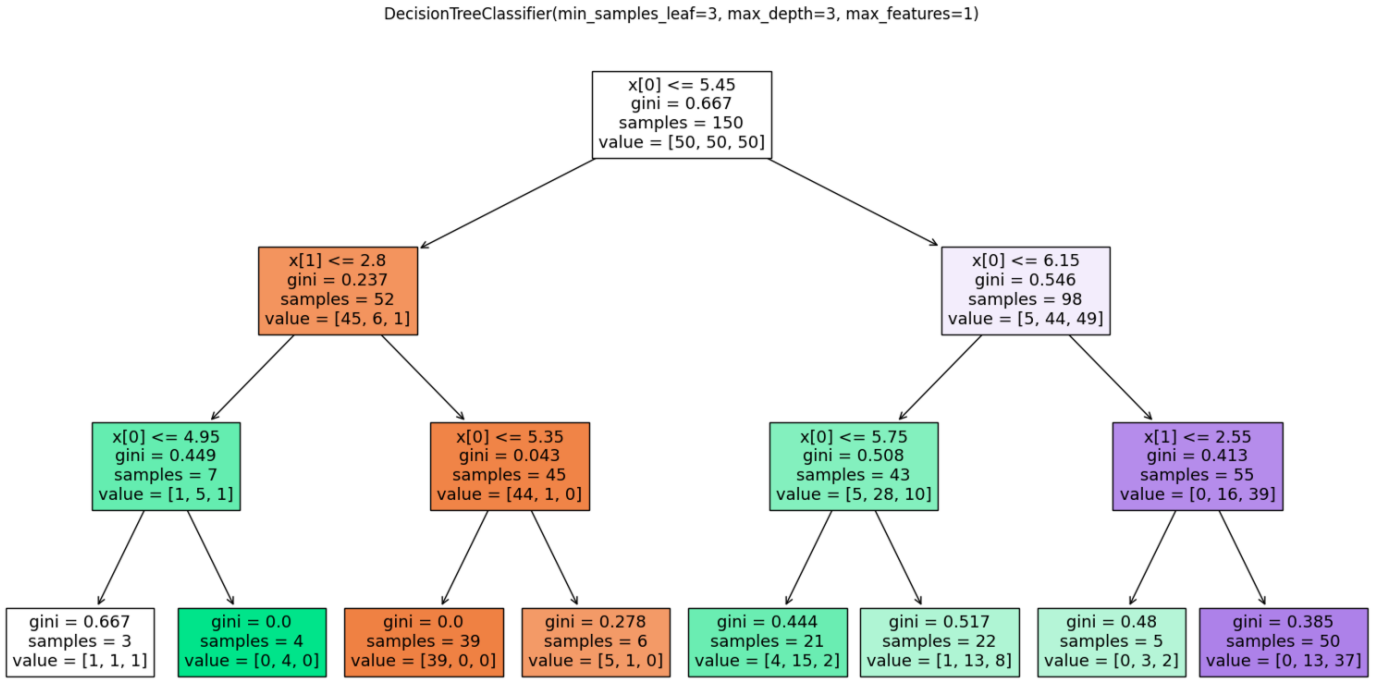
Trees

- Regression and classification trees
- Trees are universal approximators (no bias) if the size is not restricted
- Tree's are naturally able to deal with categorical features, and are not sensitive to the distribution of features values
- Parameters:
  - For each node k we have a feature F_k and threshold θ_k
  - For each leaf node l we have a weight ν_l for regression, and a class label c_l for classification
- Tree hyperparameters add constraints to its growth and weights:
  - max_depth: maximal number of internal nodes before leaf
  - max_leaf_nodes: maximal number of leaf nodes in the tree
  - min_samples_leaf: minimal number of data points in the training set within leaf
  - L1 or L2 regularization of weights

Markus Müller
Swiss Institute of Bioinformatics

Tree hyperparameters





Some Notations:

Swiss Institute of Bioinformatics

D features

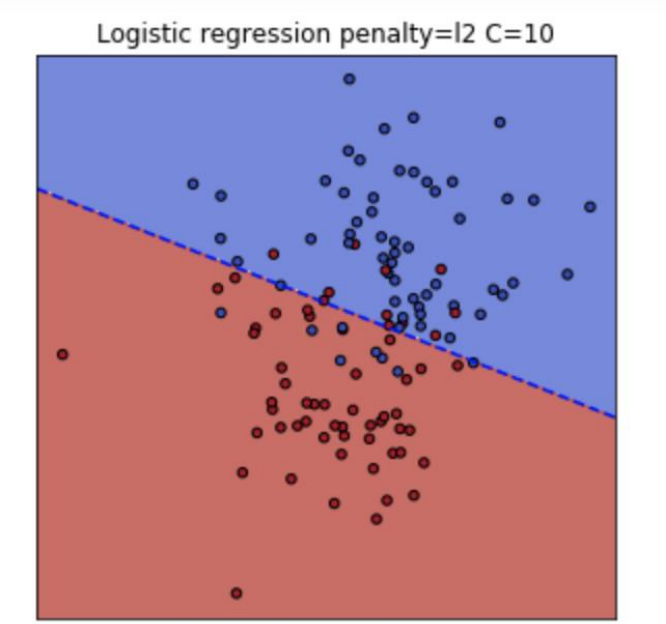
Data matrix: $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_N)^{\boldsymbol{T}} = \begin{pmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{pmatrix}$ *N items*

Outcome: $\boldsymbol{y} = (y_1, \ldots, y_N)^T$

Outcome is a known attribute of the cases (rows). It can be a binary factor (e.g. healthy/diseased), a multiclass label (e.g. protein function) or a continuous variable (e.g. survival time).

Logistic Regression

- In logistic regression we assume that our data is linearly separable by a hyperplane $h : w_0 + \sum_{i=1}^{D} w_i x_i = w_0 + \boldsymbol{w} \cdot \boldsymbol{x} = 0$.

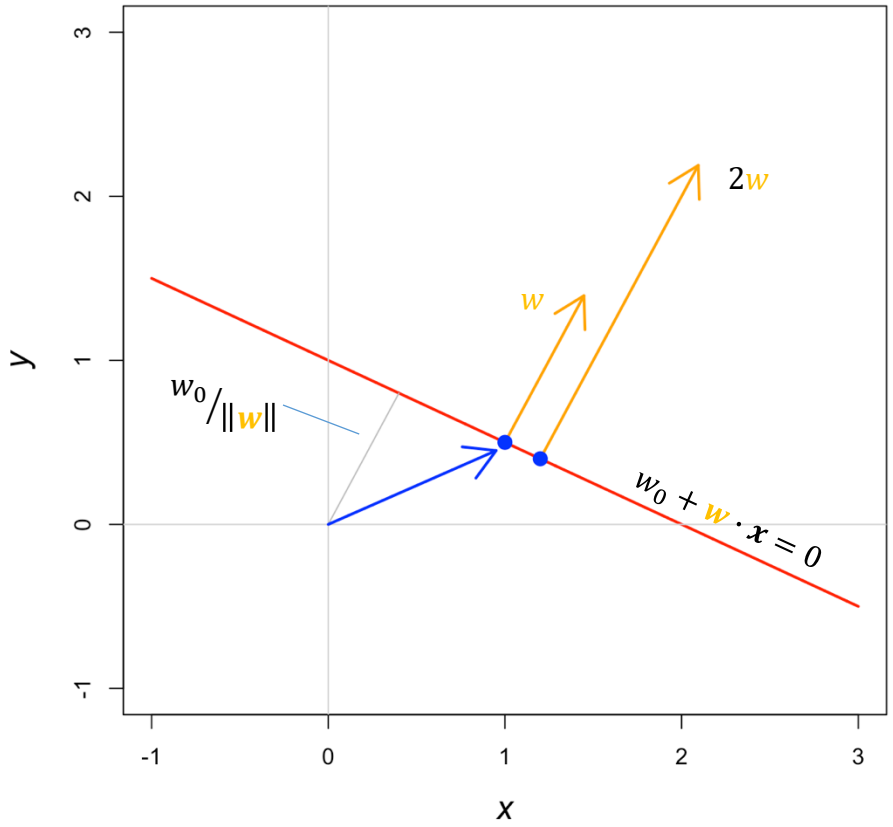


Logistic Regression

- Hyperplane $h: w_0 + \sum_{i=1}^{D} w_i x_i = w_0 + \boldsymbol{w} \cdot \boldsymbol{x} = 0$
- We can choose any $\lambda \boldsymbol{w}$ and λw_0 without changing h
- Distance d of a point $\boldsymbol{x}$ to h :

$$d \cdot \|\boldsymbol{w}\| = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

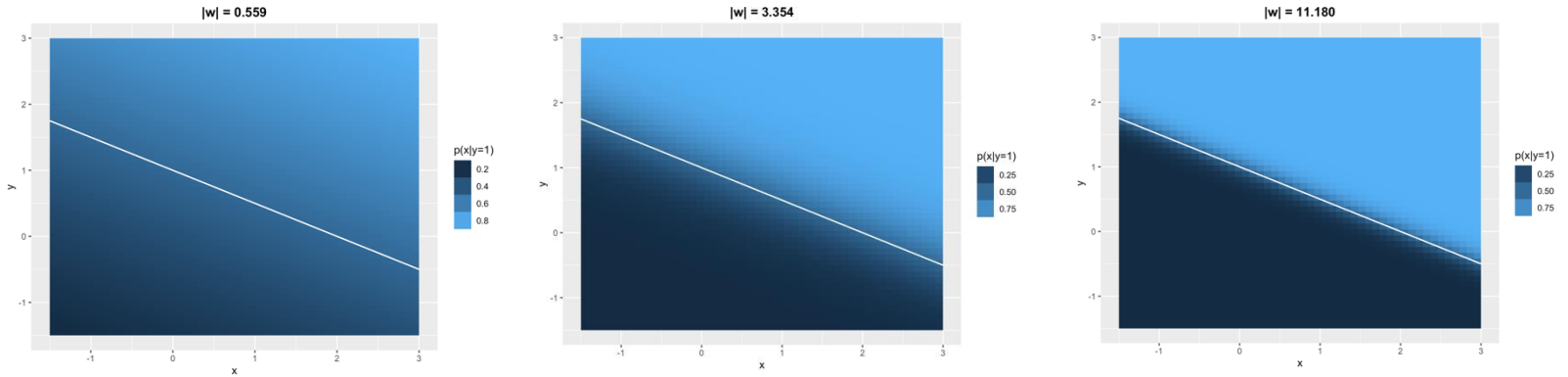
- If $\|\boldsymbol{w}\|$ is large then $h(\boldsymbol{x}) = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$ changes rapidly around h.

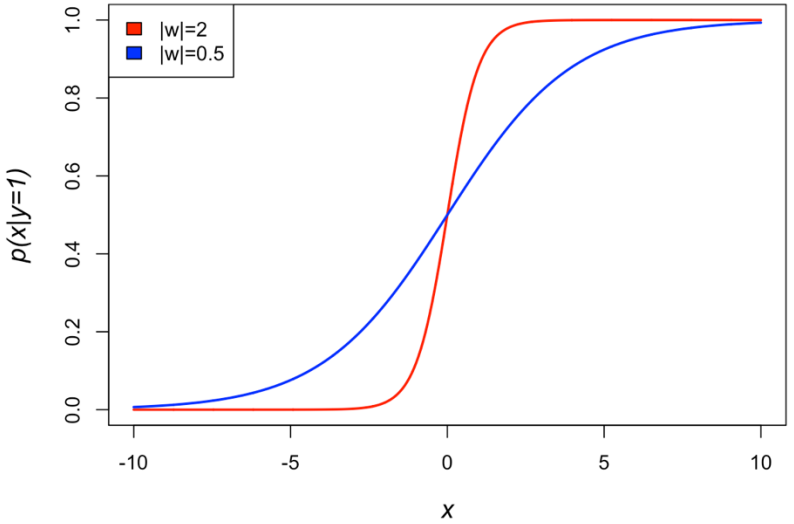


Logistic Regression

- We assume we have two classes labelled 0 and 1.
- We assume a sigmoid shaped class probability distribution p, which only depends on the distance d to h:
- $p(\boldsymbol{x}|y=1) = \frac{1}{1+e^{-(w_0+\boldsymbol{w}\cdot\ \boldsymbol{x})}}$

$$p(\boldsymbol{x}|y=0) = 1 - p(\boldsymbol{x}|y=1)$$





Logistic Regression

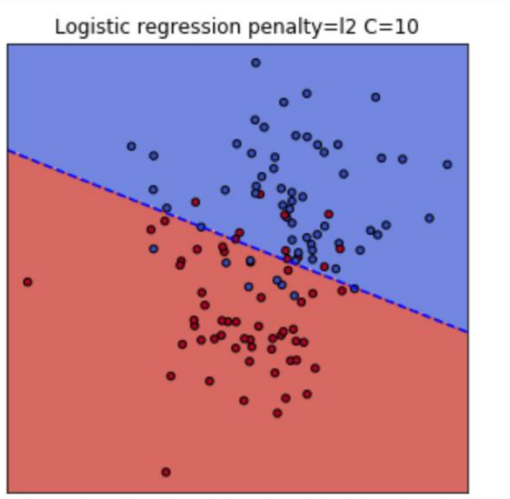
Swiss Institute of Bioinformatics

- To train the LR model, we have to find $w_0, \boldsymbol{w}$ that fit best to the training data. To evaluate the quality of a fit, we need a loss function $\mathcal{L}$.
- Loss function:

Class 1 Class 0

$$\mathcal{L}(w_0, \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) = -\frac{1}{N}\left\{\sum_{i,y_i=1} \log p(\boldsymbol{x}_i|y=1) + \sum_{i,y_i=0} \log p(\boldsymbol{x}_i|y=0)\right\} \geq 0$$

$p(\boldsymbol{x}_i|y=1) = 1 : log\{p(\boldsymbol{x}_i|y=1)\} = 0$
$p(\boldsymbol{x}_i|y=1) < 1 : log\{p(\boldsymbol{x}_i|y=1)\} < 0$



Logistic Regression

Swiss Institute of Bioinformatics

- Loss function:

$$\mathcal{L}(w_0, \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) = -\frac{1}{\sum_{i=1}^{N} s_i}\left\{ c_1 \sum_{i,y_i=1} s_i \cdot \log p(\boldsymbol{x}_i | y=1) + c_0 \sum_{i,y_i=0} s_i \cdot \log p(\boldsymbol{x}_i | y=0) \right\} \geq 0$$

sample_weight *sample_weight* *class_weight* *class_weight*

- LR finds the optimal $w_0, \boldsymbol{w}$ by minimizing $\mathcal{L}$. LR in sklearn offers several minimization algorithms.

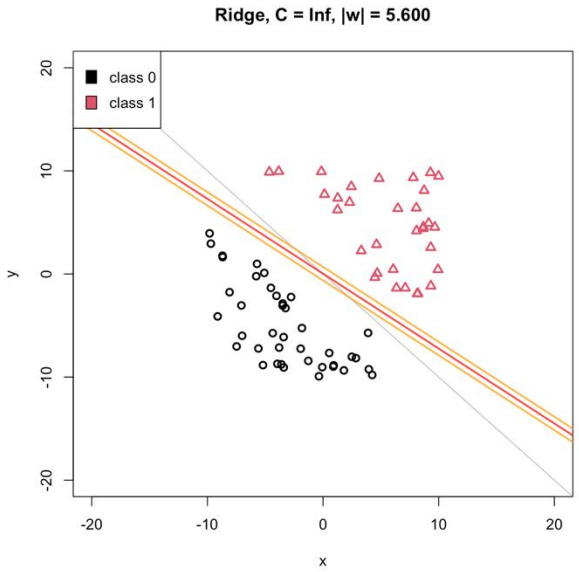
Logistic Regression

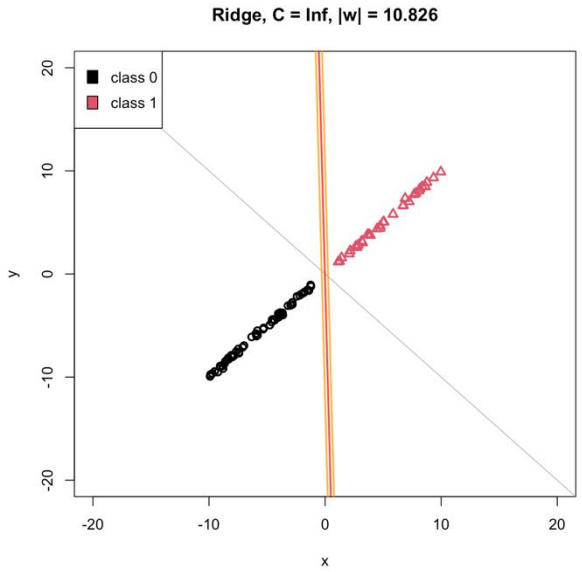
Swiss Institute of Bioinformatics

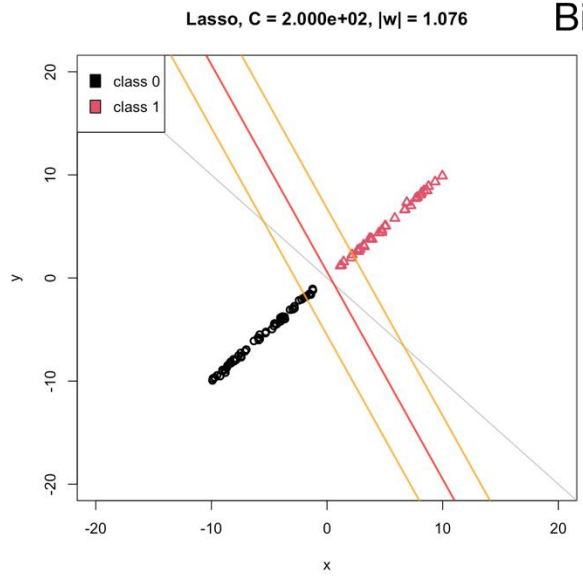
- Regularization: keep $\boldsymbol{w}$ small and boundary smooth:
  - $\mathcal{L}(w_0, \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) + \frac{1}{C}\sum_{i=1}^{D}|w_i|$: lasso or 'l1'
  - $\mathcal{L}(w_0, \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) + \frac{1}{2C}\sum_{i=1}^{D} w_i^2$: ridge or 'l2'
  - $\mathcal{L}(w_0, \boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) + \frac{1}{C}\left\{\sum_{i=1}^{D} \alpha|w_i| + \frac{(1-\alpha)}{2} w_i^2\right\}$
    - α : l1_ratio : elastic net

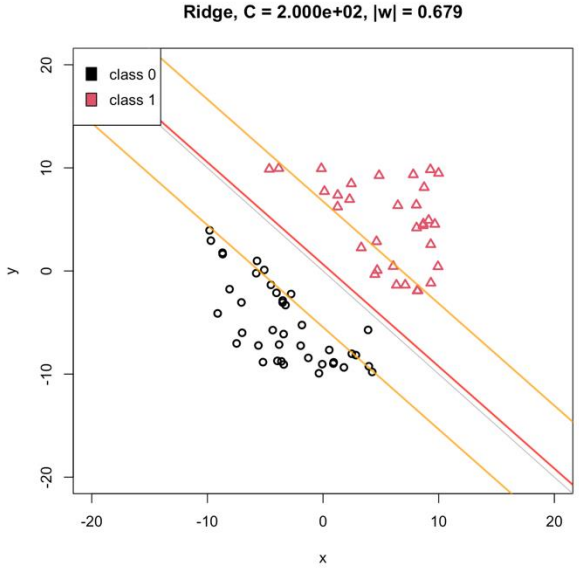
Logistic Regression

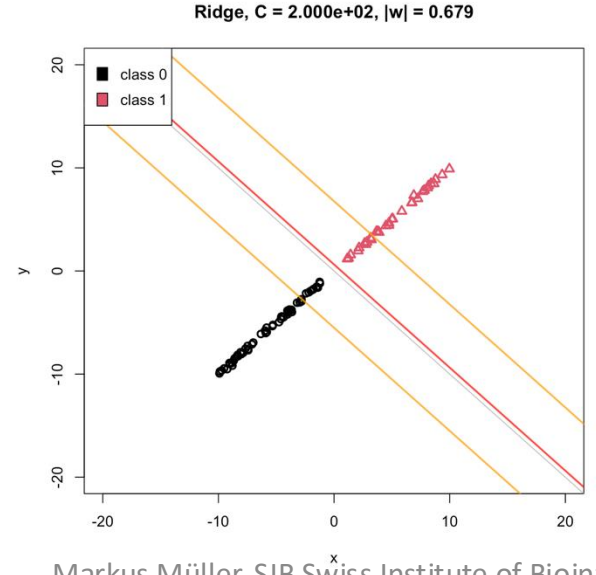
Swiss Institute of Bioinformatics

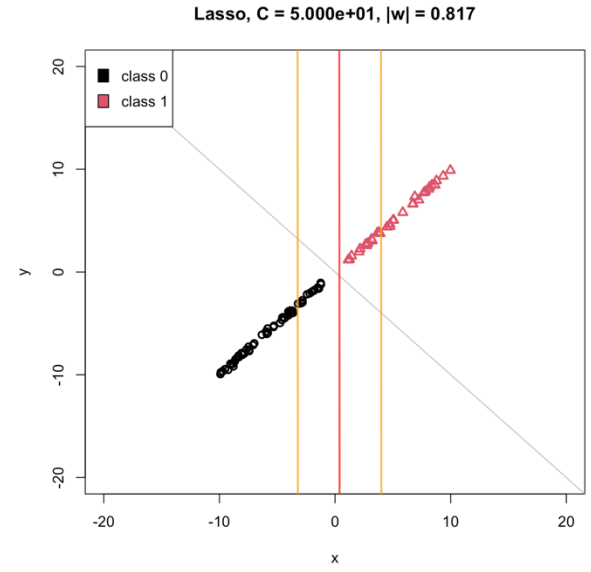












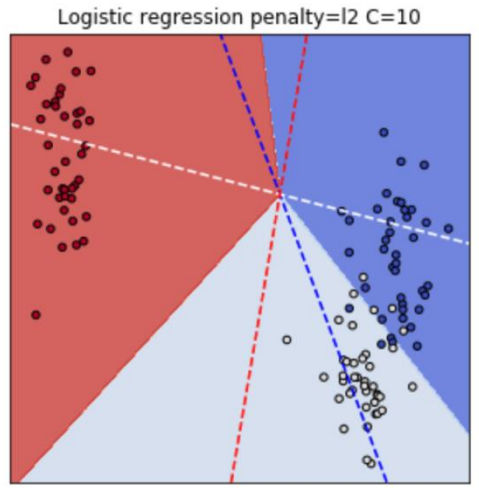
Multinomial Logistic Regression (K classes)

Swiss Institute of Bioinformatics

- $p(\boldsymbol{x}|y=k) = \frac{e^{w_{0k}+\boldsymbol{w}_k\cdot\boldsymbol{x}}}{\sum_{i=1}^{K} e^{w_{0i}+\boldsymbol{w}_i\cdot\boldsymbol{x}}}$
- Class $y(\boldsymbol{x}) = \underset{k}{\operatorname{argmax}}\, p(\boldsymbol{x}|y=k) = \underset{k}{\operatorname{argmax}}(w_{0k} + \boldsymbol{w}_k \cdot \boldsymbol{x})$
- Loss function:

$$\mathcal{L}(w_{0k}, \boldsymbol{w_k}, \boldsymbol{X}, \boldsymbol{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_i^k \log p(\boldsymbol{x}_i|y=k)\,;$$

$$y_i^k = \begin{cases} 1 : y(\boldsymbol{x}_i) = k \\ 0 : y(\boldsymbol{x}_i) \neq k \end{cases}$$



Logistic Regression

- Can be implemented very efficiently
- Scales to large data
- Good generalization thanks to regularization
- Few hyperparameters
- Multiclass

Multiclass Classification

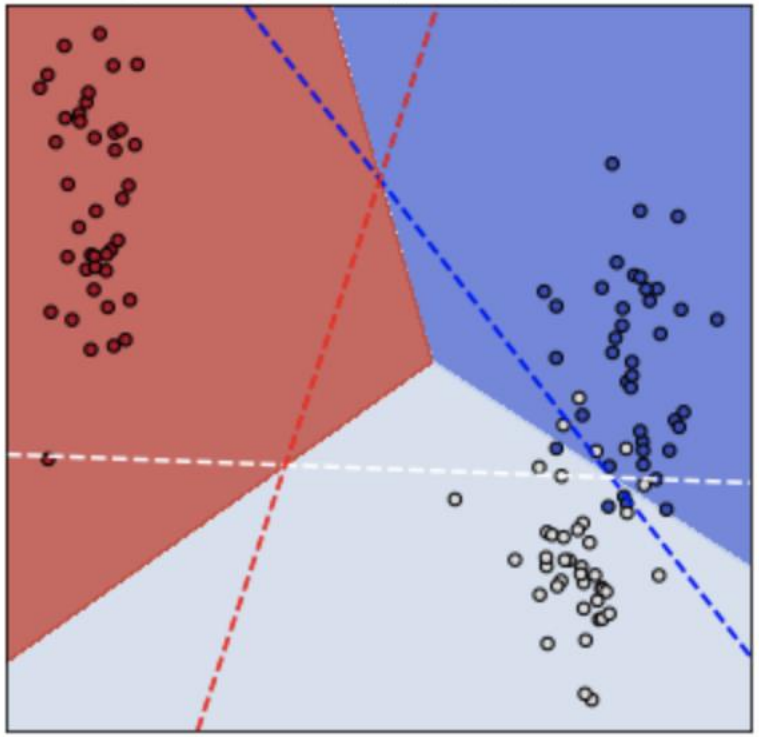
Swiss Institute of Bioinformatics

- If more than two classes there are several options to deal with them:
  - **Ovo** (one-versus-one): split the problem into $K(K-1)/2$ classification tasks. At prediction time we choose the class with the most votes. Implemented in SVC. See also sklearn OneVsOneClassifier.
  - **Ovr** (one-versus-rest): split the problem into K classification tasks. At prediction time we choose the class with the highest prediction score. Implemented in LogisticRegression. See also OneVsRestClassifier.
  - **Inherently multiclass**: Some classifiers have inherent support for multiclass problems (e.g. k-NN, LogisticRegression, DecisionTree, RandomForest)
- Check: https://scikit-learn.org/stable/modules/multiclass.html

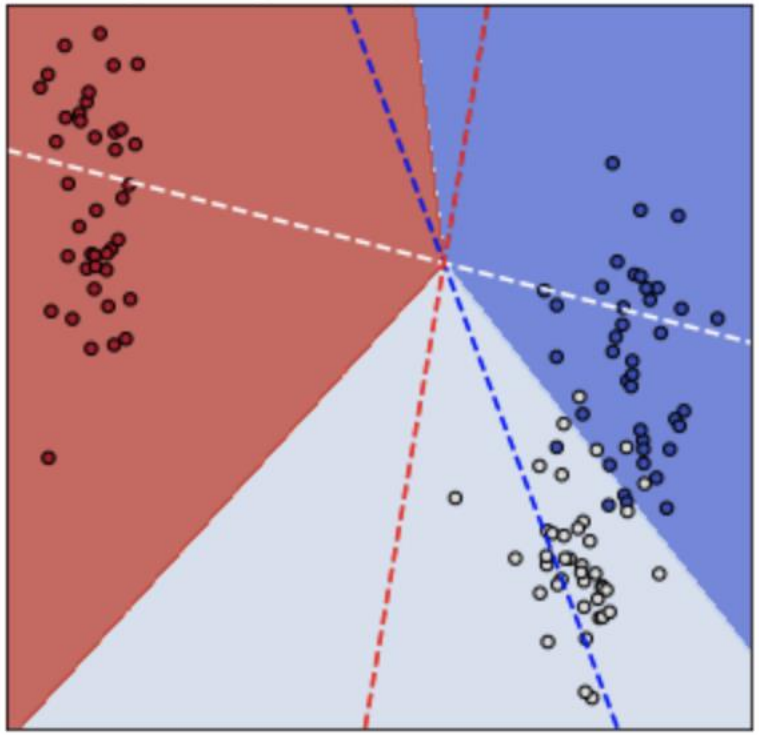
Multiclass Classification

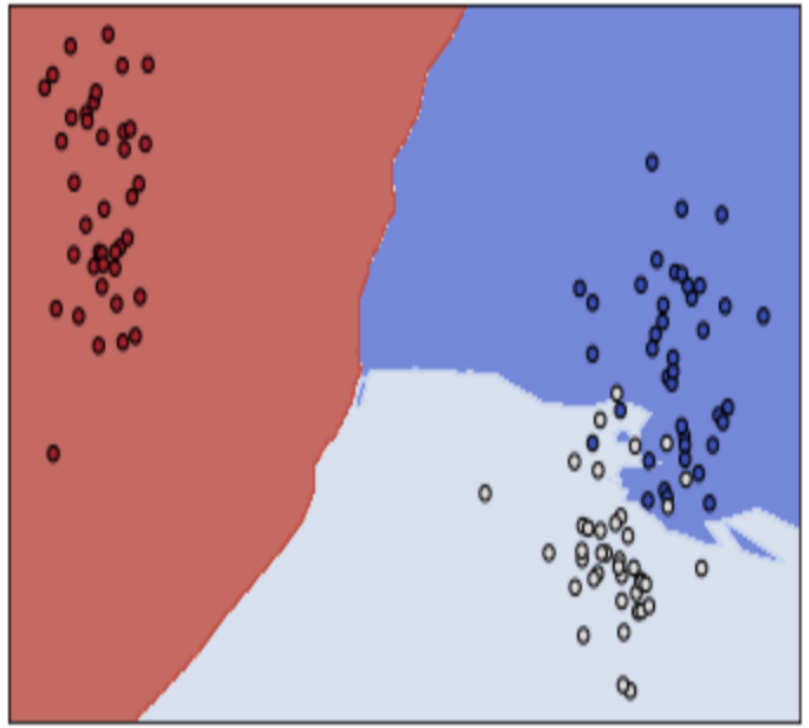
Swiss Institute of Bioinformatics

ovr



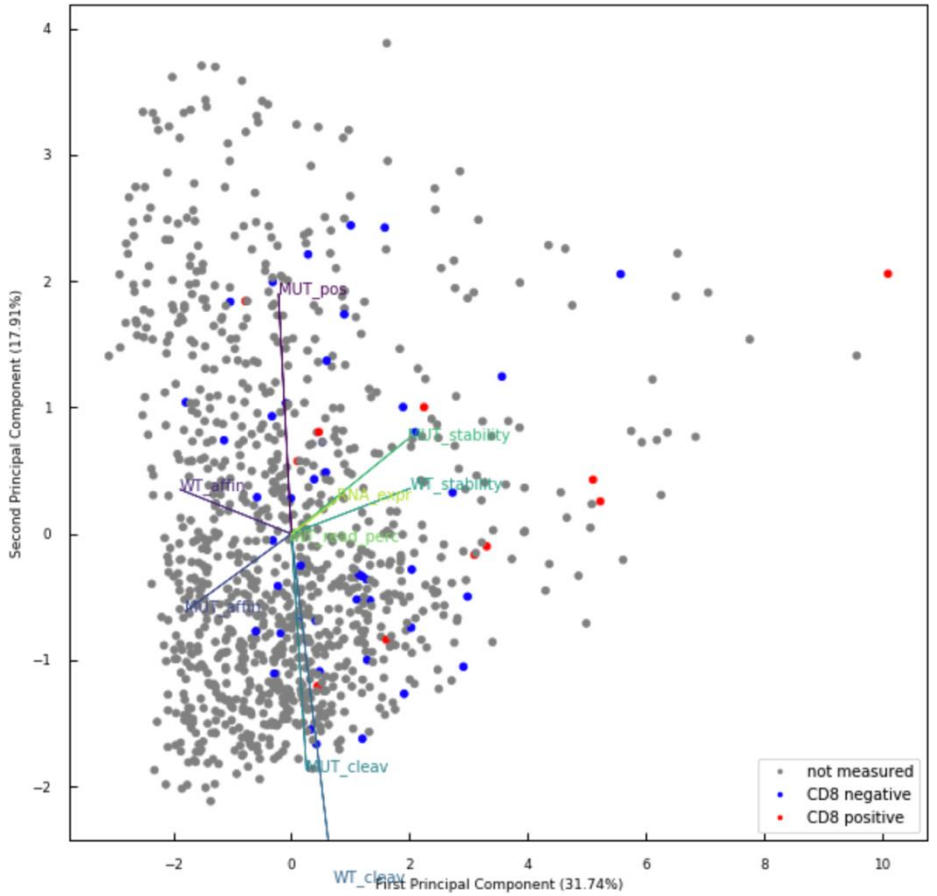
multinomial





Unbalanced Classes

- Very often in ML, there are different number of instances in each class
- This can be a problem for a ML classification method if classes are well mixed
- If '*accuracy*' is used in a CV loop a classifier will try to predict the abundant class correctly, and ignore the rare class.
- Solution: Use 'roc_*auc*' or work with '*class_weight*' parameter.
  - *class_weight*='balanced' : $cw_k = \frac{N}{K \cdot \sum_{i=1}^{N} y_i^k}$
  - Set '*class_weight*' or learn in CV loop



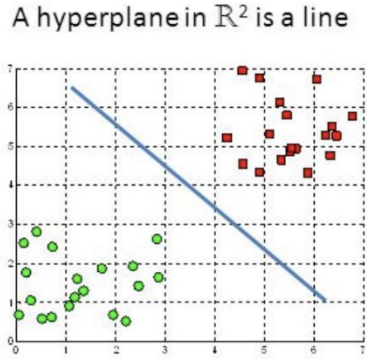
Support Vector Machines (SVM)

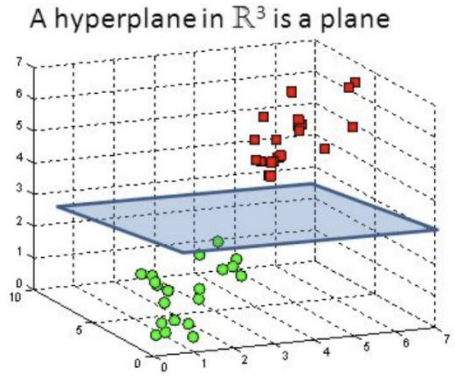
Swiss Institute of Bioinformatics

- Separating hyperplanes
- Soft margin classification
- Linear SVM
- Non-linear SVM & Kernels

Largest Margin Separating Hyperplane

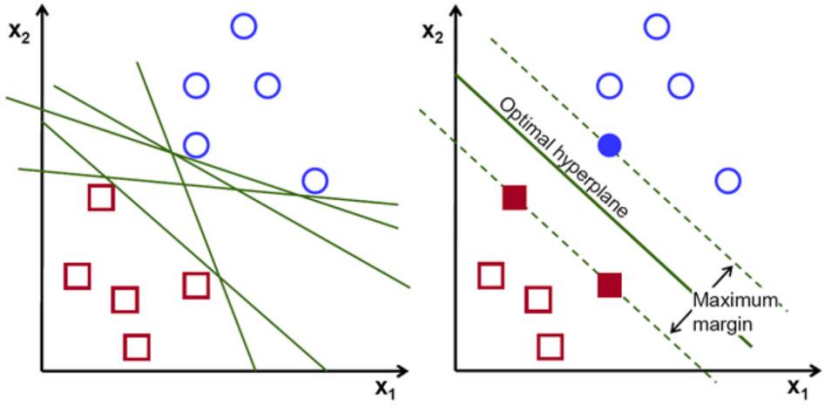
Swiss Institute of Bioinformatics





Hyperplanes are linear decision boundaries that help classify the data points.

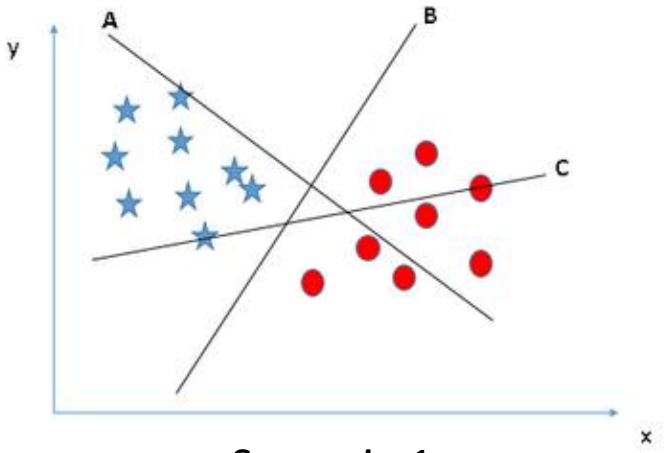
- Objective:
  - find a hyperplane in an M-dimensional space (M = number of features) that distinctly classifies the data points
  - find a plane that has the maximum margin, i.e the maximum distance between data points of both classes



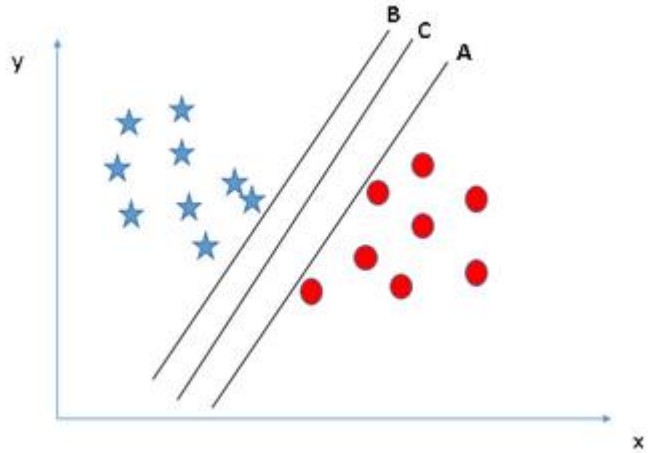
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence

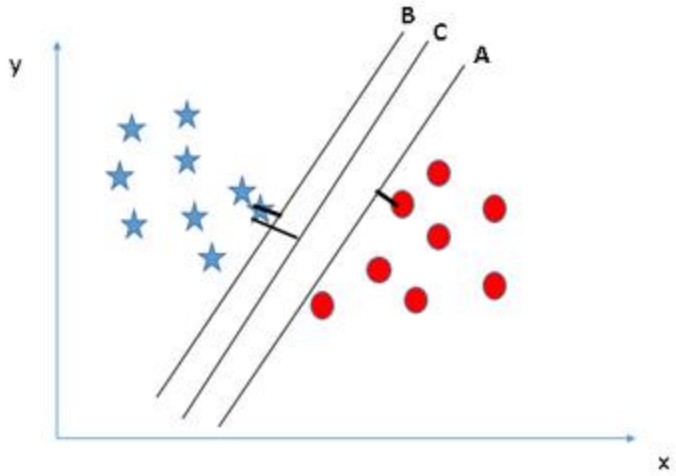
Support Vector Machines (SVM)

Swiss Institute of Bioinformatics

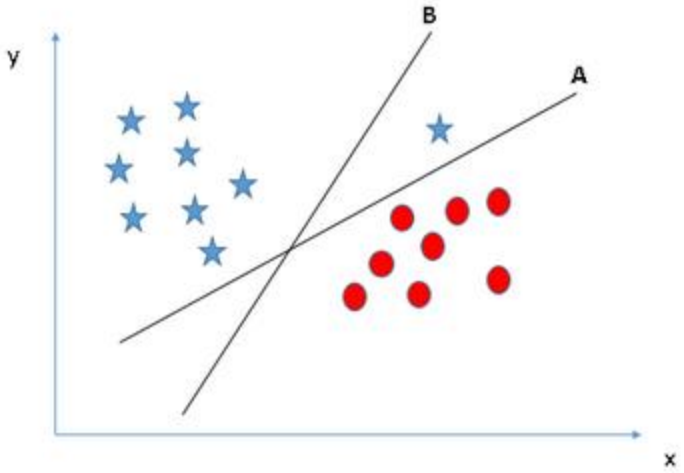


Scenario 1

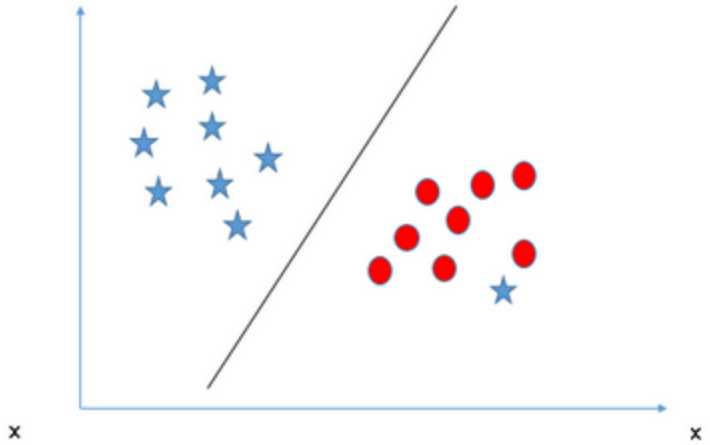




Scenario 2



Scenario 3



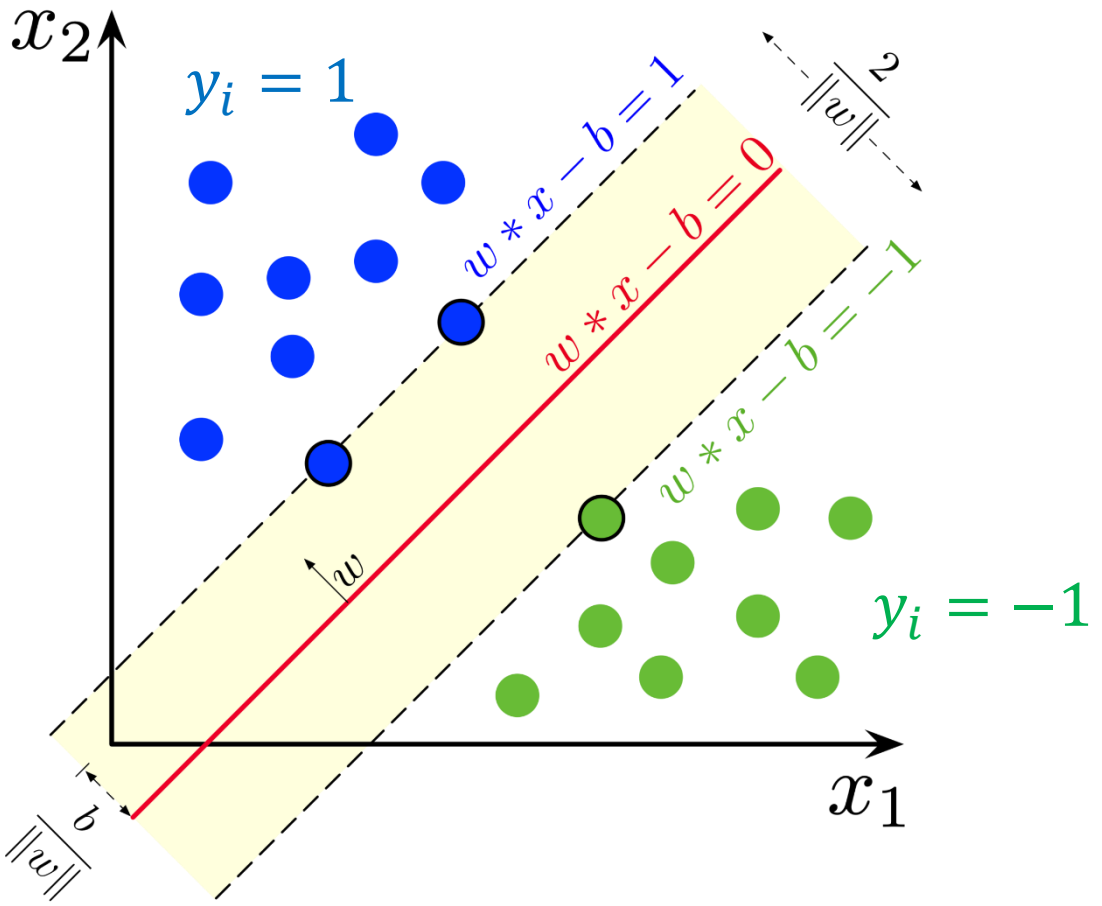
Scenario 4

Largest Margin Separating Hyperplane

Swiss Institute of Bioinformatics

https://en.wikipedia.org/wiki/Support-vector_machine

- Margin $m = \frac{2}{\|\boldsymbol{w}\|}$
- Maximal margin $m \Longrightarrow$ minimal $\|\boldsymbol{w}\|$
- $\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2$
  subject to $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b) \geq 1, i = 1..N$

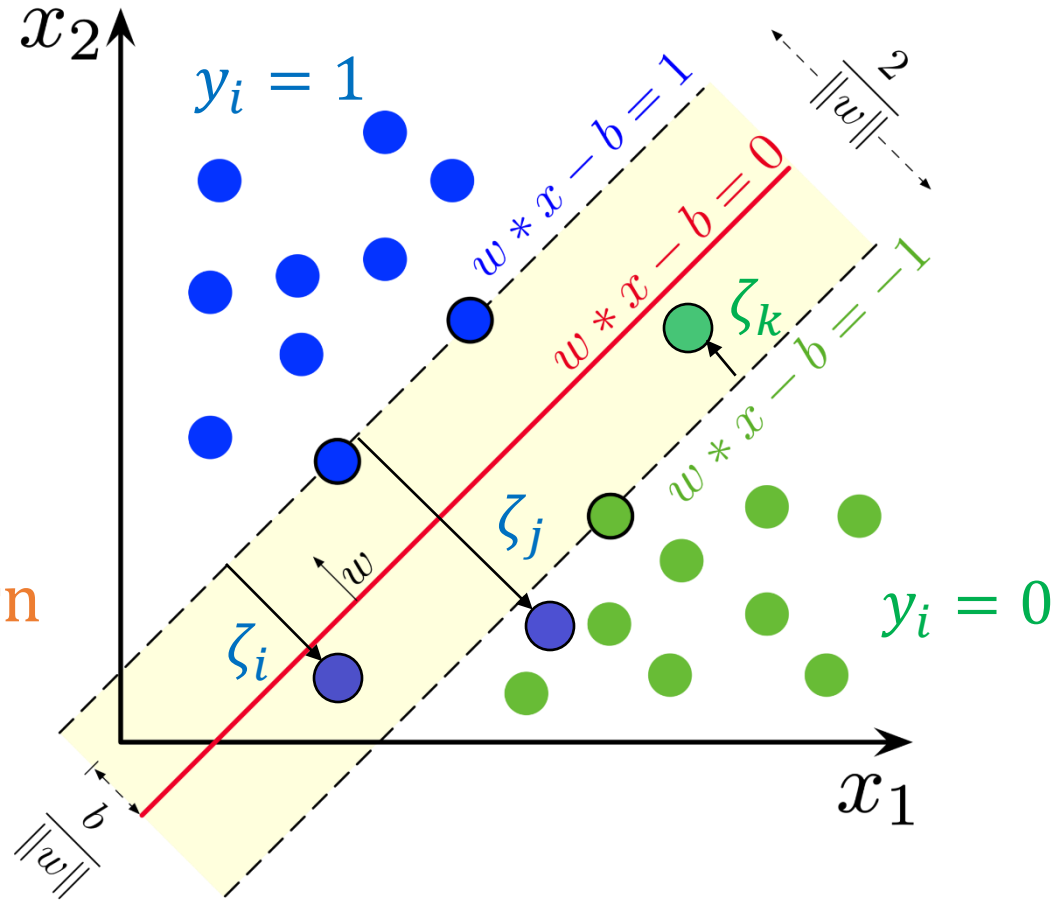


Soft Margin Separating Hyperplane

- $\zeta_i = \max\left(0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b)\right)$
- $\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} \zeta_i \right\}$

  subject to $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b) \geq 1 - \zeta_i, \zeta_i \geq 0$
  $i = 1 \ldots N$

- This is a quadratic programming problem. Efficient algorithms implemented in sklearn LinearSVC can solve this in $O(NM)$
- Non-linear SVM (see below) requires $O(N^{\alpha}M)$, where $2 < \alpha < 3$



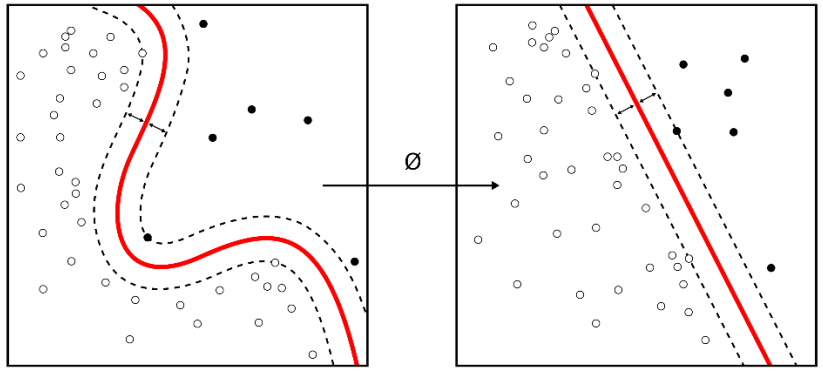
Non-Linear Support Vector Machines

- Dual form: $\min_{\alpha} \left\{ -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \, \boldsymbol{x_i} \cdot \boldsymbol{x_j} + \sum_{i=1}^{N} \alpha_i \right\}$
  subject to:

$$\sum_{i=1}^{N} y_i \alpha_i = 0 \text{ and } 0 \leq \alpha_i \leq C, i = 1 \ldots N.$$

  - $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i; S = \{i | \alpha_i > 0\}; \; b = \frac{1}{|S|} \sum_{i \in S} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)$
  - Prediction: $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x_i} \boldsymbol{x} - b$
  - Kernel trick: $\boldsymbol{x} \cdot \boldsymbol{y} \mapsto \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y})$

- Boser, Guyon & Vapnik, Proc. 5th COLT, 1992



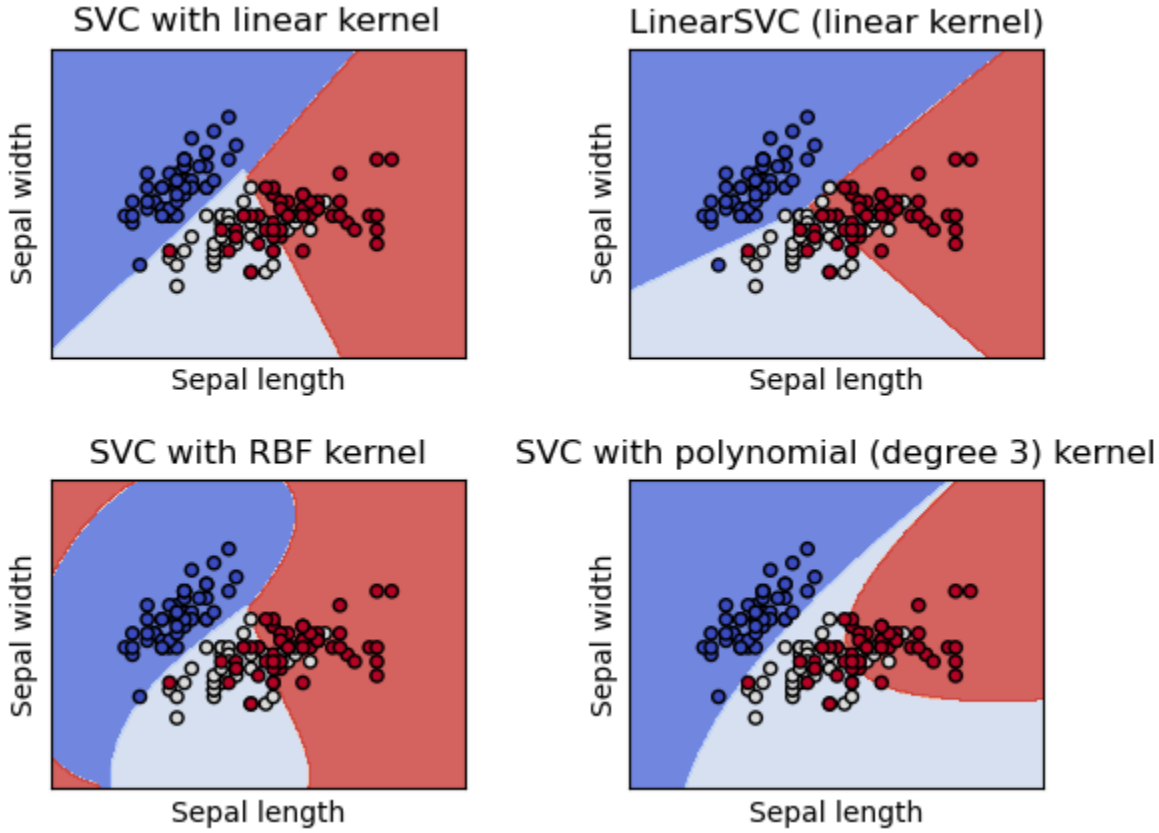
Non-Linear Support Vector Machines

- Kernel trick: $\boldsymbol{x} \cdot \boldsymbol{y} \mapsto \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y})$
  - $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{K}(\boldsymbol{y}, \boldsymbol{x}); \boldsymbol{c}\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j})\boldsymbol{c}^{\boldsymbol{T}} \geq \boldsymbol{0}, \forall \boldsymbol{c} \in \mathbb{R}^{\boldsymbol{n}}, \boldsymbol{x_i}, \boldsymbol{i} = \boldsymbol{1} \ldots \boldsymbol{n} \in \mathbb{R}^{\boldsymbol{m}}$
  - Linear: $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$
  - Polynomial: $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x} \cdot \boldsymbol{y} + r)^{\boldsymbol{d}}$
  - Radial: $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{e}^{-\gamma \|\boldsymbol{x}-\boldsymbol{y}\|^{\boldsymbol{2}}}$
  - Sigmoid: $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\gamma \boldsymbol{x} \cdot \boldsymbol{y} + r)$

Support vector machines (SVM)

Swiss Institute of Bioinformatics

- Hyperparameters
  - Type of kernel (linear, polynomial, RBF; etc.)
  - Parameters of the kernel
  - Type of regularization
  - λ: regularization strength

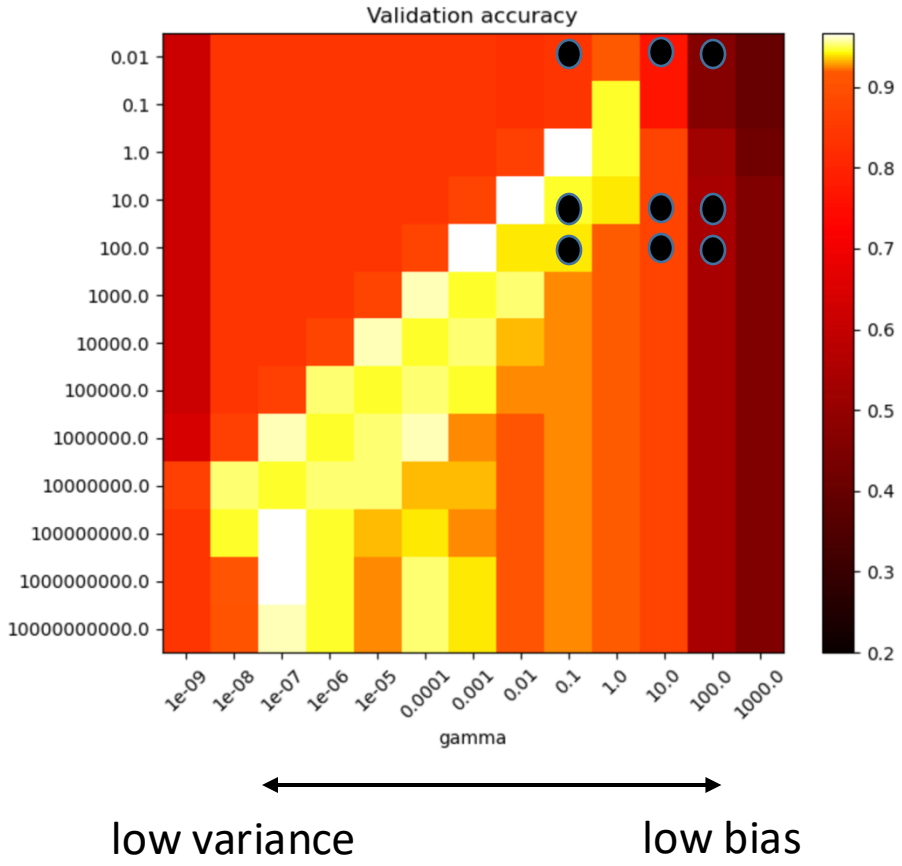


Radial Basis Kernel

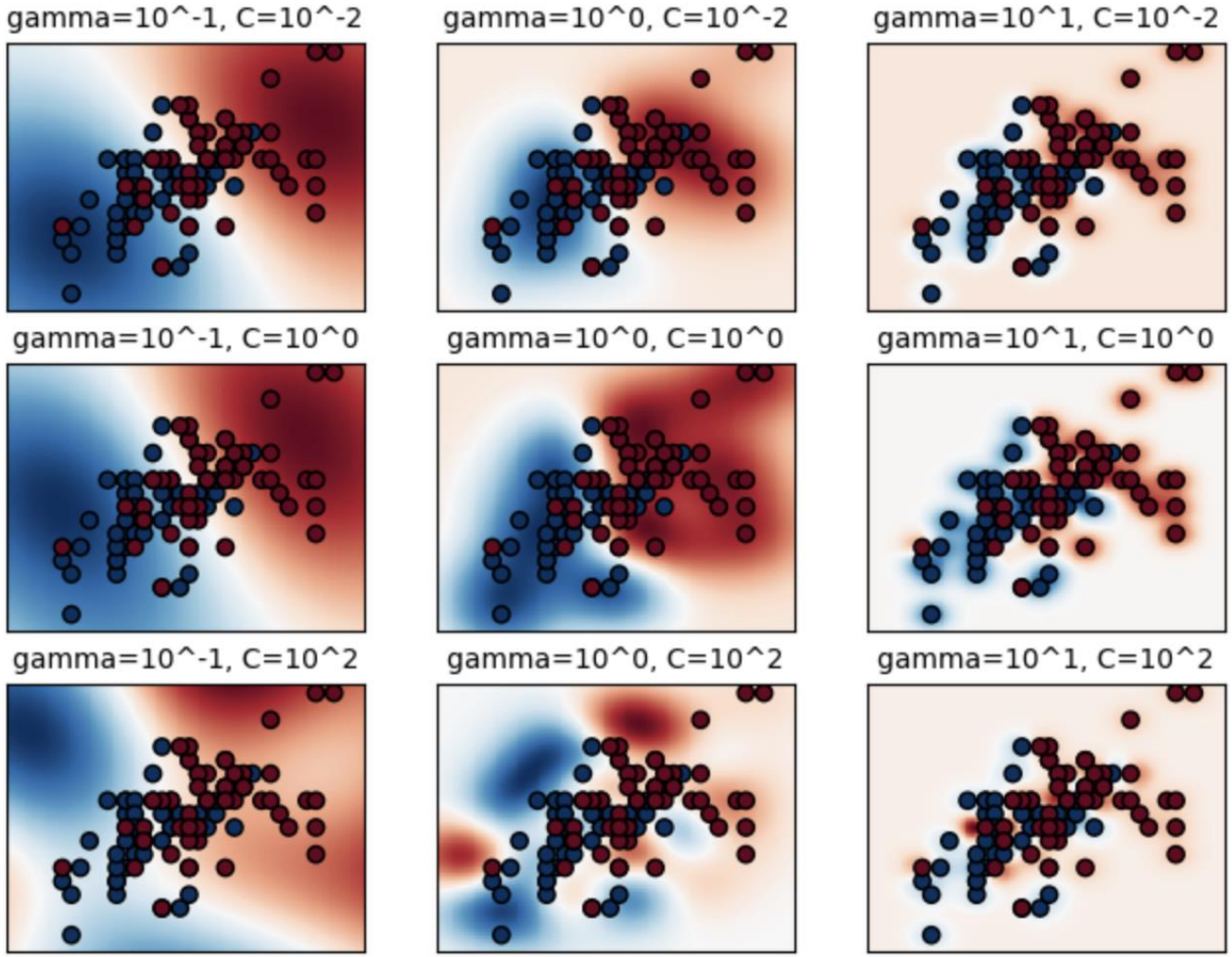
low variance

C

low bias



low variance — low bias



[sklearn SVC radial basis]

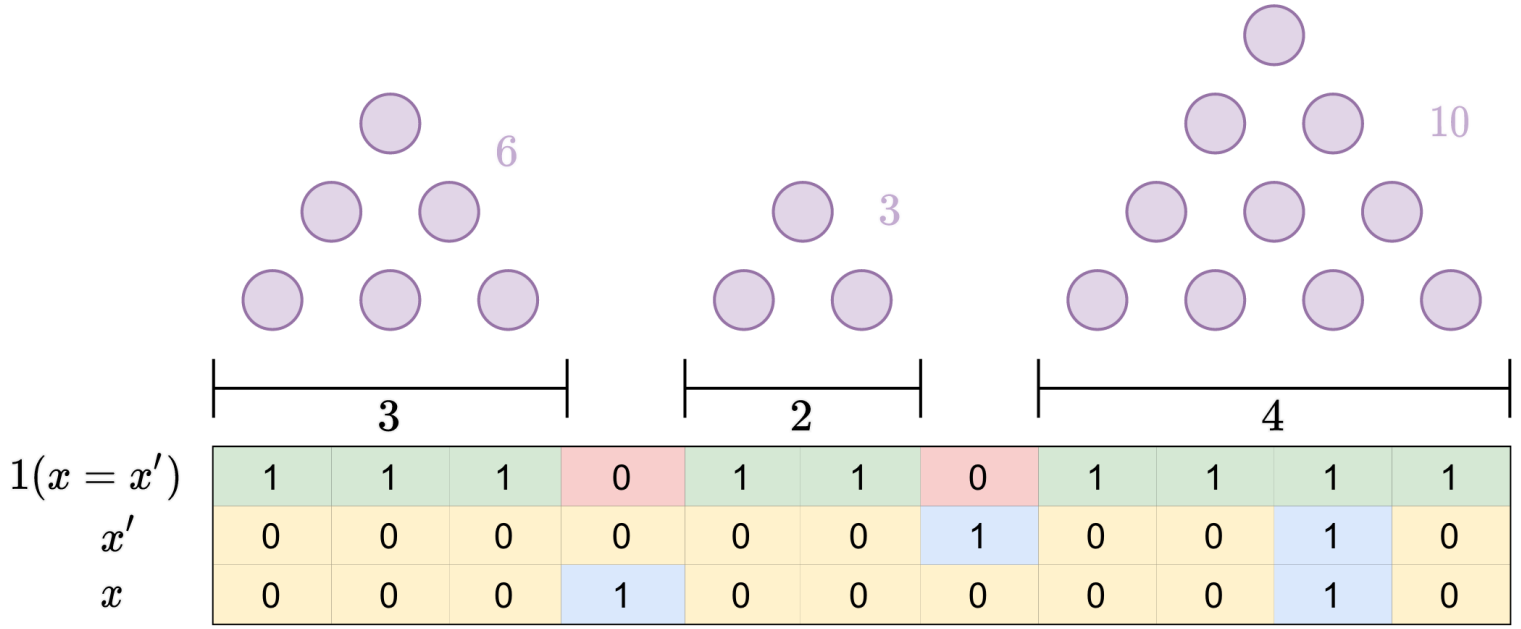
Kernels

- The beauty of SVM's or other kernel based approaches is that one does not need the data matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_N)^{\boldsymbol{T}}$ explicitely, but only the relation between the data points $\boldsymbol{x_i}$ given by the kernel $\boldsymbol{K_{ij}} = \boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j})$
- This can be advantage when dealing with data items that cannot trivially be tranformed to vectors, but for which similarity kernels exist.
- For example, if $\boldsymbol{x_i}$ are sequences, then $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j})$ can be defined by a sequence similarity score, without the need to turn sequences into vectors.

String Kernels (not included in sklearn.svm)

Swiss Institute of Bioinformatics

Example of string kernel from [Hilmarsson et al. BioRxiv, 2021]



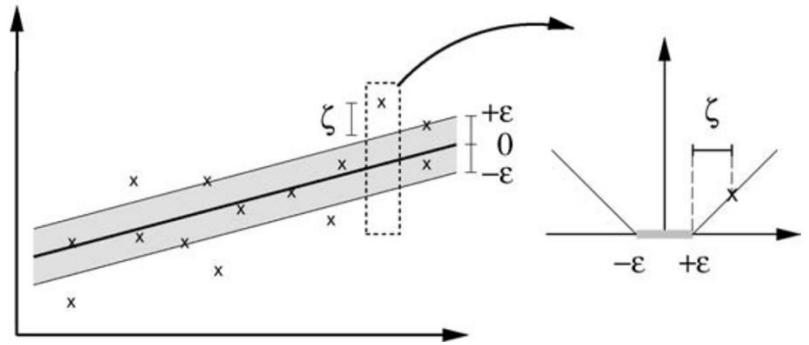
$$K(x, x') = T_3 + T_2 + T_4 = \frac{3(3+1)}{2} + \frac{2(2+1)}{2} + \frac{4(4+1)}{2} = 6 + 3 + 10 = 19$$

Figure 2. String kernel computation with triangular numbers visualized for two sequences, x and x', of length eleven.

Support Vector Regression (SVR)

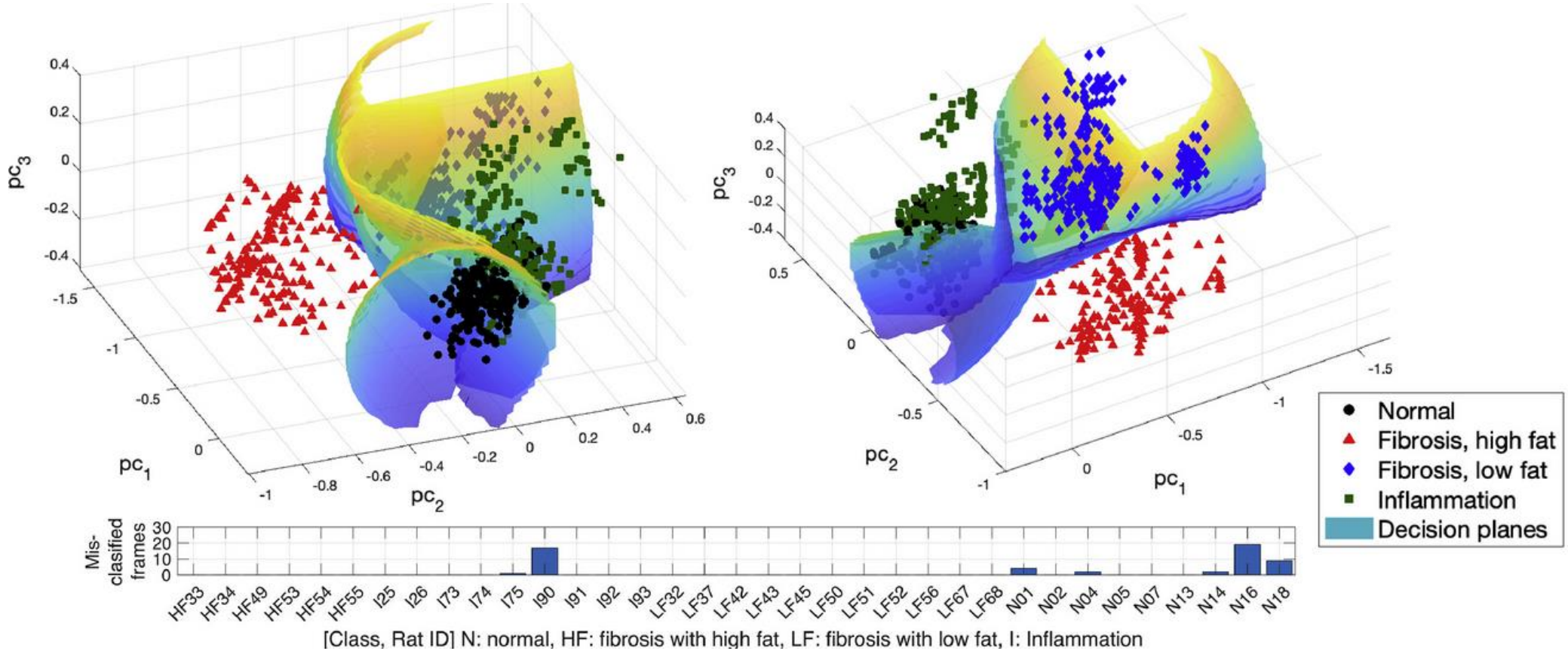
$$\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} (\zeta_i + \zeta_i^*) \right\}$$

subject to $$\begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leq \varepsilon + \zeta_i \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leq \varepsilon + \zeta_i^* , \qquad i = 1 \ldots N \\ \zeta_i, \zeta_i^* \geq 0 \end{cases}$$



For dual form and kernel trick see Smola & Schölkopf, Statistics and Computing, 2004

Support vector machines (SVM)



Baek et al. (2020) Ultrasound Med Biol 46(12):3379-3392.

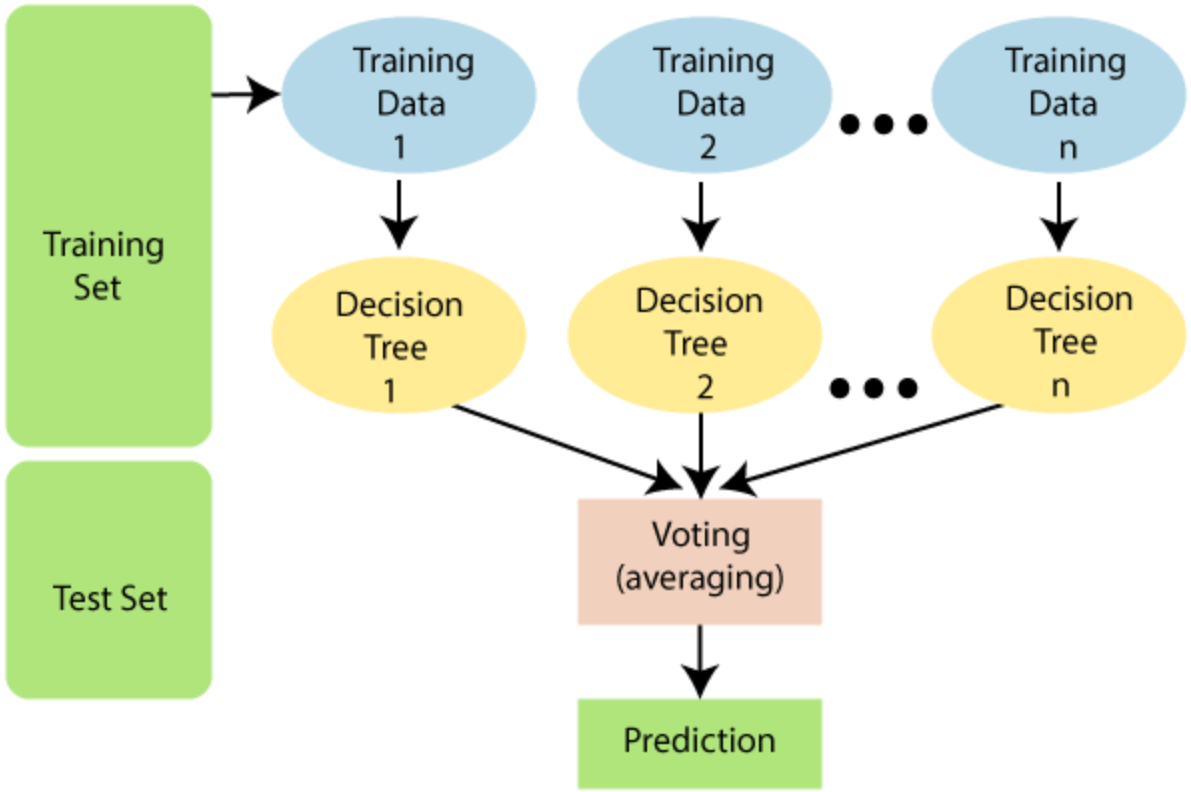
Random forest

Swiss Institute of Bioinformatics

- popular machine learning algorithm which can be used for both Classification and Regression problems in ML
- based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random forest

- Bootstrap (select the features and the training samples at random, with replacement, to create the *n* training datasets)
- Hyperparameters:
  - Same as for the decision trees
  - Number of trees in the forest



Random forest

Swiss Institute of Bioinformatics

› Proc Natl Acad Sci U S A. 2019 Sep 3;116(36):18119-18125. doi: 10.1073/pnas.1813645116. Epub 2019 Aug 16.

Identification of the expressome by machine learning on omics data

Ryan C Sartor [1], Jaclyn Noshay [2], Nathan M Springer [2], Steven P Briggs [3]

Affiliations + expand

PMID: 31420517 PMCID: PMC6731682 DOI: 10.1073/pnas.1813645116

Free PMC article

Abstract

Accurate annotation of plant genomes remains complex due to the presence of many pseudogenes arising from whole-genome duplication-generated redundancy or the capture and movement of gene fragments by transposable elements. Machine learning on genome-wide epigenetic marks, informed by transcriptomic and proteomic training data, could be used to improve annotations through classification of all putative protein-coding genes as either constitutively silent or able to be expressed. Expressed genes were subclassified as able to express both mRNAs and proteins or only RNAs, and CG gene body methylation was associated only with the former subclass. More than 60,000 protein-coding genes have been annotated in the reference genome of maize inbred B73. About two-thirds of these genes are transcribed and are designated the filtered gene set (FGS). Classification of

Isabelle Dupanloup, BCF

Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics

- AdaBoost algorithms can be used for both classification and regression problem
- Combine multiple “weak classifiers” into a single “strong classifier”
- Based on the concept of **ensemble learning**: power of ensembling is such that we can still build powerful ensemble models even when the individual models in the ensembles are extremely simple.
- Boosting algorithms try to build a strong learner (predictive model) from the mistakes of several weaker models.
- Boosting basically tries to reduce the bias error which arises when models are not able to identify relevant trends in the data. This happens by evaluating the difference between the predicted value and the actual value.
- You start by creating a model from the training data. Then, you create a second model from the previous one by trying to reduce the errors from the previous model. Models are added sequentially, each correcting its predecessor, until the training data is predicted perfectly or the maximum number of models have been added.

Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics

- The weak learners in AdaBoost are decision trees with a single split, called **decision stumps**.
- Stumps alone are not a good way to make decisions. A full-grown tree combines the decisions from all variables to predict the target value. A stump, on the other hand, can only use one variable to make a decision
- Difference between Random Forest and AdaBoost:
  - AdaBoost uses a forest of stumps rather than trees.
  - AdaBoost stumps have different contributions to the prediction.
  - In AdaBoost, models are added sequentially, each correcting its predecessor.
- Each stump is made by taking the mistakes of the previous stumps into account by applying an importance weight on each sample. At the beginning of the algorithm, all the samples have the same weight but then after the first stump, the missclassified points will have a higher weight so that the next stumps really focuses on predicting them well.

Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics

1. A weak classifier (e.g. a decision stump) is made on top of the training data based on the weighted samples. Here, the weights of each sample indicate how important it is to be correctly classified. Initially, for the first stump, we give all the samples equal weights.
2. We create a decision stump for each variable and see how well each stump classifies samples to their target classes.
3. More weight is assigned to the incorrectly classified samples so that they're classified correctly in the next decision stump. Weight is also assigned to each classifier based on the accuracy of the classifier, which means high accuracy = high weight !
4. Reiterate from Step 2 until all the data points have been correctly classified, or the maximum iteration level has been reached.

Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics

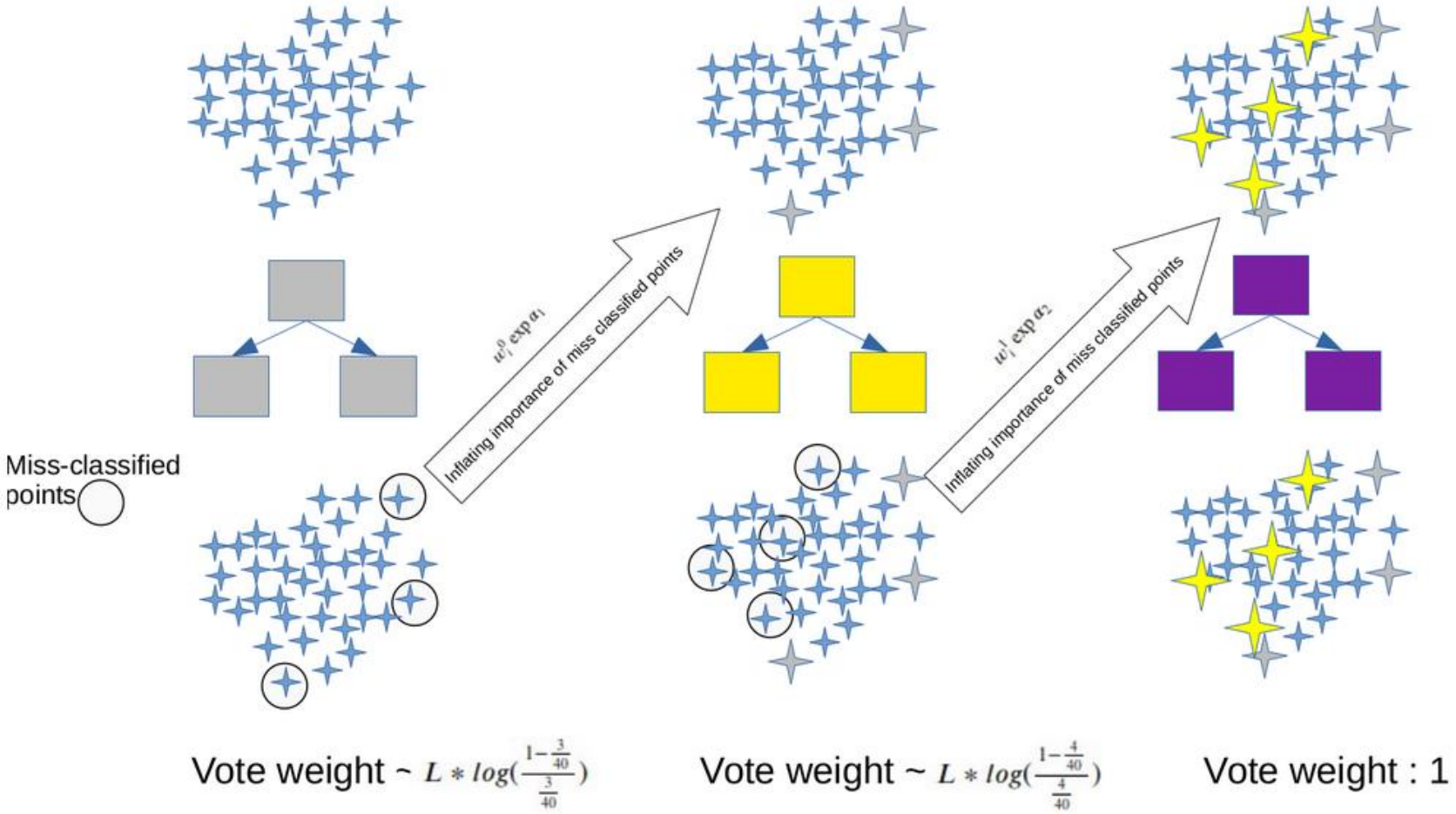
- Misclassification rate is calculated for each model:

$$\propto = L * \ln(\frac{1-error}{error}) \text{ with L = learning rate and error = } \frac{\text{sum(w(i) * terror(i))}}{\text{sum(w(i))}}$$

$$w_i^{t+1} = w_i^t \exp \alpha_{t+1} \text{ if } stump_{t+1} \text{ made a mistake for point i}$$

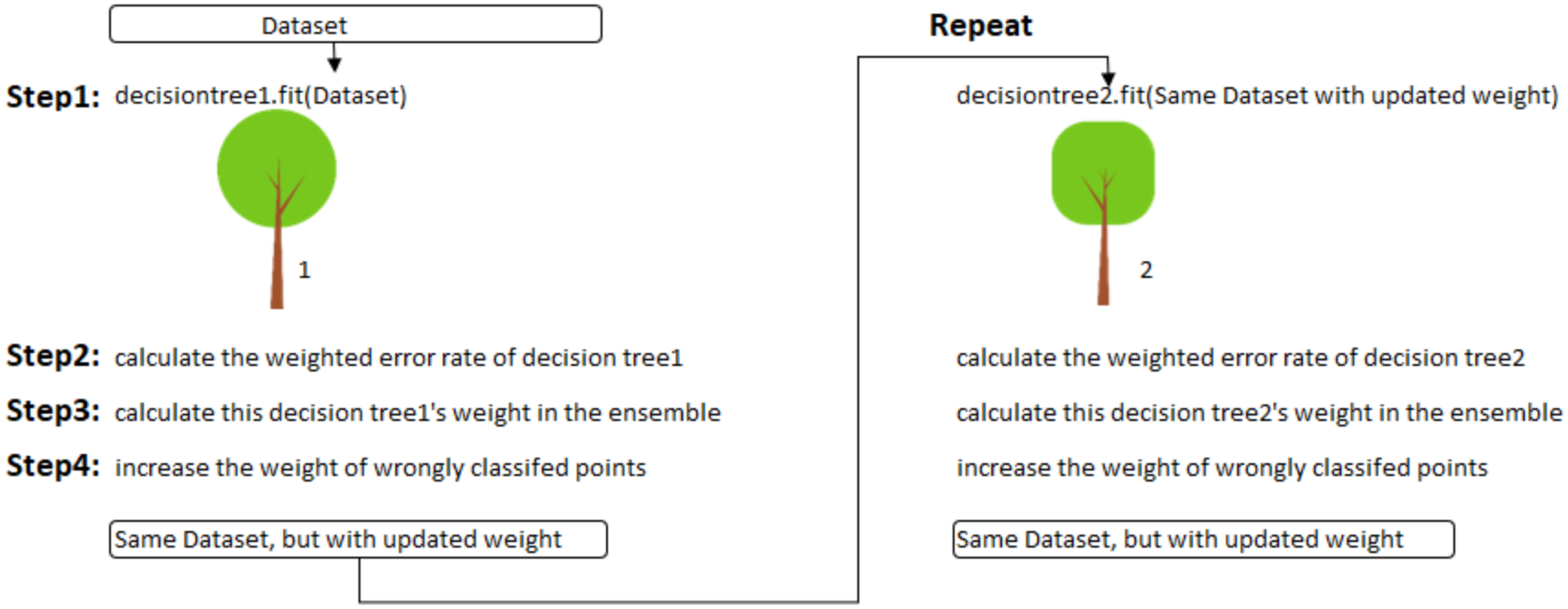
- Predictions are made by calculating the weighted average of the weak classifiers.
- Hyperparameters:
  - Number of stumps
  - Learning rate L: weight applied to each classifier at each boosting iteration
- Need high quality data !

Adaptive Boosting (AdaBoost)



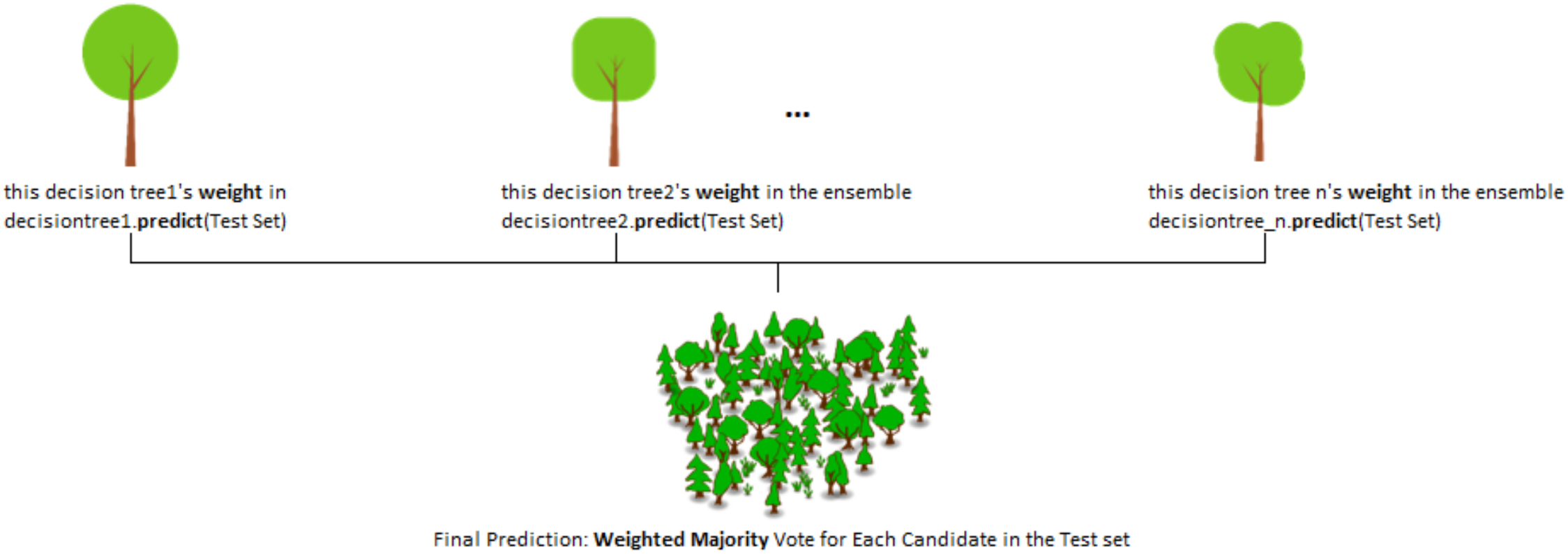
Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics



Adaptive Boosting (AdaBoost)

Swiss Institute of Bioinformatics



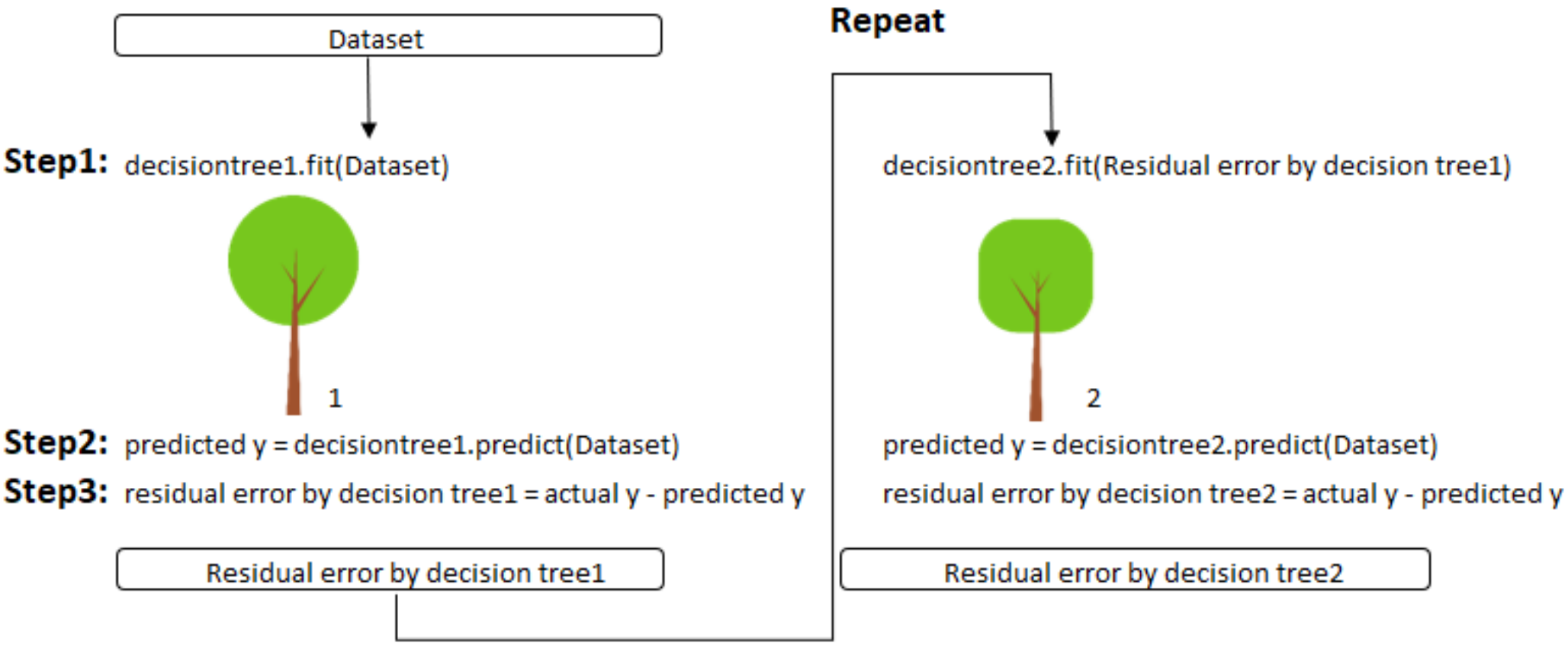
Gradient Boosting

Swiss Institute of Bioinformatics

- Another boosting model. Remember, boosting model's key is learning from the previous mistakes.
- Can be used for both classification and regression problem
- Gradient boosting involves three elements:
    1. A loss function to be optimized.
    2. A weak learner to make predictions: decision trees (with 4-to-8 levels)
    3. An additive model to add weak learners to minimize the loss function:
        - Trees are added one at a time, and existing trees in the model are not changed.
        - A gradient descent procedure is used to minimize the loss when adding trees.

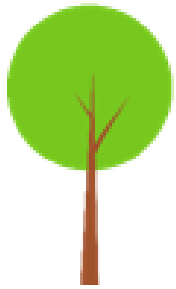
Gradient Boosting

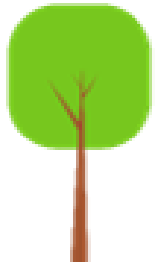
Swiss Institute of Bioinformatics



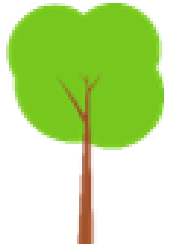
Gradient Boosting

Swiss Institute of Bioinformatics





...



decisiontree1.**predict**(Test Set)

decisiontree2.**predict**(Test Set)

decisiontree_n.**predict**(Test Set)

Prediction for sample *i*: 0.2

Prediction for sample *i*: 0.6

Learning rate = 1 Prediction for sample *i*: 0.2+1*0.6=0.8

Learning rate = 0.1 Prediction for sample *i*: 0.2+0.1*0.6=0.26

Final Prediction: **Sum of** Prediction of tree1,tree2...tree n for Each Candidate in the Test set

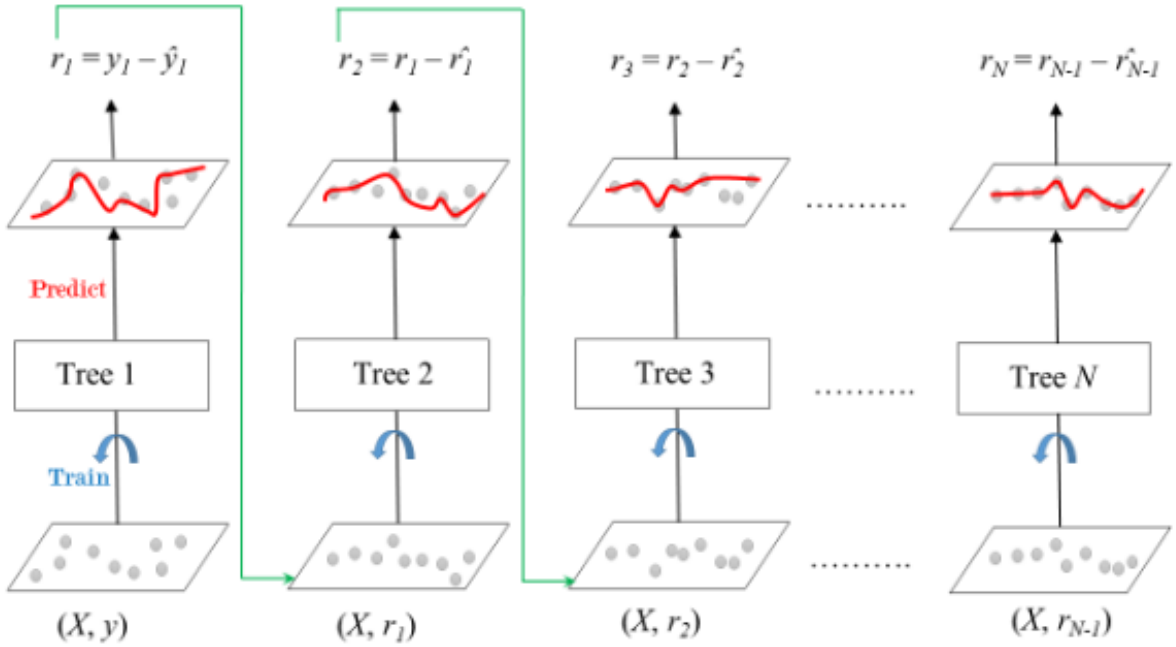
Isabelle Dupanloup, BCF

Gradient Boosting

Swiss Institute of Bioinformatics

- The intuition behind gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better
- Once we reach a stage that residuals do not have any pattern that could be modeled, we can stop modeling residuals (otherwise it might lead to overfitting). Algorithmically, we are minimizing our loss function, such that test loss reach its minima.

Gradient Boosting



- The ensemble consists of *N* trees. Tree1 is trained using the feature matrix *X* and the labels *y*. The predictions labelled *y1(hat)* are used to determine the training set residual errors *r1*. Tree2 is then trained using the feature matrix *X* and the residual errors *r1* of Tree1 as labels. The predicted results *r1(hat)* are then used to determine the residual *r2*. The process is repeated until all the *N* trees forming the ensemble are trained.

Gradient Boosting

Swiss Institute of Bioinformatics

- Gradient boosting is a greedy algorithm and can overfit a training dataset quickly.
- It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.
- Enhancements to basic gradient boosting:
  - Tree Constraints (small number of trees, short trees)
  - Shrinkage (learning rate)
  - Random sampling (bootstrap the training dataset)
- Hyperparameters:
  - Loss function: deviance (= in logistic regression) or exponential (= in AdaBoost)
  - Learning rate: shrinks the contribution of each tree
  - Number of trees, trees length, minimum number of samples required to split an internal node, minimum number of samples required to be at a leaf node
  - Number of features to consider when looking for the best split, etc.

Naive Bayes

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Swiss Institute of Bioinformatics

- classification technique based on Bayes' Theorem with an assumption of independence among predictors (= features)
- a classification problem may have *k* class labels *y1, y2, …, yk* and *n* input variables, *X1, X2, …, Xn*. We can calculate the conditional probability for a class label with a given instance or set of input values for each column *x1, x2, …, xn* as follows: P(yi | x1, x2, …, xn)
- The conditional probability can then be calculated for each class label in the problem and the label with the highest probability can be returned as the most likely classification.

P(yi | x1, x2, …, xn) = P(x1, x2, …, xn | yi) * P(yi) / P(x1, x2, …, xn)

Independence between features

P(x1|yi) * P(x2|yi) * … P(xn|yi)

estimated from the data

prior *P(yi)* easy to estimate from the data

constant used in calculating the conditional probability of each class for a given instance

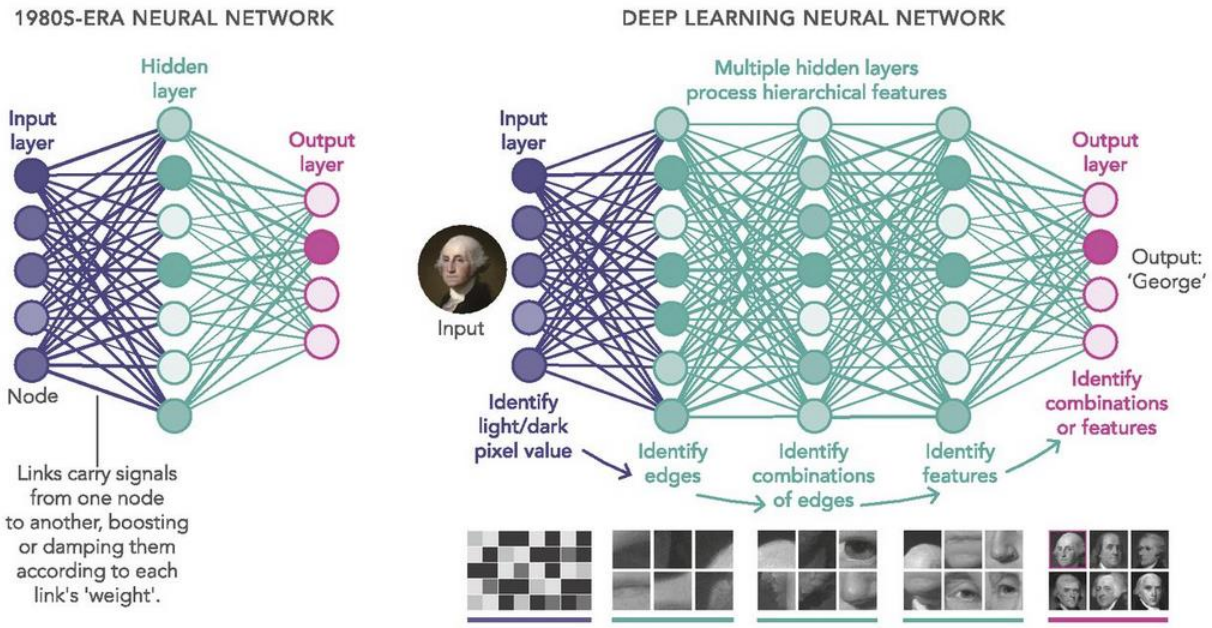
Naive Bayes

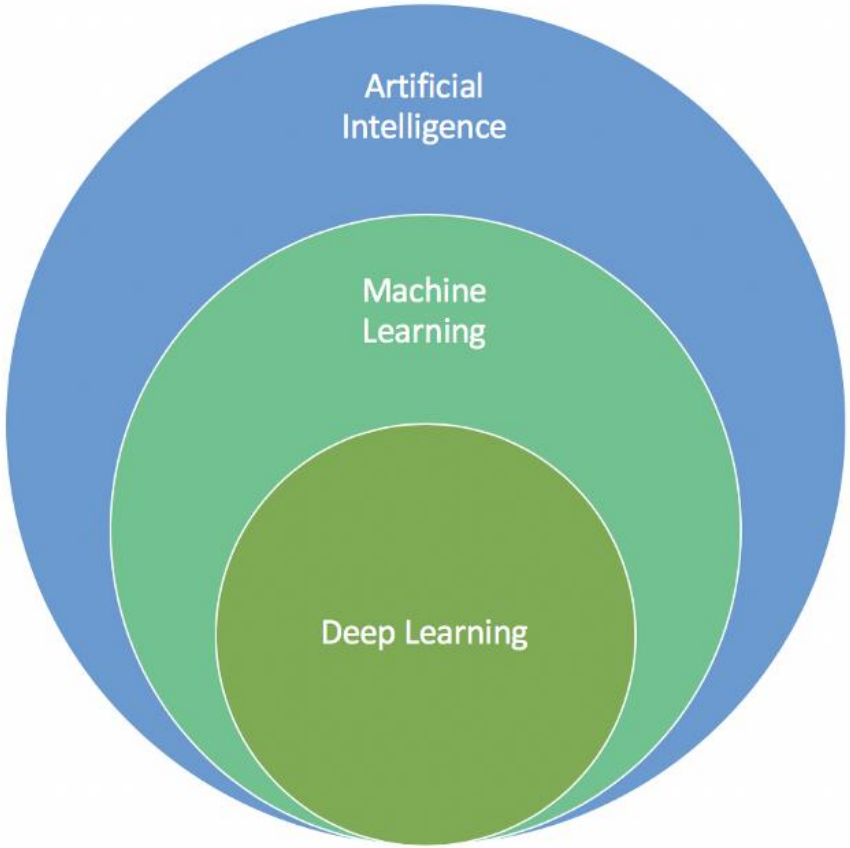
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Swiss Institute of Bioinformatics

- Advantages:
  - very easy to build and particularly useful for very large data sets.
  - outperform even highly sophisticated classification methods.
  - a good choice when CPU and memory resources are a limiting factor.
  - a good method if something fast and easy that performs pretty well is needed.
- Main disadvantage: does not consider the interactions between features.
- Naive Bayes has been used in real-world applications such as:
  - mining housekeeping genes
  - genetic association studies
  - discovering Alzheimer genetic biomarkers from whole genome sequencing (WGS) data

Deep learning





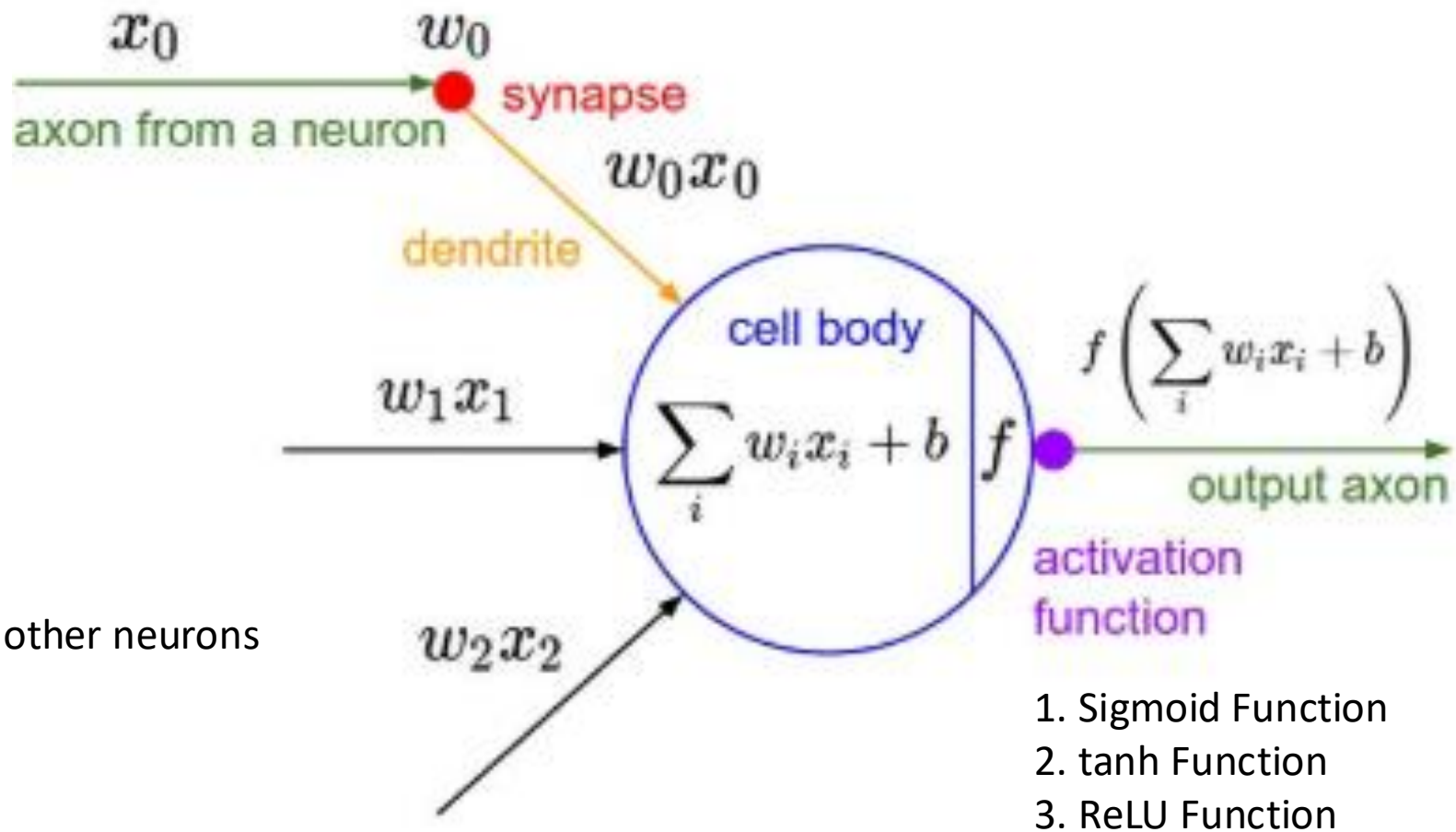
Isabelle Dupanloup, BCF

Deep learning

Swiss Institute of Bioinformatics

- Machine Learning consists of two steps:
  1. specify a template
  2. find the best parameters for that template.
- Deep Learning is simply a subset of the architectures (or templates) that employs “neural networks” which we can specify during Step 1.
- “Neural networks” (more specifically, artificial neural networks) are loosely based on how our human brain works, and the basic unit of a neural network is a neuron.
- At the basic level, a neuron does two things:
  1. Receive input from other neurons and combine them together
  2. Perform some kind of transformation to give the neuron’s output

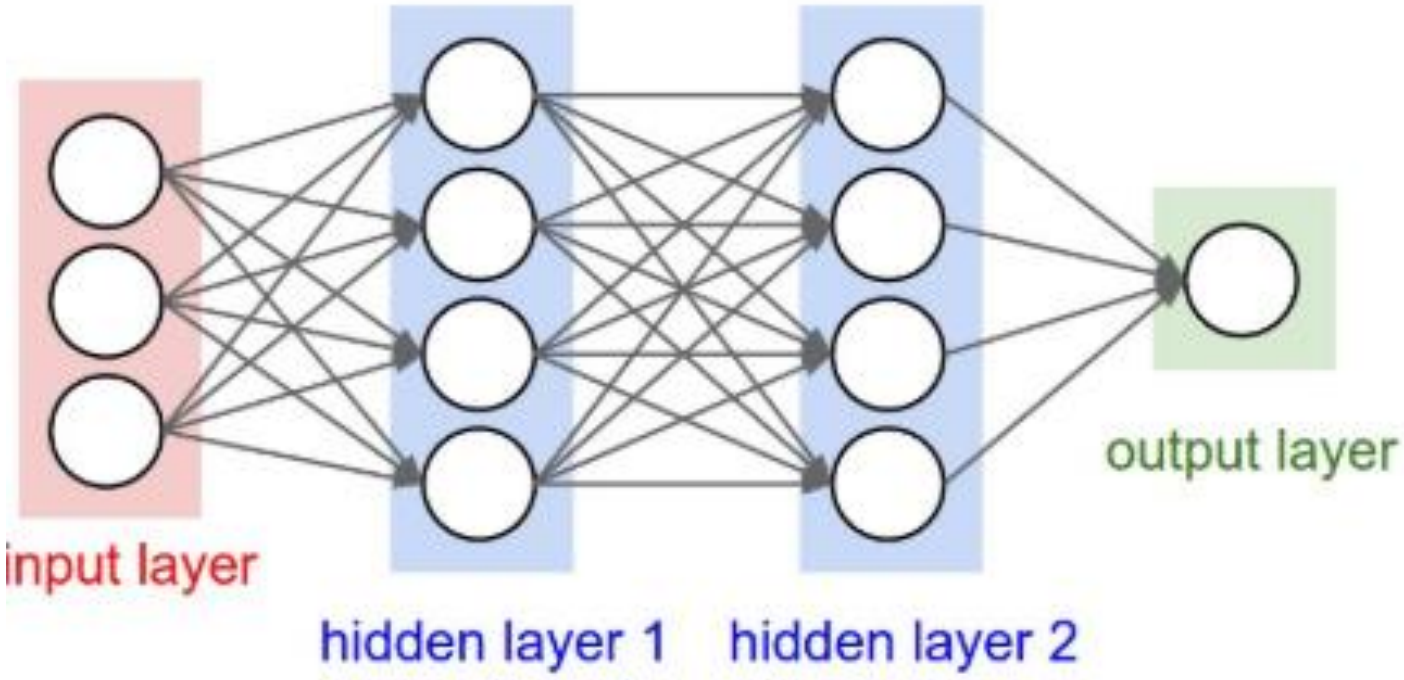
Neural networks



xi: input from other neurons
wi: weights
b: bias

1. Sigmoid Function
2. tanh Function
3. ReLU Function

Neural networks



A neural network is simply a complicated ‘template’ we specify which turns out to have the flexibility to model many complicated relationships between input and output.

Deep learning

Swiss Institute of Bioinformatics

- Model performance assessed with a loss function
- Minimizing the loss function using gradient descent (or other strategies)
- How to choose the best architecture of a neural network (hyperparameters: number of hidden layers, number of neurons in each hidden layer, type of activation functions etc.)
    1. Specify some hyper-parameters (the template)
    2. Train on the training dataset (filling in the parameters)
    3. Record the validation loss
    4. Repeat Steps 1 to 3 with a different set of hyper-parameters (many times)
    5. Pick the model with the lowest validation loss as our best model
    6. Run the chosen model on a separate test dataset to assess performance

Deep learning

Swiss Institute of Bioinformatics

Deep Learning in Omics Data Analysis and Precision Medicine

Jordi Martorell-Marugán [1], Siham Tabik [2], Yassir Benhammou [2], Coral del Val [2], Igor Zwir [2], Francisco Herrera [2], Pedro Carmona-Sáez [1]
Holger Husi [1], editors.

In: Computational Biology [Internet]. Brisbane (AU): Codon Publications; 2019 Nov 21. Chapter 3.

Affiliations + expand

PMID: 31815397 Bookshelf ID: NBK550335 DOI: 10.15586/computationalbiology.2019.ch3
Free Books & Documents

Excerpt

The rise of omics techniques has resulted in an explosion of molecular data in modern biomedical research. Together with information from medical images and clinical data, the field of omics has driven the implementation of personalized medicine. Biomedical and omics datasets are complex and heterogeneous, and extracting meaningful knowledge from this vast amount of information is by far the most important challenge for bioinformatics and machine learning researchers. In this context, there is an increasing interest in the potential of deep learning (DL) methods to create predictive models and to identify complex patterns from these large datasets. This chapter provides an overview of the main applications of DL methods in biomedical research, with focus on omics data analysis and precision medicine applications. DL algorithms and the most popular architectures are introduced first. This is followed by a review of some of the main applications and problems approached by DL in omics data and medical image analysis. Finally, implementations for improving the diagnosis, treatment, and classification of complex diseases are discussed.

Isabelle Dupanloup, BCF

Deep learning

Swiss Institute of Bioinformatics

› Sci Rep. 2021 Apr 26;11(1):8992. doi: 10.1038/s41598-021-88172-0.

Machine learning and deep learning to predict mortality in patients with spontaneous coronary artery dissection

Chayakrit Krittanawong [1 2], Hafeez Ul Hassan Virk [3], Anirudh Kumar [4], Mehmet Aydar [5], Zhen Wang [6 7], Matthew P Stewart [8 9], Jonathan L Halperin [10]

Affiliations + expand

PMID: 33903608 PMCID: PMC8076284 DOI: 10.1038/s41598-021-88172-0

Free PMC article

Abstract

Machine learning (ML) and deep learning (DL) can successfully predict high prevalence events in very large databases (big data), but the value of this methodology for risk prediction in smaller cohorts with uncommon diseases and infrequent events is uncertain. The clinical course of spontaneous coronary artery dissection (SCAD) is variable, and no reliable methods are available to predict mortality. Based on the hypothesis that machine learning (ML) and deep learning (DL) techniques could enhance the identification of patients at risk, we applied a deep neural network to information available in electronic health records (EHR) to predict in-hospital mortality in patients with SCAD. We extracted patient data from the EHR of an extensive urban health system and applied several ML and DL models using candidate clinical variables potentially associated with mortality. We partitioned the data into training and evaluation sets with cross-validation. We estimated model performance based on the area under the receiver-operator characteristics curve (AUC) and balanced accuracy. As sensitivity analyses, we examined

Isabelle Dupanloup, BCF

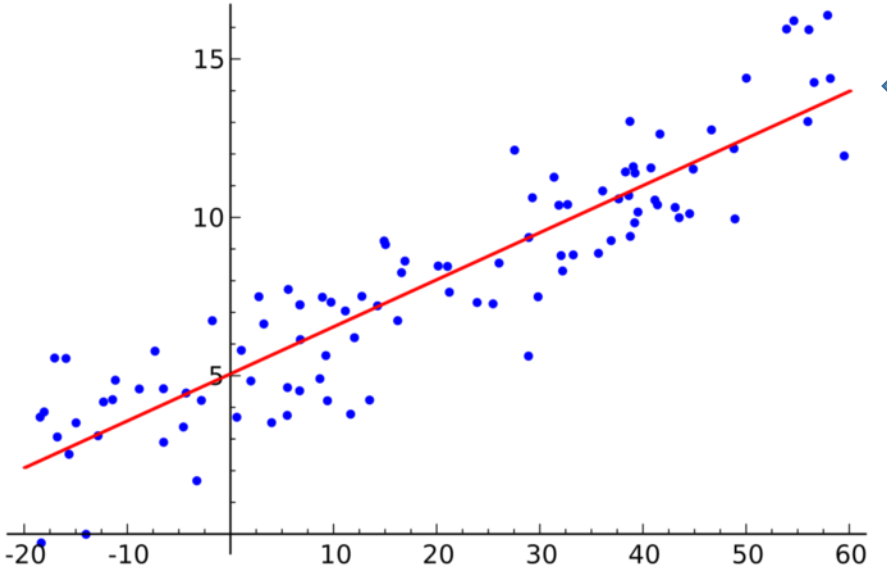
Miscellaneous

Swiss Institute of Bioinformatics

- **NumPy** stands for Numerical Python
  - It is a Python library used for working with arrays.
  - It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
  - NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- **Matplotlib** is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- **Seaborn** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Scikit-learn** (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Linear regression

- Regression: method of modelling a target value based on independent predictors
- Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.
- Simple linear regression: type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent(x) and dependent(y) variable.



Best fit straight line to the data points

$y = a + b*x$

How to find the best values for a and b ?

Least Square Linear Regression

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{NM} \end{pmatrix}$$, data matrix $\boldsymbol{X}$

$\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_M)^{\boldsymbol{T}}$, coefficients for linear regression

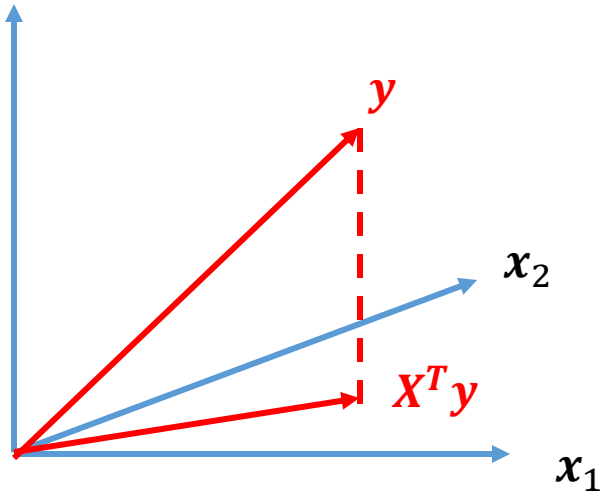
$$\boldsymbol{X} \cdot \boldsymbol{\beta} = \begin{pmatrix} \beta_0 + \sum_{i=1}^{M} x_{1i}\beta_i \\ \vdots \\ \beta_0 + \sum_{i=1}^{M} x_{Ni}\beta_i \end{pmatrix} \sim \boldsymbol{y}$$

$$\mathbf{RSS}(\boldsymbol{\beta}) = (\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) \Longrightarrow \boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \mathbf{RSS}(\boldsymbol{\beta})$$

Least Square Linear Regression

$$\frac{\partial \mathbf{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 2\boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) = \mathbf{0}$$

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$: Pseudo-inverse

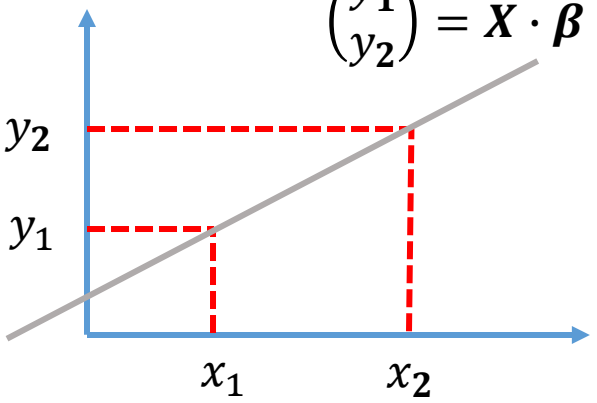


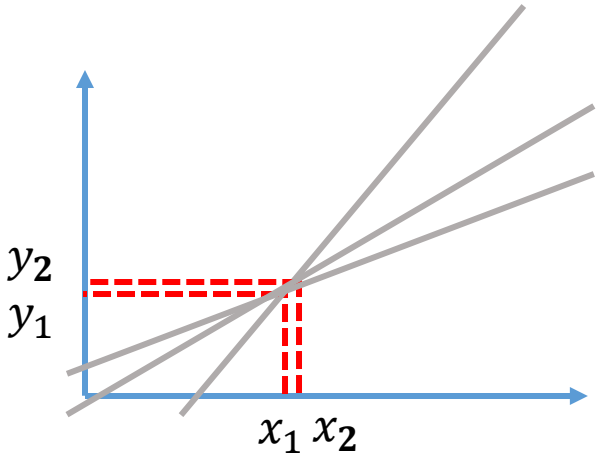
sklearn.linear_model.LinearRegression

- This approach can be problematic if the features are strongly correlated. Then $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ can become close to singular and the calculation of $\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-1}$ unstable leading to large fluctuations in the values of $\boldsymbol{\beta}$ and unstable solutions.

Least Square Linear Regression

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \boldsymbol{X} \cdot \boldsymbol{\beta} = \begin{pmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \end{pmatrix}$$





$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \Rightarrow \boldsymbol{X^T X} = \begin{pmatrix} 1 & 1 \\ x_1 & x_2 \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} = \begin{pmatrix} 2 & x_1 + x_2 \\ x_1 + x_2 & x_1^2 + x_2^2 \end{pmatrix} \Rightarrow$$

$$\left(\boldsymbol{X^T X}\right)^{-1} = \frac{1}{2(x_1^2 + x_2^2) - (x_1 + x_2)^2} \begin{pmatrix} x_1^2 + x_2^2 & -x_1 - x_2 \\ -x_1 - x_2 & 2 \end{pmatrix}$$

$$= \frac{1}{(x_1 - x_2)^2} \begin{pmatrix} x_1^2 + x_2^2 & -x_1 - x_2 \\ -x_1 - x_2 & 2 \end{pmatrix}$$

Ridge (L2) Linear Regression

$$\mathbf{RSS}(\boldsymbol{\beta}) = (\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + \alpha \boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{\beta}$$

$$\frac{\partial \mathbf{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 2\boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + 2\alpha \boldsymbol{\beta} = \mathbf{0}$$

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \boldsymbol{\lambda}\mathbf{I}\right)^{-1} \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

sklearn.linear_model.Ridge

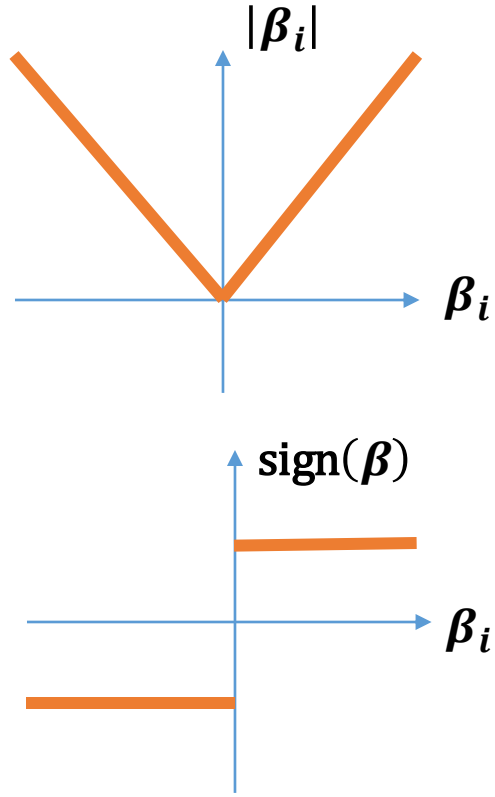
- This approach puts a square ($\boldsymbol{\lambda}\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{\beta}$) penalty on the coefficients $\boldsymbol{\beta}$. Large $\boldsymbol{\beta}$ produce a large penalty and are disfavored. This produces more stable and smoother solutions.

- Ridge regression is important if features are colinear (strongly correlated) or if there are not enough data to calculate robust solutions.

Lasso (L1) Linear Regression

$$\mathbf{RSS}(\boldsymbol{\beta}) = (\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + \alpha \sum_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{M}} |\boldsymbol{\beta}_{\boldsymbol{i}}|$$

$$\frac{\partial \mathbf{RSS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 2\boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + \alpha \cdot \mathbf{sign}(\boldsymbol{\beta}) = \mathbf{0}$$



[sklearn.linear_model.Lasso]

- This approach puts a linear ($\alpha \sum_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{M}} |\boldsymbol{\beta}_{\boldsymbol{i}}|$) penalty on the coefficients $\boldsymbol{\beta}$. Large $\boldsymbol{\beta}$ produce a large penalty and are disfavored, but less compared to Ridge. This produces more stable and smoother solutions, where many $\boldsymbol{\beta}_{\boldsymbol{i}}$ are 0 (variable selection)

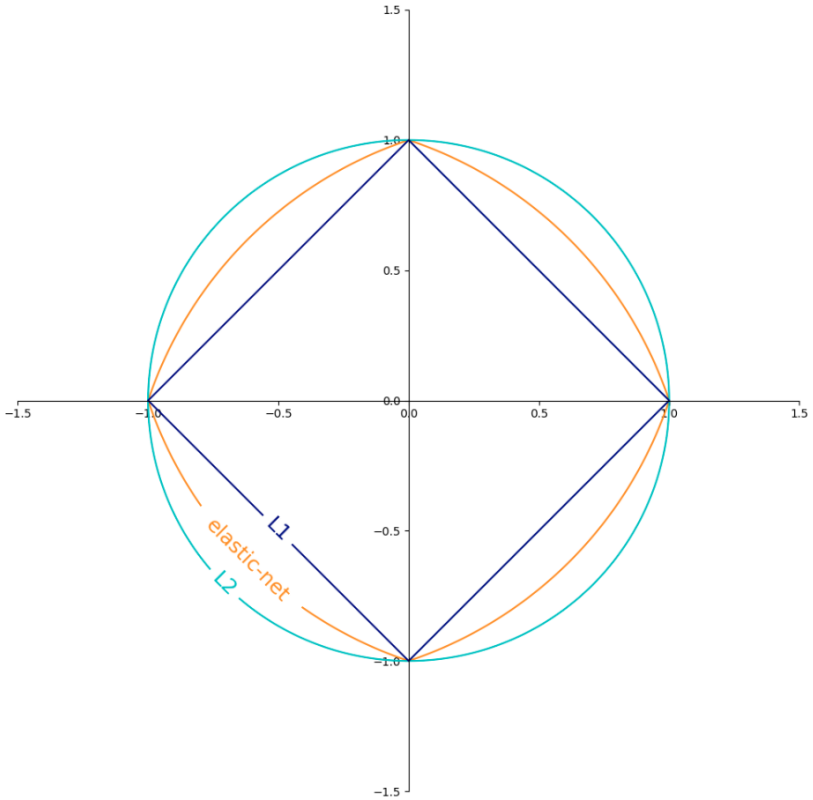
- Lasso regression is important if features are colinear (strongly correlated) or if there are not enough data to calculate robust solutions.

Elastic Net Linear Regression

$$\mathbf{RSS}(\boldsymbol{\beta}) = (\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y})^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + \frac{\alpha}{2}\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{\beta} + (1-\alpha)\sum_{\boldsymbol{i=0}}^{\boldsymbol{M}}|\boldsymbol{\beta_i}|$$

$$\frac{\boldsymbol{\partial}\mathbf{RSS}(\boldsymbol{\beta})}{\boldsymbol{\partial\beta}} = 2\boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{X} \cdot \boldsymbol{\beta} - \boldsymbol{y}) + \alpha\boldsymbol{\beta} + (1-\alpha)\mathbf{sign}(\boldsymbol{\beta})$$

Stochastic gradient descent (SGD)



[sklearn.linear_model.SGDRegressor]

- Elastic net is a hybrid of ridge and lasso regularization. It has been shown that Elastic net can outperform Lasso or Ridge especially if features are highly correlated.
- Lasso can produce solutions that are too sparse.
- On the other hand, Ridge regression can be dependent on outliers and produce many small values in high dimensions.

Linear regression

Swiss Institute of Bioinformatics

- **Linear Assumption.** Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).
- **Remove Noise.** Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.
- **Remove Collinearity.** Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.
- **Gaussian Distributions.** Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on you variables to make their distribution more Gaussian looking.
- **Rescale Inputs:** Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.