Basics of machine learning

Nicolas Duchateau, Rémi Emonet, Carole Lartizien

Deep Learning for Medical Imaging

Lyon (FR) - 15/04/2019

1. Introduction

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI

Program

~30min	**1. Introduction** Carole Lartizien
~75min	**2. Supervised learning** Rémi Emonet + Carole Lartizien
~75min	**3. Unsupervised learning** Nicolas Duchateau + Rémi Emonet
~30min	**4. Methods evaluation** Carole Lartizien + Rémi Emonet + Nicolas Duchateau
~30min	**5. Conclusion/To go further**

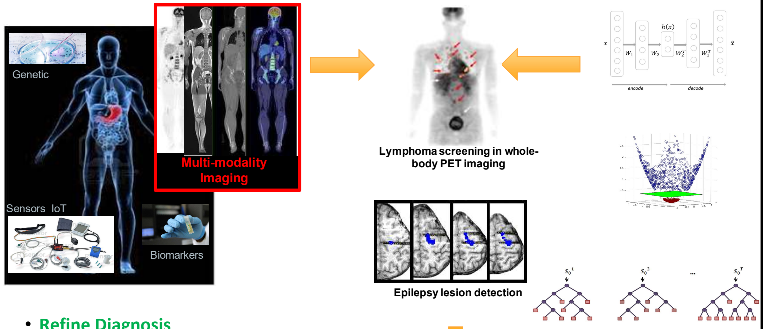
1. Introduction

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI

1. Introduction

Scientific and medical context

Precision medicine — Image-based pathology characterization — **Machine learning**

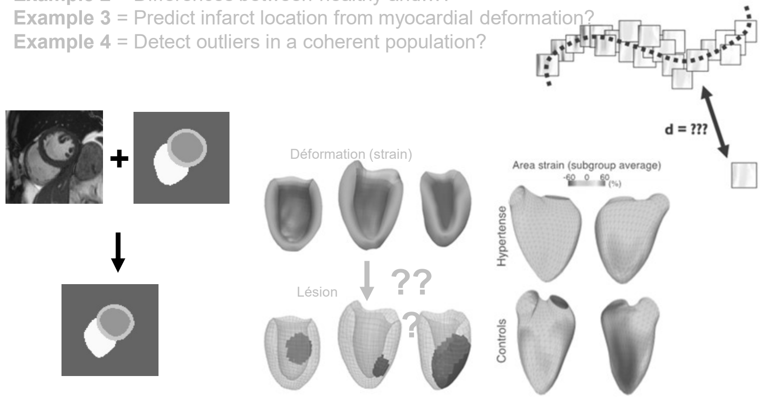


- **Refine Diagnosis**
- **Improve therapy planning**
- **Predict therapy outcome**
- **Develop preventive medicine.**

1. Introduction

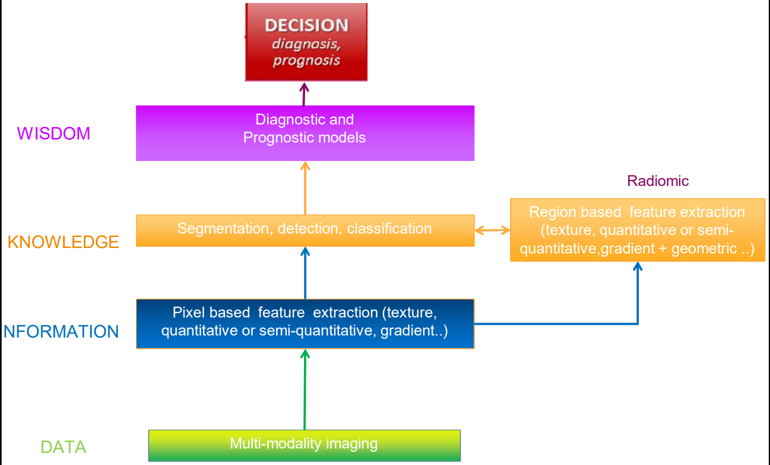
Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and…?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?



1. Introduction

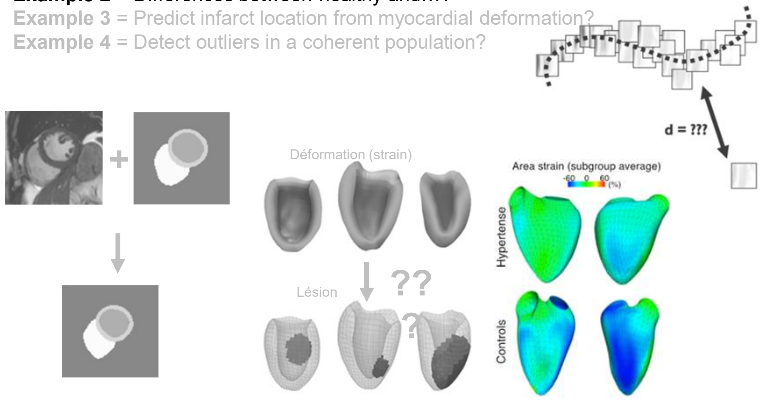
From imaging data to wisdom



1. Introduction

Some examples

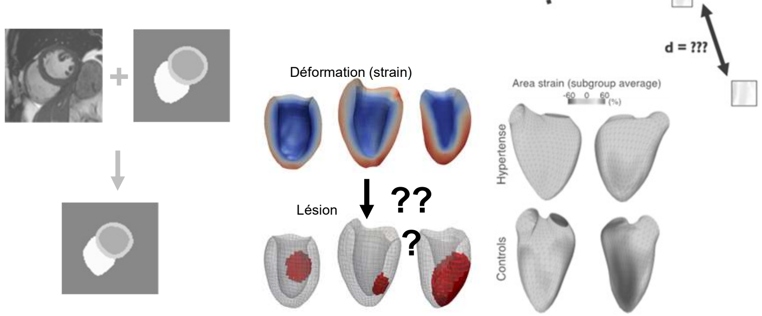
Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and…?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?



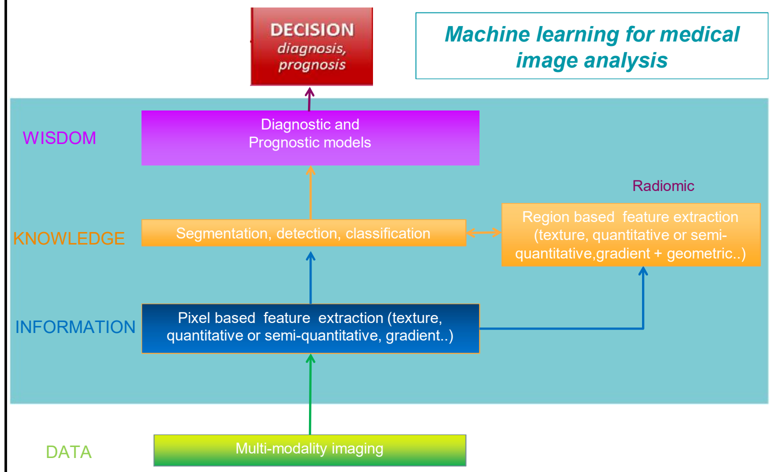
1. Introduction

Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and…?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?



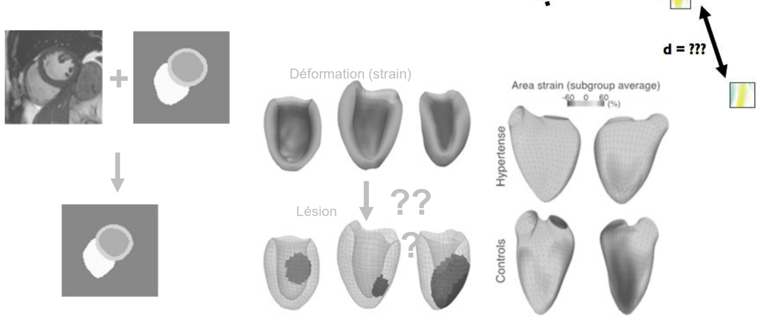
1. Introduction



1. Introduction

Some examples

Example 1 = Automatic segmentation?
Example 2 = Differences between healthy and…?
Example 3 = Predict infarct location from myocardial deformation?
Example 4 = Detect outliers in a coherent population?



1. Introduction

1. Scientific and medical context
2. **Basics of machine learning**
3. Some Historical highlights of AI

1. Introduction

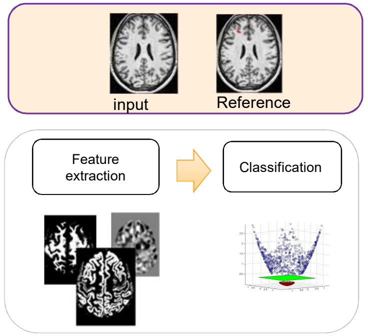
Basics of machine learning

1. Define a **task T**
2. Formulate this task as a decision model
3. Learn the decision model based on samples (**Data D**) and a **performance metric P**
4. Infer decision from this model on new samples

1. Introduction

Basics of machine learning …

2. Problem formulation within the statistical decision framework



→ **Binary supervised classification problem**

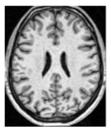
Design your model :
- Which features ?
- Which classifier model?

1. Introduction

Basics of machine learning

1. Task definition

- Detect lesions on brain T1 MRI



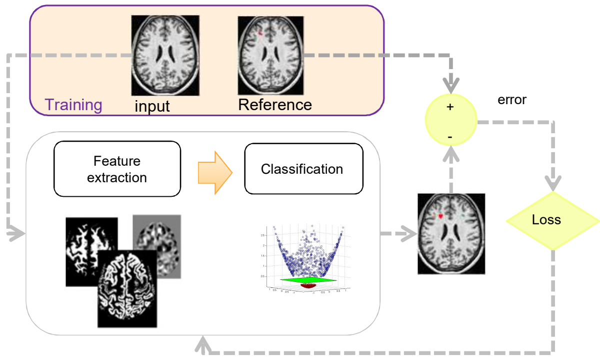
2. Problem formulation as a decision task

- Decide whether each voxel of the brain MR scan is a 'lesion' or 'normal tissue'
  - ➢ **Binary classification problem**
- Depending on the available samples, consider this problem as
  - ➢ **supervised,**
  - ➢ **unsupervised or**
  - ➢ **weakly supervised learning**

1. Introduction

Basics of machine learning …

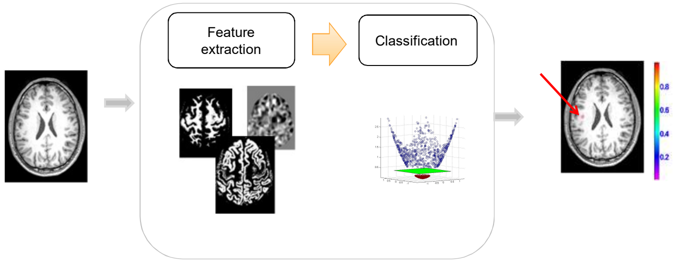
3. Learn the decision model based on training samples and performance metric



1. Introduction

Basics of machine learning …

4. Infer decision on new samples

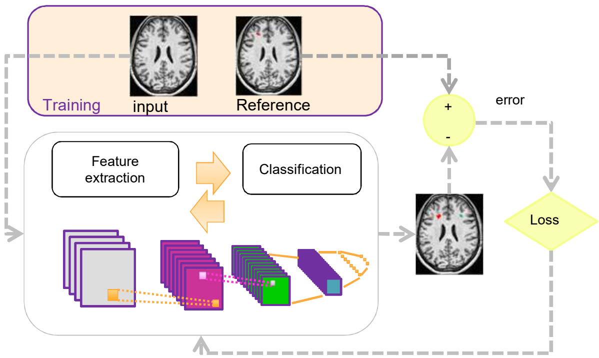


1. Introduction

1. Scientific and medical context
2. Basics of machine learning
3. Some Historical highlights of AI

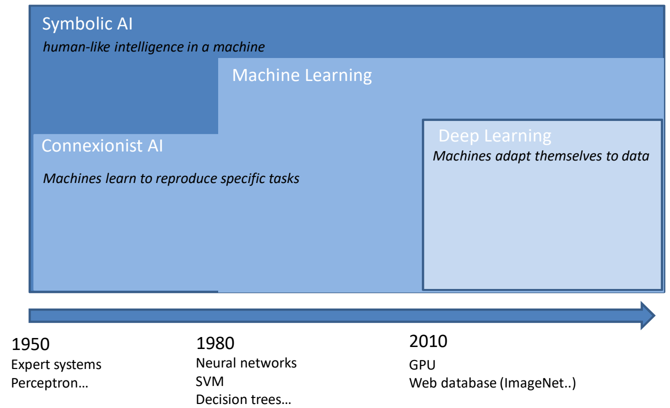
1. Introduction

…and deep learning



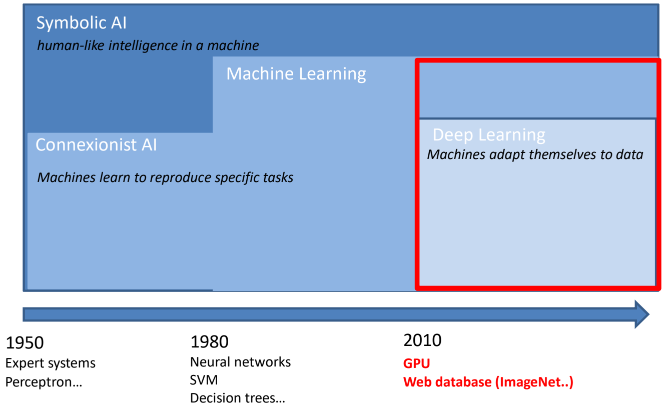
1. Introduction

Artificial Intelligence



1. Introduction

Artificial Intelligence



1. Introduction

Deep learning for medical image analysis

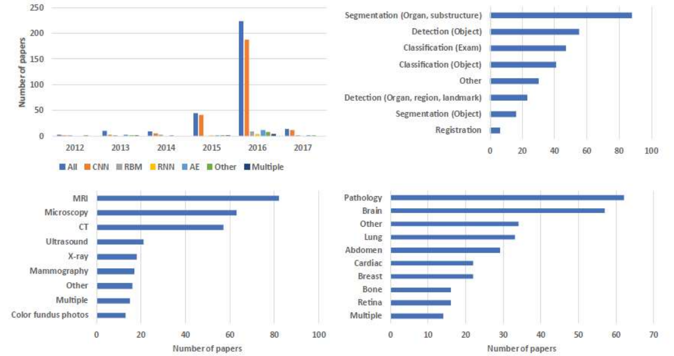
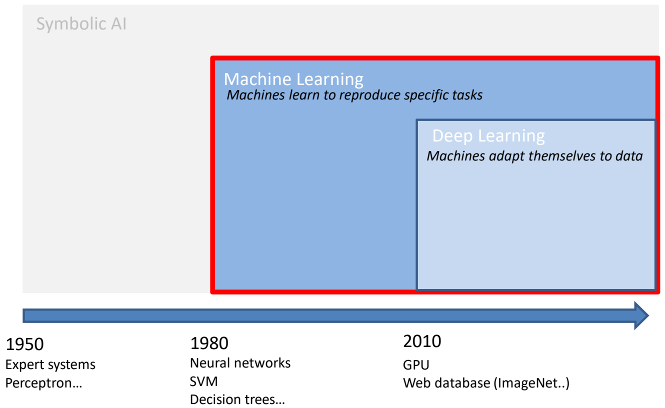


Figure 1: Breakdown of the papers included in this survey in year of publication, task addressed (Section 3), imaging modality, and application area (Section 4).

[Litjens, MEDIA 17]

1. Introduction

Artificial Intelligence



Program

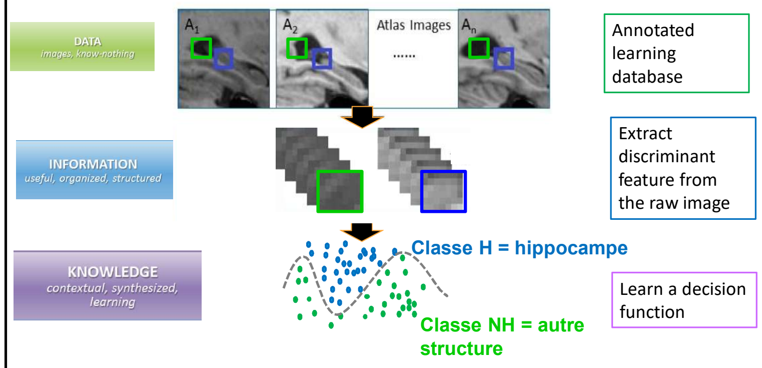
~30min	**1. Introduction** Carole Lartizien
~75min	**2. Supervised learning** Rémi Emonet + Carole Lartizien
~75min	**3. Unsupervised learning** Nicolas Duchateau + Rémi Emonet
~30min	**4. Methods evaluation** Carole Lartizien + Rémi Emonet + Nicolas Duchateau
~30min	**5. Conclusion / To go further**

2. Supervised learning

- i. Use case
- ii. Standard pipeline
- iii. Learning a decision function
- iv. Decision model based on the minimization of the misclassification error
- v. Decision trees
- vi. Neural networks

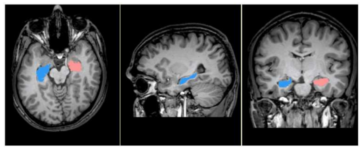
2. Supervised learning

Case study:



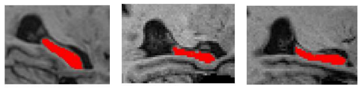
2. Supervised learning

Case study:



- **Problem definition**
  To automatically segment hippocampe in MRI T1 images

- **Material** : MRI brain images databases with hippocampes manually annotated by experts



- **Problem formulation as a decision task**

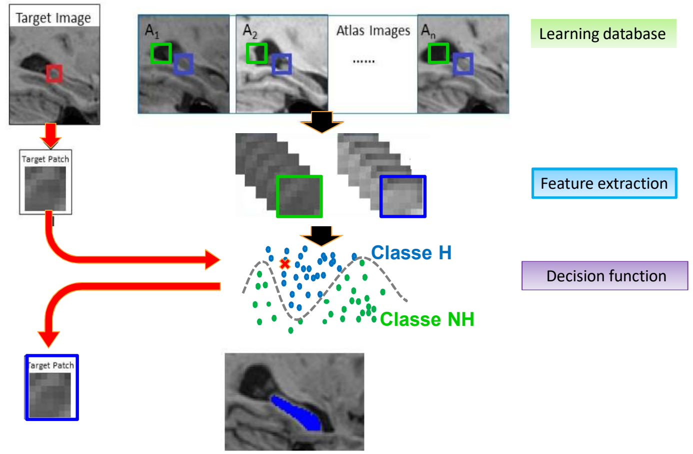
Decide whether each voxel belongs to the hippocampe structure or not

→ **Binary classification problem**

→ **Supervised classification problem**

2. Supervised learning

Case study:



2. Supervised learning

i. Use case

ii. Standard pipeline

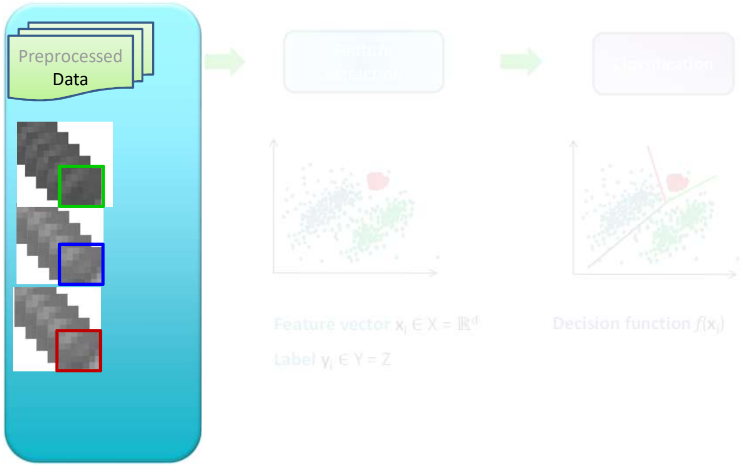
iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

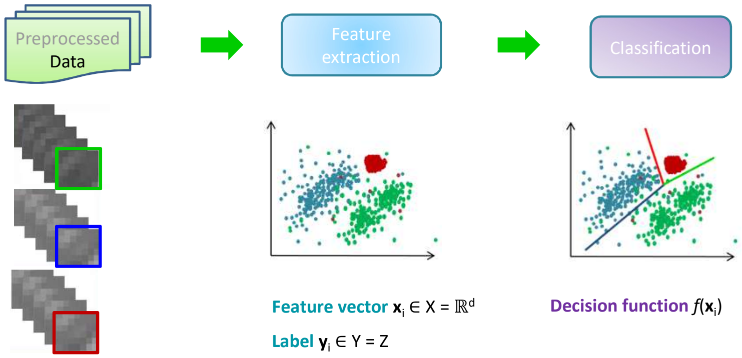
v. Decision trees

vi. Neural networks

2. Supervised learning



2. Supervised learning



Feature vector $\mathbf{x}_i \in X = \mathbb{R}^d$

Label $\mathbf{y}_i \in Y = Z$

Decision function $f(\mathbf{x}_i)$

2. Supervised learning

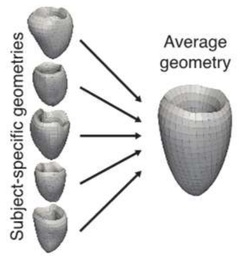
Data preprocessing

Data ready to be analyzed?

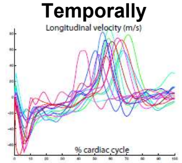
Different:
- **sampling**
- **geometries**
- **dynamics**

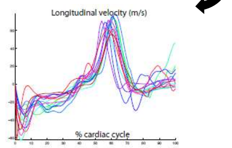
= normalize first !

Spatially



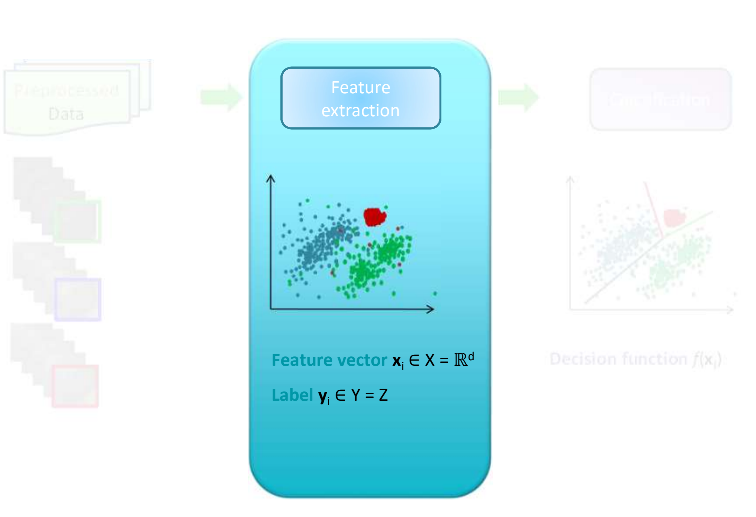
Temporally





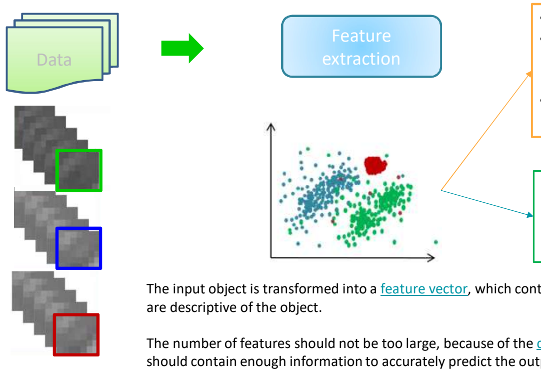
Duchateau et al. *Med Image Anal* 2011
De Craene et al. *ISBI* 2012

2. Supervised learning



2. Supervised learning

Feature extraction



- Raw Image intensity
- First and second order statistics, eg texture parameters
- Histograms

- Semi-quantitative and quantitative pharmacokinetic parameters

The input object is transformed into a feature vector, which contains a number of features that are descriptive of the object.

The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

➢ This topic will not be covered during this lecture!

2. Supervised learning

Feature extraction

Learning… but on what?

"known" descriptors / features… or to discover automatically ?

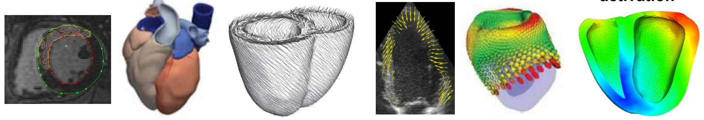
Images
- Gray level, texture, …

Shapes
- Geometry (meshes, curvature,…), fibers, …

Functional features
- Global: clinical measurements, outcome, …
- Local: **mechanical (motion / deformation)**, electrical, …

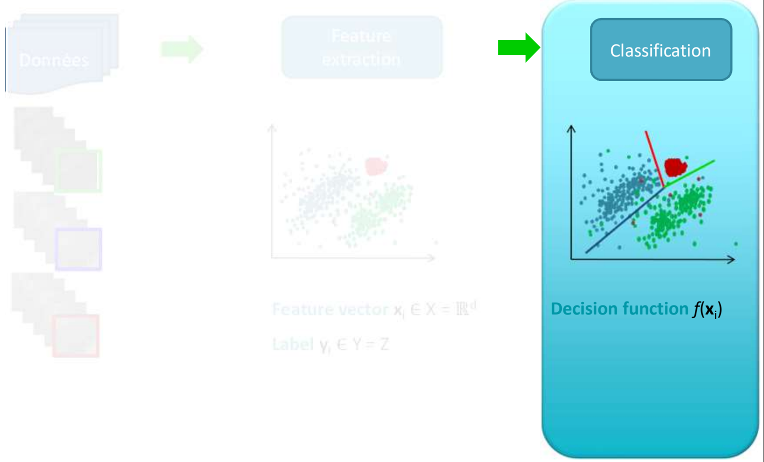
Specificities / constraints ?
- Physiology, specific structure (manifold)
- 4D (space + time) … or 5D (longitudinal data)
- High dimensionality
- Population size: from ~20 to 500+ subjects

Gray level | **Shape** | **Fibers** | **Velocities** | **Strain** | **Electrical activation**



2. Supervised learning

Learning a decision function in the feature space $X = \mathbb{R}^d$

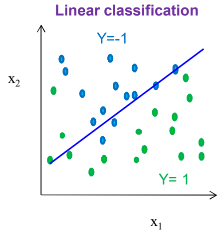


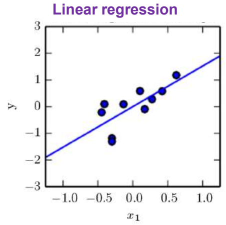
2. Supervised learning

Classification versus regression

$\mathbf{y} \in \mathbb{N}$, → **Classification**

$\mathbf{y} \in \mathbb{R}$, → **Regression**





In this lecture, we focus on **classification** models
To simplify, we consider a binary classification problem

2. Supervised learning

Objective

To **learn a function *f*** that maps an input $\mathbf{x}$ to an output y based on a series of annotated samples S= $\{(\mathbf{x}_1, y_1),\ldots(\mathbf{x}_n, y_n)\}$

- f is an element of some space of possible functions, usually called the *hypothesis space*.
- Usually, the class y is not directly outputted
  - f is either a **scoring function** eg a signed distance to the hyperplane,

$$f : \mathcal{X} \to \mathbb{R}$$

  - Or f is a **probability** of $\mathbf{x}$ belonging to class y

$$f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

- The output y is defined by a **decision rule** applied on the output of the scoring function

$$D(\mathbf{x}) = signe(f(\mathbf{x}))$$

2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks

2. Supervised learning

Supervised learning in a nutshell

- **Split the sample dataset** into three parts : a training, a validation and a test dataset
- **Choose a parameterized model function** with parameters Θ_1 and hyperparameters Θ_2 from an hypothesis space H
- **Fit the model parameters Θ_1** to the training dataset for a fixed value of Θ_2
  - **Choose an error function** that measures the misfit between the decision function D(f(xi)) and the class yi of all training data points (xi, yi)
  - **Minimize the error function**
- **Evaluate the performance** of your model on the validation dataset
- Retrain your model with another hyperparameter set Θ_2
- Select the best parameter set
- **Evaluate the performance of your best model** on the test dataset

2. Supervised learning

Supervised learning in a nutshell

- **Split the sample dataset** into three parts : a training, a validation and a test dataset
- **Choose a parameterized model function** with parameters Θ_1 and hyperparameters Θ_2 from an hypothesis space H
- **Fit the model parameters Θ_1** to the training dataset for a fixed value of Θ_2
  - **Choose an error function** that measures the misfit between the decision function D(f(xi)) and the class yi of all training data points (xi, yi)
  - **Minimize the error function**
- **Evaluate the performance** of your model on the validation dataset
- Retrain your model with another hyperparameter set Θ_2
- Select the best parameter set
- **Evaluate the performance of your best model** on the test dataset

2. Supervised learning

- **Différents types of error functions**
  - **Missclassification Error Risk** :
    - Bayesian classifier, SVM, logisitic regression, neural networks
  - **Other functionals**:
    - Fisher criterion for discriminant linear analysis (LDA)
    - Entropy for decision trees or neural networks
    - ....

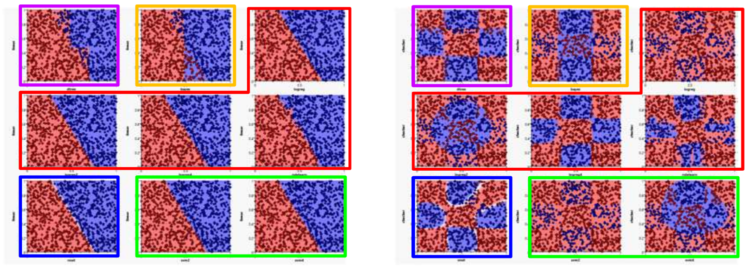
2. Supervised learning

How to choose and fit the decision function *f(x)*

There are different approaches to the classification problem

- **Two types of decision models**
  - **Linear models**: linear SVM, logistic regression logistique, Linear discriminant analysis
  - **Non linear models** : neural networks, kernel machine, decision trees
- **Different strategies to minimize the error function**
  - **Global minimization**: In the original feature space $\mathbb{R}^d$
  - **Recursive minimisation**: based on a recursive method applied in a one-dimensional space (eg decision trees)

2. Supervised learning



Linear classification task

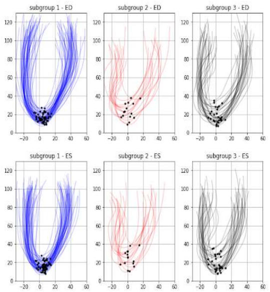
Non linear classification task

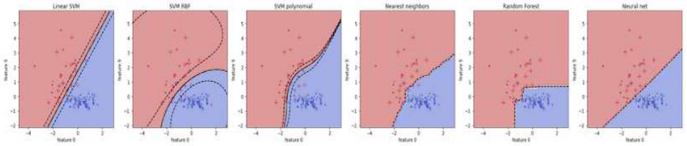
Decision tree Naïve Bayes Logistic regression Neural net SVM

2. Supervised learning

Hands-on session = Tuesday

Classification of **diseased vs. healthy** based on **cardiac shapes / function**





2. Supervised learning

Risk minimisation

Statistical learning theory is based on the notion of **risk** ***R*** also referred to as **prediction error**

Parameters of the decision function f for a given classification task are derived from the **minimization of the prediction error between** the estimated class labels $f(\mathbf{x}_i)$ and the true class labels y_i

$$R(f)=\mathbb{E}\left[L(Y,f(\mathbf{X}))\right]=\int_{\mathcal{X}\times\mathcal{Y}} L(y_i,f(\mathbf{x}_i))\mathbb{P}(\mathbf{x}_i,y_i)\,d\mathbf{x}_i dy_i$$

$L(.,.)$ is a **cost function** quantifying the cost of the prediction error

$\mathbb{P}(\mathbf{x}_i,y_i)$ is the **joint probability** of observing $\mathbf{x}_i$ and y_i

2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks

2. Supervised learning

Risk minimisation – Discriminative models

The decision function $f(\mathbf{x})$ is estimated directly, ie

- **Without modeling** and estimating the **posterior probability densities**
- By modeling directly the decision function and estimating the parameters of this function based on training samples.

2. Supervised learning

Empirical risk minimisation

- As seen above, the minimisation of risk R requires to estimate the joint probability distribution, which may not be trivial

$$\boxed{R(f) = \mathbb{E}\left[L(Y, f(\mathbf{X}))\right] = \int_{\mathcal{X}\times\mathcal{Y}} L(y_i, f(\mathbf{x}_i))\mathbb{P}(\mathbf{x}_i, y_i)\, d\mathbf{x}_i dy_i}$$

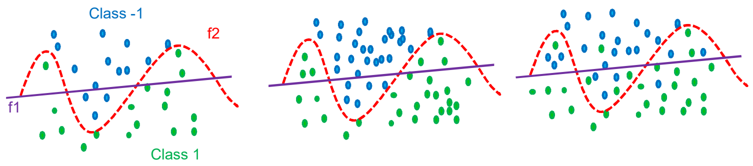
- An alternative is to minimize the empirical risk *Remp(f)* based on the learning data samples

$$\boxed{R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))}$$

$$\min_{f\in H} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$$

2. Supervised learning

Risk versus empirical risk minimisation



	complexity	Error
f1	-	(3,3)
f2	+	(0,0)

Error
(5,6)
(0,0)

Error
(4,4)
(6,4)

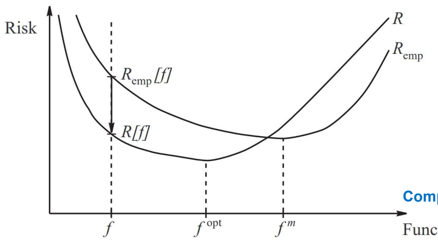
➢ Need to compromise the empirical prediction error and the complexity of the decision function

2. Supervised learning

Risk versus empirical risk minimisation

$R_{emp}(f) \rightarrow R(f)$ when $n \rightarrow \infty$ with n the number of training data samples

For a fixed n, $R_{emp}(f)$ et $R(f)$ *depend on the complexity / capacity of f* and converge to a minimun



$Min(R_{emp}(f)) \neq Min(R(f))$

[Courtesy of B. Scholkopf, NIPS 2001]

2. Supervised learning

Structural risk minimisation

- Minimisation of the empirical risk minimisation under constraint of good generalization performance

$$\min_{f\in H} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathrm{x}_i)) + \lambda\Omega(f)$$

Loss function **Regularisation function**

2. Supervised learning

Structural risk minimisation

- To solve this minimisation problem under constraints, we make some hypothesis :
  - On the model of the ***decision function f***
    - f is assumed to be a linear hyperplane in the feature space $\mathcal{X} = \mathbb{R}^d$

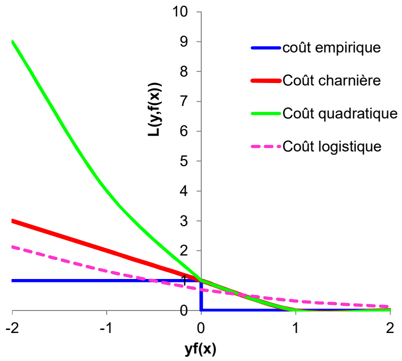
$$f : \mathcal{X} \to \mathbb{R}$$
$$\mathbf{x}_i \mapsto \mathbf{w}'\mathbf{x}_i + b$$

  - On the **loss function** L and the **regularisation function Ω**

2. Supervised learning

Exemple cost functions

- **Loss 0-1**
  $L(y_i, f(\mathbf{x}_i)) = (1 - \mathrm{sgn}(y_i f(\mathbf{x}_i))/2)$
- **Hinge Loss**
  $L(y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i))$
- **Quadratic loss**
  $L(y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i))^2$
- **Logistic loss**
  $L(y_i, f(\mathbf{x}_i)) = \ln(1 + \exp(-y_i f(\mathbf{x}_i)))$

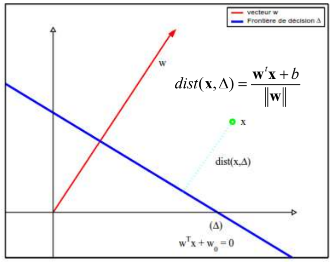


2. Supervised learning

Some mathematical reminders

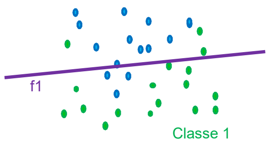
Training database $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$ of class $y_i \in \{-1,+1\}$

$$f(\mathbf{x}) = \mathbf{w}'\mathbf{x} + b$$



For samples $\mathbf{x}_i$ correctly classified:

$f_I(\mathbf{x}_i) * y_i > 0$



For samples $\mathbf{x}_i$ badly classified:

$f_I(\mathbf{x}_i) * y_i < 0$

2. Supervised learning

Exemple regularisation functions

When $f(\mathbf{x})$ is a linear decision function of the form

$f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$

avec **w** la normale à l'hyperplan séparateur et ***b*** un terme de biais

The regularisation function belongs to the familly of ℓ_p norms

$$\Omega_p(\mathbf{w}) = \|\mathbf{w}\|_p = \left(\sum_{i=1}^{d} |w_i|^p\right)^{\frac{1}{p}}$$

For p=1 (norme ℓ_1) and p=2 (norme ℓ_2)

2. Supervised learning

Structural risk minimisation

$$\min_{f\in H}\frac{1}{n}\sum_{i=1}^{n}L\left(y_i,f\left(\mathbf{x}_i\right)\right)+\lambda\Omega\left(f\right)$$

$$\Leftrightarrow$$

If $L(.,.)$ and $\Omega(.,.)$ are convex functions

$$\begin{cases}\min\limits_{f\in H}\frac{1}{n}\sum_{i=1}^{n}L\left(y_i,f\left(\mathbf{x}_i\right)\right)\\ s.c.\quad \Omega\left(f\right)\leq\tau\end{cases}$$

- No closed form solution for f in most of the cases
- Quadratic problem that can be solved with a standard optimization algorithm, eg conjugate gradients

2. Supervised learning

Exemple : Linear SVM

Problem formulation

$$\begin{cases}\max m\\ s.c.\ y_i\left(\frac{\mathbf{v}^t\mathbf{x}_i+a}{\|\mathbf{v}\|}\right)>m,\quad i=(1,2,..n)\end{cases}$$

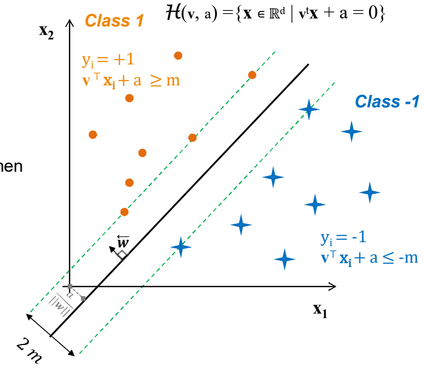
Ill-posed problem: if $(\mathbf{v}, a)$ is a solution, then

$(k^*\mathbf{v}, k^*a)$), $\forall$ 0 < k is a solution too

→ We define:

$$\mathbf{w}=\frac{\mathbf{v}}{m\|\mathbf{v}\|}\ soit\ \|\mathbf{w}\|=\frac{1}{m}$$

$$b=\frac{a}{m\|\mathbf{v}\|}$$



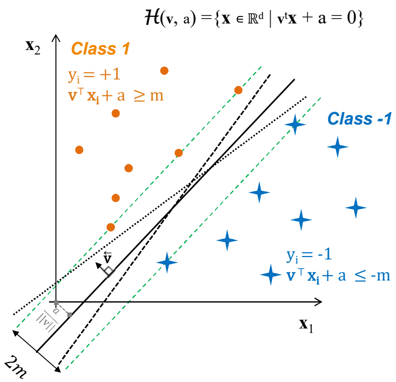
2. Supervised learning

Exemple : Linear SVM

- Find a linear hyperplane

$$f\left(\mathbf{x}_i\right)=\mathbf{v}^T\mathbf{x}_i+a$$

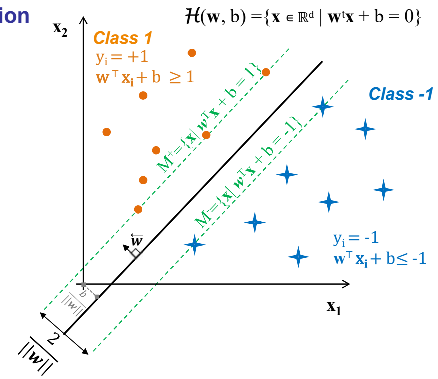
- that **maximises the margin** m between the training samples of both classes
- The margin m is the **smallest distance of any training sample to the decision hyperplane**



2. Supervised learning

Canonical problem formulation

$$\begin{cases}\min\frac{1}{2}\|\mathbf{w}\|^2\\ s.c.\ y_i\left(\mathbf{w}^t\mathbf{x}_i+b\right)>1,\quad i=(1,2,..n)\end{cases}$$



2. Supervised learning

Exemple : Linear SVM

When the data are almost linearly separable :

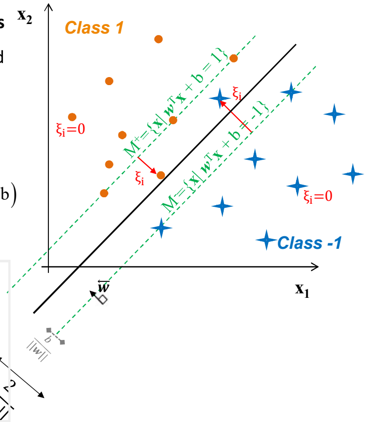
Errors are modeled as **positive slack variables** ξ_i associated to each sample $(\mathbf{x}_i, y_i)$ and measuring the distance to the margin

No error: $y_i(\mathbf{w}^t\mathbf{x}_i + b) \geq 1 \Rightarrow \zeta_i = 0$

Error : $y_i(\mathbf{w}^t\mathbf{x}_i + b) < 1 \Rightarrow \zeta_i = 1 - y_i(\mathbf{w}^t\mathbf{x}_i + b)$

→ Hinge loss $\zeta_i = \max(0, 1 - y_i f(x_i))$

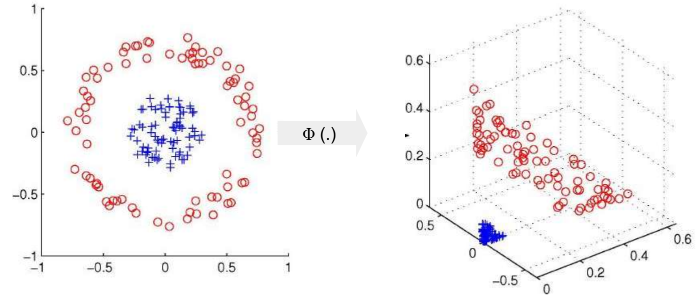
$$\begin{cases} \min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \zeta_i \\ s.c.\ y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1 - \zeta_i, \quad i=(1,2,..n) \\ \zeta_i > 0 \end{cases}$$



2. Supervised learning

Generalisation to nonlinear problem

Find a **mapping function** Φ that maps the data form the original representation space $\mathcal{X}$ into a redescription space $\mathcal{H}$ of higher dimension where the classification problem is linear ie the decision function may be written as $h(x) = w^t\varphi(x) + b$



2. Supervised learning

Exemple : Linear SVM

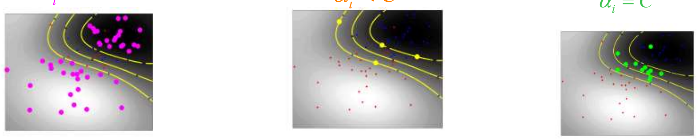
Equation of the separating hyperplane is as follows :

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i{}^t \mathbf{x} + b$$

Where αi are the Lagrange coefficients

Lagrange coefficients, support vectors and cost variable C

$\alpha_i = 0$ $\alpha_i < C$ $\alpha_i = C$



données inutiles bien classées — données importantes support — données suspectes

[Source :wikistat : Machine à vecteurs supports]

2. Supervised learning

Non Linear SVM

$$\begin{cases} \max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle_H + \sum_{i=1}^{n}\alpha_i \\ avec \quad 0 \leq \alpha_i \leq C, \qquad i = 1,\ldots,n \\ et \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle_{\mathcal{H}} + b$$

Equation of the decision function is a weighted sum of scalar products between pairs of vectors of the redescription space

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x})\rangle_{\mathcal{H}}$$

➡ **This enables to use the kernel trick**

2. Supervised learning

SVM and kernel trick

Instead of defining a non-linear projection Φ , we use a kernel function associated to the projection function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

Une kernel function k is a **similarity function**. It has to satisfy some properties referred to as the Mercer conditions to guarantee the existence of the corresponding function Φ

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

Symetry

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$$

Positive definite

$$\sum_{i,j} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) > 0, \; \forall \mathbf{x}_i \in \mathcal{X}, \forall \alpha_i \in \mathbb{R}$$

2. Supervised learning

SVM and kernel trick

Advantages of the kernel function :

- The computation of the the kernel function is performed in the native representation space $\mathcal{X}$, which is less computationally intensive than performing a scalar product in a high-dimensional space
- Projection Φ does not need to be explicitely formulated. It is thus possible to consider complex project in potentially infinite redescription space
- The **kernel function is constructed** based on the Mercer conditions **without formulating the corresponding projection function Φ**

2. Supervised learning

Non linear SVM with kernel formulation

The dual problem is reformulated as

$$\begin{cases} \max_{\alpha} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) + \sum_{i=1}^{n}\alpha_i \\ avec \quad 0 \le \alpha_i \le C, \qquad i = 1, \ldots n \\ et \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \end{cases}$$

With solution:

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b}$$

2. Supervised learning

Some standard kernels

- Linear kernel : trivial case equivalent to linear classifier.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^t \mathbf{x}_j$$

- Polynomial kernel

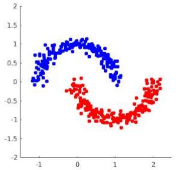
$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^t \mathbf{x}_j + 1)^d$$

- Gaussian kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

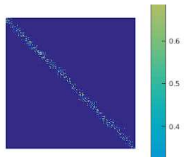
68

2. Supervised learning

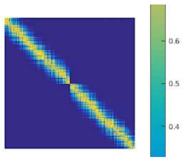


$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

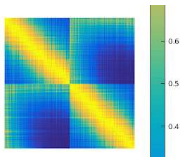
σ = 0.05



σ = 0.25



σ = 1



2. Supervised learning

Risk minimisation - Generative models

- **Make some hypothesis on the distribution of the conditional probabilities** $\mathbb{P}(\mathrm{X} = \mathbf{x}_i | \mathrm{Y} = k)$ and priors $\mathbb{P}\left(\mathrm{Y} = k\right)$
- **Learn the conditional probabilities** on the training database
- **Estimate the posterior probabilities** based on the Bayes Theorem

$$\mathbb{P}\left(\mathrm{Y}=1 | \mathrm{X}=\mathbf{x}_i\right) = \frac{\mathbb{P}\left(\mathrm{X}=\mathbf{x}_i | \mathrm{Y}=1\right)\mathbb{P}\left(\mathrm{Y}=1\right)}{\mathbb{P}\left(\mathrm{X}=\mathbf{x}_i | \mathrm{Y}=1\right)\mathbb{P}\left(\mathrm{Y}=1\right)+\mathbb{P}\left(\mathrm{X}=\mathbf{x}_i | \mathrm{Y}=-1\right)\mathbb{P}\left(\mathrm{Y}=-1\right)}$$

2. Supervised learning

Exemple Nonlinear discriminant analysis using kernel function operator

- Kernel Fisher discriminant analysis
- Kernel logistic regression
- ....

2. Supervised learning

Bayesian classifier

The Bayes classifier minimizes the risk of classifying sample $\mathbf{x}_i$ in class k as

$$\underset{k=\{0,\ldots,L\}}{\arg\min}\, \mathrm{R}\left(k, \mathbf{x}_i\right)$$

$$R\left(k, \mathbf{x}_i\right) = \sum_{j=1}^{L} L\left(k, j\right)\mathbb{P}(Y = j | X = \mathbf{x}_i)$$

$L\left(k, j\right)$ The cost of assigning a class j to any sample belonging to class k

$\mathbb{P}(Y = j | X = \mathbf{x}_i)$ The posterior probability of assigning class j to sample $\mathbf{x}_i$

Le classifieur bayesien

Dans le cas d'un problème de classification binaire, , l'expression du risque pour chaque classe est $y_i \in \{-1,1\}$

$$R(1,\mathbf{x}_i) = L(1,1)\mathbb{P}(Y=1|X=\mathbf{x}_i) + L(1,-1)\mathbb{P}(Y=-1|X=\mathbf{x}_i)$$

$$R(-1,\mathbf{x}_i) = L(-1,1)\mathbb{P}(Y=1|X=\mathbf{x}_i) + L(-1,-1)\mathbb{P}(Y=-1|X=\mathbf{x}_i)$$

L'étiquette de $\mathbf{x}_i$ sera $y_{i=1}$ ssi $R(1,\mathbf{x}_i) < R(-1,\mathbf{x}_i)$
soit

$$\big(L(1,1)-L(-1,1)\big)\mathbb{P}(Y=1|X=\mathbf{x}_i) < \big(L(1,-1)-L(-1,-1)\big)\mathbb{P}(Y=-1|X=\mathbf{x}_i)$$

En supposant que $L(1,1)-L(-1,1)<0$ on obtient

Rapport des probabilités postérieures ⇨ $$\boxed{\frac{\mathbb{P}(Y=1|X=\mathbf{x}_i)}{\mathbb{P}(Y=-1|X=\mathbf{x}_i)}} > \frac{L(1,-1)-L(-1,-1)}{L(1,1)-L(-1,1)}$$ ⇦ constante

2. Supervised learning

Bayesian classifier

For a binary classification problem, the decision function is

$$\boxed{f(\mathbf{x}_i) = \ln\left(\frac{\mathbb{P}(X=\mathbf{x}_i|Y=1)}{\mathbb{P}(X=\mathbf{x}_i|Y=-1)}\right)}$$

Likelihood ratio

The corresponding decision rule is

$$\boxed{D(\mathbf{x}_i) = \begin{cases} 1 & si \quad f(\mathbf{x}_i) \geq k \\ -1 & \text{sinon} \end{cases}}$$

Le classifieur bayesien

D'après le théorème de Bayes

$$\mathbb{P}(Y=1|X=\mathbf{x}_i) = \frac{\mathbb{P}(X=\mathbf{x}_i|Y=1)\mathbb{P}(Y=1)}{\mathbb{P}(X=\mathbf{x}_i|Y=1)\mathbb{P}(Y=1)+\mathbb{P}(X=\mathbf{x}_i|Y=-1)\mathbb{P}(Y=-1)}$$

L'étiquette de x_i sera donc l_l ssi

$$\mathbb{P}(Y=1|X=\mathbf{x}_i) > \frac{L(1,-1)-L(-1,-1)}{L(1,1)-L(-1,1)}\mathbb{P}(Y=-1|X=\mathbf{x}_i)$$

$$\Leftrightarrow \mathbb{P}(X=\mathbf{x}_i|Y=1)\mathbb{P}(Y=1) > \frac{L(1,-1)-L(-1,-1)}{L(1,1)-L(-1,1)}\mathbb{P}(X=\mathbf{x}_i|Y=-1)\mathbb{P}(Y=-1)$$

$$\Leftrightarrow \frac{\mathbb{P}(X=\mathbf{x}_i|Y=1)}{\mathbb{P}(X=\mathbf{x}_i|Y=-1)} > \frac{L(1,-1)-L(-1,-1)}{L(1,1)-L(-1,1)}\frac{\mathbb{P}(Y=-1)}{\mathbb{P}(Y=1)}$$ ⇦ constante

On pose

$$\boxed{f(\mathbf{x}_i) = \ln\left(\frac{\mathbb{P}(X=\mathbf{x}_i|Y=1)}{\mathbb{P}(X=\mathbf{x}_i|Y=-1)}\right)}$$ ⇦ **Rapport de vraisemblance**

2. Supervised learning

Naive Bayes classifier

Hypothesis of conditional independence between every pair of features given the value of the class variable.

$$\mathbf{x}_i^j \perp \mathbf{x}_i^k, \quad \forall (j,k) \in \{1,\dots,d\}, \forall i \in \{1,\dots,n\}$$

$$\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d$$

$$\mathbf{x}_i = (x_i^1, x_i^2, \dots x_i^d)$$

The Bayes theorem gives

$$\mathbb{P}(Y=k|X=\mathbf{x}_i) = \frac{\mathbb{P}(X=\mathbf{x}_i|Y=k)\mathbb{P}(Y=k)}{\sum_{j=1}^{K}\mathbb{P}(X=\mathbf{x}_i|Y=j)\mathbb{P}(Y=j)}$$

Using the naive conditional independence assumption

$$\mathbb{P}(Y=k|X=\mathbf{x}_i) = \frac{\mathbb{P}(Y=k)\prod_{k=1}^{p}\mathbb{P}(X^k=\mathbf{x}_i^k|Y=l)}{\sum_{j=1}^{K}\mathbb{P}(X=\mathbf{x}_i|Y=j)\mathbb{P}(Y=j)}$$

The denominator is constant given the input

$$\mathbb{P}(Y=k|X=\mathbf{x}_i) \propto \mathbb{P}(Y=k)\prod_{k=1}^{p}\mathbb{P}(X^k=\mathbf{x}_i^k|Y=l)$$

2. Supervised learning

Naive Bayes classifier

we can use the following classification rule:

$$\boxed{y_i = \arg\max_l \mathbb{P}(\mathrm{Y}=l)\prod_{k=1}^{d}\mathbb{P}\left(\mathrm{X}^k = \mathrm{x}_i^k \middle| \mathrm{Y}=l\right)}$$

$$\mathbf{x}_i = \left(\mathrm{x}_i^1, \mathrm{x}_i^2, \ldots \mathrm{x}_i^d\right)$$

- The independent conditional probabilities are estimated separately for each feature
- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of the conditional probabilities

2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks

2. Supervised learning

Models based on risk minimisation: Advantages and Limitations

→ **Advantages**

- **Quiet flexible inputs**: based on kernel computation
- **Interpretable** (somewhat): people are able to understand decision tree models
- **Produce an exact solution**
- Kernel trick to **efficiently compute non-linear models**
- **Quite robust to small size and unbalanced training datasets**

→ **Limitations**

- **Low interpretability** : difficult to extract the most discriminant features
- **Problem with large scaled datasets**

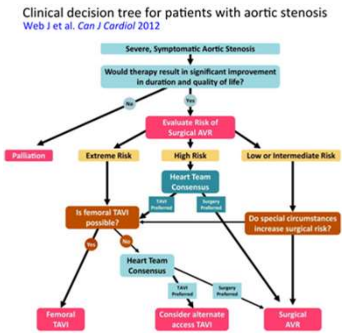
2. Supervised learning

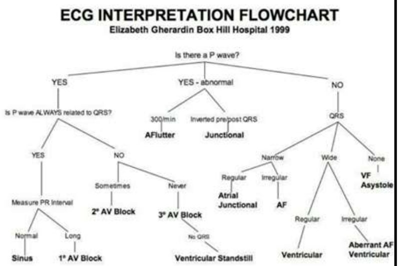
Decision Trees

2. Supervised learning

Decision Trees: Motivation

Existing decision trees based on **clinical evidence and expertise**, use **elementary features** and manual **thresholds.**





2. Supervised learning

Decision Tree in Computer Science

→ A ***model*** describing a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that
- to any input value/point $x \in \mathcal{X}$
- associates $f(x) \in \mathcal{Y}$

→ **A** Tree-shaped **representation**
- a root node, other split nodes, and leaves
- each split n has a *test function* $d^n(x)$ that gives a child index usually ***using a <u>single coordinate</u>*** (e.g., $x_8 \leq 42$)
- each *leaf* l has a prediction model usually **very simple** (e.g. constant value) $x \mapsto f^l(x)$

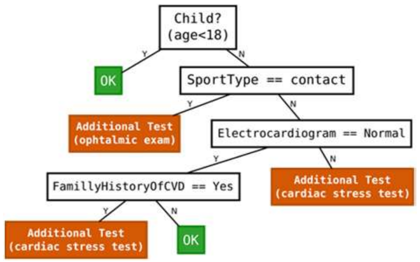
→ A simple **way of computing** the output (for a point x)
- start at the root
- if in a *split* node n, **move** to child $d^n(x)$
- if in a *leaf* node l, **return** $f^l(x)$

2. Supervised learning

What is a Decision Tree (DT)?

→ Big idea of DT
- Use very **elementary decision rules**
- **Combine them** into a tree

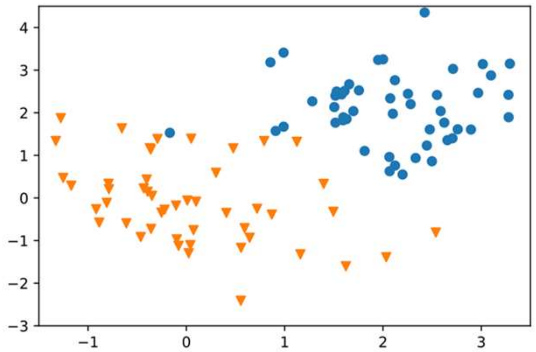
→ Example: binary DT for sport practice




```
if age<18:
  return "OK"
else:
  if sport_type == contact:
  | return "Additional Test (ophthalmic exam)"
  else:
  | if electrocardiogram == normal:
  | | if family_history_of_CVD:
  | | | return "Additional Test (cardiac stress test)"
  | | else:
  | | | return "OK"
  | else:
  | | return "Additional Test (cardiac stress test)"
```

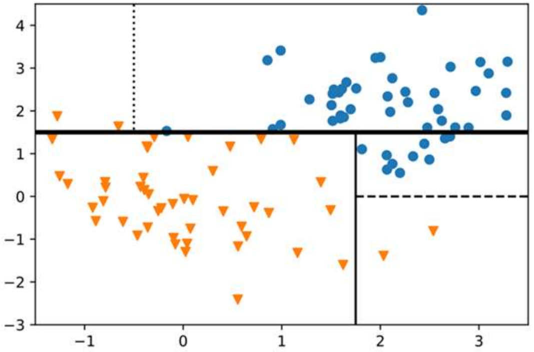
2. Supervised learning

Decision Tree in Computer Science



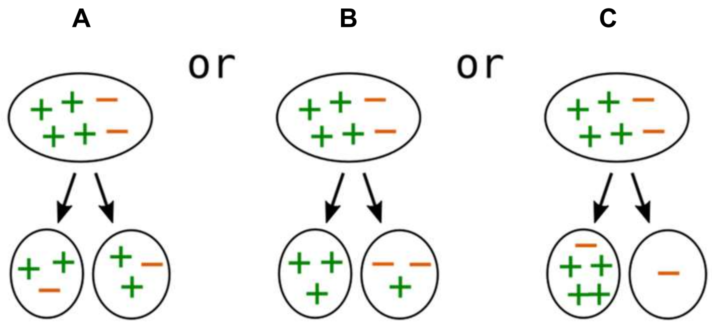
2. Supervised learning

Decision Tree in Computer Science



2. Supervised learning

Good versus Bad Splits for Decision Trees? which is a better split?



2. Supervised learning

Learning Optimal Decision Trees

- ➔ **Very complex** (NP-complete), even for simple definitions of "optimal"
- ➔ Use of **heuristics** and of a greedy Top-Down approach

- ➔ Principle of TDIDT **(Top-Down Induction/learning of DT)**
  - ◆ Start with an *empty tree and all examples* (dataset)
  - ◆ Find a *good test*
    - good test?
    - examples with same class fall on the same side
    - or, similar examples fall on the same side
    - for each possible test outcome, create child node
  - ◆ *Move each example* to a child, according to the test outcome
  - ◆ *Repeat* for each child that is not "pure"

- ➔ **Main question**
  - ◆ how to decide **which test/split is "best"**

2. Supervised learning

Toward finding the best test/split (for building classification trees)

- ➔ Find test for which children are **as "pure" as possible**

- ➔ **Entropy as purity** (borrowed from the information theory)
  - ◆ Entropy is a measure of "missing information"
  - ◆ More precisely, the number of bits needed to represent the missing information,
  - ◆ … on average, using the optimal encoding

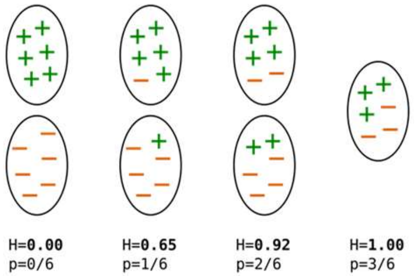
- ➔ **Entropy definition**
  - ◆ given a set S
  - ◆ with instances belonging to class C with prob p_c
  - ◆ we have:

$$Entropy(S) = -\sum_c p_c \, log_2(p_c)$$

2. Supervised learning

Entropy

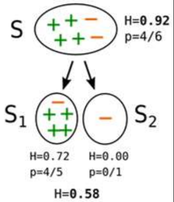
- Considering a node n, with a part of the dataset
- Denoting *p* = proportion of instances of class +1 in set n
- Note that if p=0, p×log2(p) is undefined but tend to 0
- The maximum entropy of 1
  is reached when the 2 classes are perfectly mixed



$$Entropy(S) = -\sum_c p_c \, log_2(p_c)$$

2. Supervised learning

Information gain



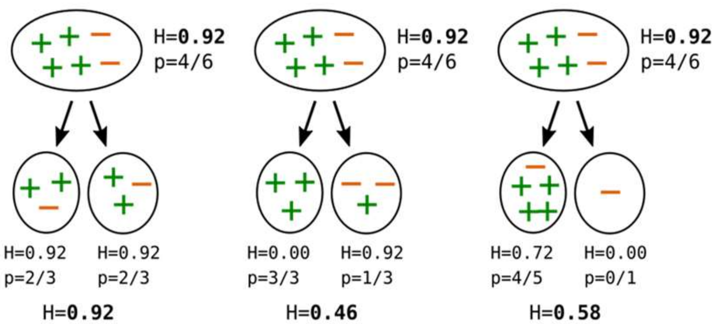
- **Heuristic for choosing a test in a node**
  - (on average over the children)
  - on average, provides most information about the class
  - on average, reduces the class entropy the most
  - expected reduction of entropy = information gain
- **Information gain**

$$Gain(S, A) = Entropy(S) - \sum_v \frac{|S_v|}{|S|} Entropy(S_v)$$

  - S = set of instances in a given node n
  - Sv = set of instances of S that go in child v of n
  - |Sv| / |S| = proportion of instances in Sv

2. Supervised learning

Good versus Bad Splits for Decision Trees?



2. Supervised learning

Other purity measure/gain (alternative to entropy)

- **Gini impurity index**
  - (not to be confused with gini coefficient)
  - "measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset" (*lower is better*).

$$Gini(S) = \sum_c p_c(1 - p_c) = \sum_c p_c - \sum_c p_c^2 = 1 - \sum_c p_c^2$$

- (for binary classification) $Gini(S) = 2p(1-p)$

$$Gain(S, A) = Gini(S) - \sum_v \frac{|S_v|}{|S|} Gini(S_v)$$

2. Supervised learning

Nature of Decision Tree Inputs $x \in \mathcal{X}$

- ➔ $\mathcal{X}$ can have any number of coordinates with arbitrary types
- ➔ *numbers*, e.g., $x_i \in \mathbb{N}$ or $x_i \in \mathbb{R}$
- ➔ *categorical variable*, e.g.,

  $x_i \in \{TRUE, FALSE\}$
  $x_i \in \{-1, 1\}$
  $x_i \in \{rainy, snowy, sunny\}$
  $x_i \in \{monday, tuesday, \cdots, sunday\}$
  $x_i \in \{yes, no\}$

  and others (templates, binary image patches, …)

- ➔ Test functions $d^n(x)$ can take various forms
  - ◆ two children: equal or not? $d^n : x \to (x_i = cst)$
  - ◆ two children: greater than? $d^n : x \to (x_i \geq cst)$
  - ◆ one child per possible outcome: $d^n : x \to x_i$
    e.g., 3 children with indices $snowy \;\; sunny \;\; rainy$

2. Supervised learning

DT: Advantages and Limitations

- ➔ **Advantages**
  - ◆ **Flexible input**: numerical and categorical data, no need for normalization, no assumptions, etc
  - ◆ **Interpretable** (somewhat): people are able to understand decision tree models
  - ◆ White-box: easy to know why a decision is taken
  - ◆ Performs well on large datasets
  - ◆ Universal approximator
  - ◆ Automatic **feature selection**
- ➔ **Limitations**
  - ◆ **Lack of robustness**: small change in the training data ⇒ possible large change in the tree
  - ◆ NP-complete problem requires **heuristics** and greedy algorithms
  - ◆ Need to take care of "imbalanced" categorical features
  - ◆ Need to take care of the **overfitting**

2. Supervised learning

Nature of Decision Tree Outputs $y \in \mathcal{Y}$

- ➔ *Categorical output*, e.g.,
  - ◆ **Binary classification,** $y \in \{-1, 1\}$
  - ◆ **Multiclass classification,** $y \in \{dog, cat, car, truck, \cdots\}$
  - ◆ **Rating: 1 to 5 stars**

⇒ Leaf $f^l(x) = cst_l$ (one of the outcomes)

- ➔ *Numerical output*, e.g.,
  - ◆ **Regression,** $y \in \mathbb{R}$
  - ◆ **Multi-dimensional regression, e.g.** $y \in \mathbb{R}^D$
  - ◆ **Count-regression**

⇒ Leaf $f^l(x) = cst_l$ or $f^l(x) = w_l^T x + b$ (affine, …)

- ➔ **Classification** and **regression** trees, but also **clustering** trees…
- ➔ NB: we focused on classification trees

2. Supervised learning

Avoiding overfitting with DT

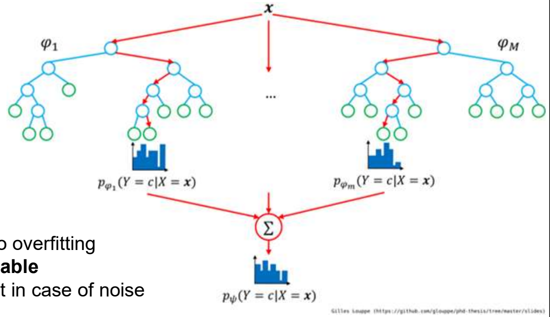
- ➔ **Option 1**
  - ◆ stop adding nodes when overfitting starts occurring
  - ◆ needs a stopping criterion:
  - ◆ predefined (max-depth, min-leaf-size)
  - ◆ using a validation set
  - ◆ using statistical tests or MDL (minimum description length)
- ➔ **Option 2**
  - ◆ don't bother about overfitting when growing the tree
  - ◆ after the tree has been built, start pruning it
  - ◆ **prune** to get better trees (validation)

2. Supervised learning

Random Forests (RF), in a few words

- ➔ Ensemble learning, bagging: **multiple trees** (M trees)
  - ◆ **Random dataset** (bootstrap: same size, drawn with replacement)
  - ◆ **Random features** (m features) at each split
  - ◆ Fully grown trees (no pruning)

- ➔ Prediction using a **majority vote**, or **average**
- ➔ NB: bagging allow to estimate the error of each tree



- ➔ **Fast and robust** to overfitting
- ➔ But, **non-interpretable**
  may still overfit in case of noise

2. Supervised learning

i. Use case

ii. Standard pipeline

iii. Learning a decision function

iv. Decision model based on the minimization of the misclassification error

v. Decision trees

vi. Neural networks

2. Supervised learning

Example of Decision Forests

Structured Decision Forests for Multi-modal Ultrasound Image Registration
Ozan Oktay et al., MICCAI 2015

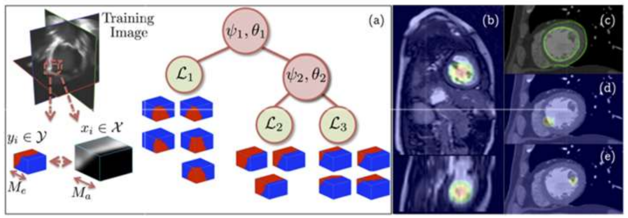
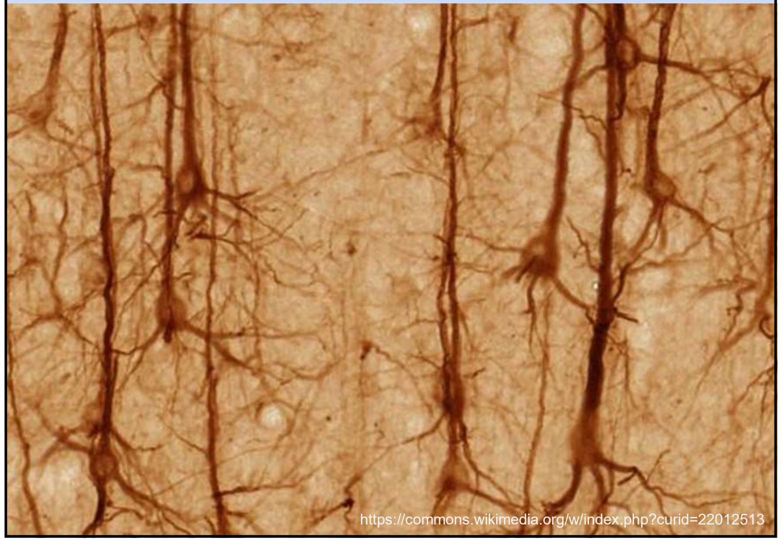


Fig. 2. Structured decision tree training procedure, label patches are clustered at each node split (a). Mid-ventricle (b), mid-septal (d) and mid-lateral (e) wall landmark localization by using PEMs (in green)(c) and regression nodes.

2. Supervised learning

Neural Networks

2. Supervised learning



2. Supervised learning

Artificial Neural Networks: some history

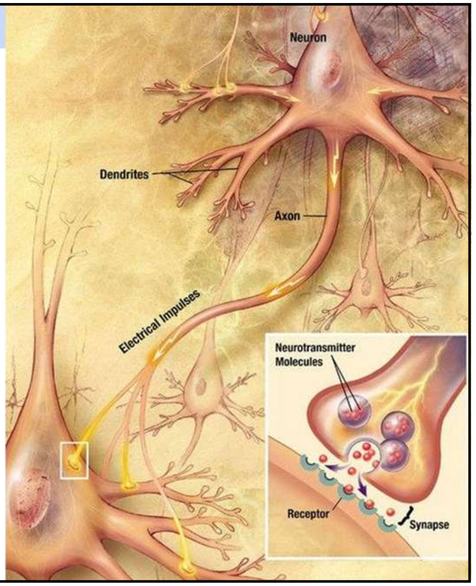
- → **Started in the 50s**
- → **Became more popular in the 80s**
  (“backpropagation” in 1975)

  Rumelhart, Hinton, and McClelland (1986)
  A General Framework for Parallel Distributed Processing: explorations in the microstructure of cognition
- → **Big slow down in the 90s**
- → **2010s**
  - ◆ More data, more processing power (GPU)
  - ◆ Advances in optimization, architecture (convolution, ReLU, skip conn.)
  - ◆ “Deep Neural Networks”
  - ◆ State of the art performance in image, video, audio, … processing

2. Supervised learning

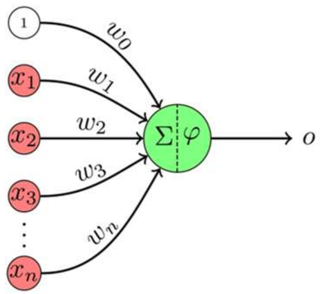
Biological Neurons (40s)

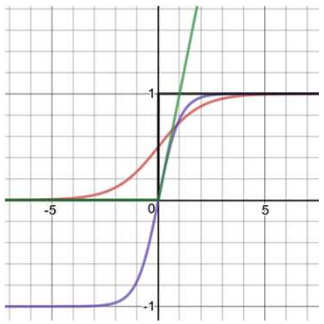
- → A single neuron
  - ◆ Threshold on a sum of inputs
- → Complex organisation
  - ◆ Thousands of inputs
  - ◆ Billions of connections
  - ◆ 3D layout
- → Of much interest
  - ◆ Biology
  - ◆ Neuroscience
  - ◆ Neuropsychology
  - ◆ Artificial intelligence
  - ◆ …



2. Supervised learning

Single Neuron, The Perceptron





$$o = \varphi\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)$$

$$o = \varphi\left(\sum_{i=0}^{n} w_i x_i\right) \quad x_0 = 1$$

Step function
Sigmoid
Tanh
Rectified Linear Unit

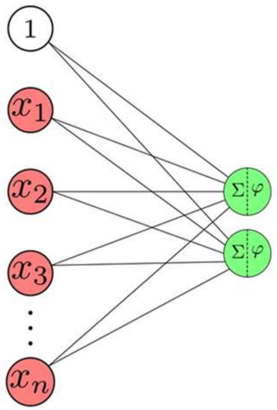
2. Supervised learning

Multiple Outputs: Fully Connected Layer

- ➔ **One “perceptron” per output**
- ➔ **Different weights for each output**

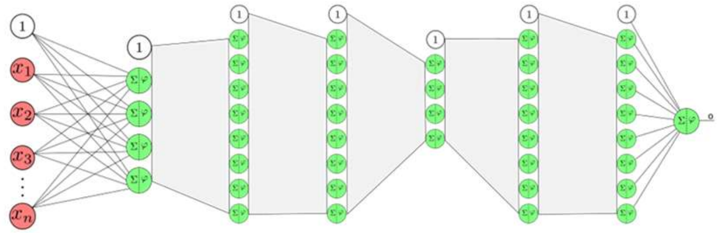
$$o_1 = \varphi\left(w_0^1 + \sum_{i=1}^{n} w_i^1 x_i\right)$$

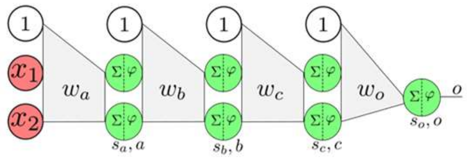
$$o_2 = \varphi\left(w_0^2 + \sum_{i=1}^{n} w_i^2 x_i\right)$$



2. Supervised learning

Multilayer Perceptron (MLP)



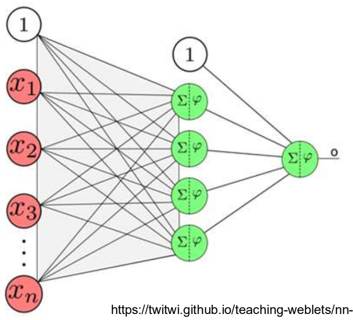


2. Supervised learning

Two-layer Perceptron

$$\rightarrow h_j = \varphi\left(w_0^j + \sum_{i=1}^{n} w_i^j x_i\right)$$

$$o = \varphi\left(w_0^o + \sum_{j=1}^{4} w_j^o h_j\right)$$



https://twitwi.github.io/teaching-weblets/nn-3d-steps/nn-3d-one-step.html

2. Supervised learning

Expressive Power of Multilayer Perceptrons

(universal approximation theorem)

We can approximate any continuous function
with a multilayer perceptron that has a single hidden layer (not “deep”)
but that is sufficiently wide (a lot of neurons on the hidden layer)

- ➔ Question: should we prefer adding
  - ◆ more layers (deeper)?
  - ◆ more neurons in a single hidden layer (wider)?

⇒ **Deeper networks generalize better**

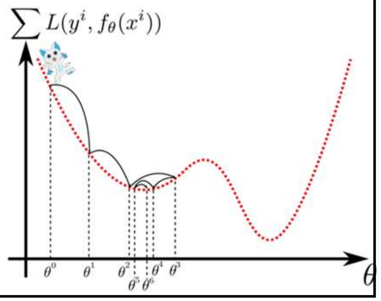
- ➔ Most probably because they create **successive abstractions** (observed empirically, on many real problems)

2. Supervised learning

Training Neural Networks (finding θ, a good set of weights)

- ➔ **Originally, the perceptron algorithm**
- ➔ **Today, mainly, gradient descent (and variants)**
  - ◆ We want to optimize $\mathcal{L}(\theta) = \sum_i l(f_\theta(x^i), y^i)$
  - ◆ Start with random weights θ^0

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta \mathcal{L}(\theta^t)$$



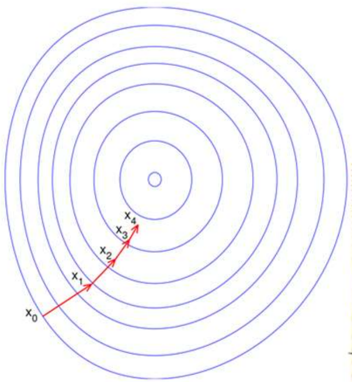
2. Supervised learning

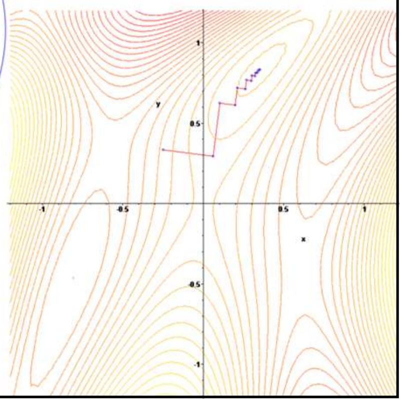
Training Neural Networks (finding θ, a good set of weights)

- ➔ **Today, mainly, gradient descent (and variants)**
  - ◆ We want to optimize $\mathcal{L}(\theta) = \sum_i l(f_\theta(x^i), y^i)$
    (sum over the training set)
  - ◆ Start with random weights θ^0
  - ◆ "**Vanilla**" batch Gradient Descent $\theta^{t+1} = \theta^t - \gamma \nabla_\theta \mathcal{L}(\theta^t)$
  - ◆ **Mini-batch** Gradient Descent iterates over

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta \mathcal{L}_{\mathcal{B}}(\theta^t)$$
$$\mathcal{L}_{\mathcal{B}}(\theta^t) = \sum_{i \in \mathcal{B}} l(f_\theta(x^i), y^i)$$

Each iteration considers a random minibatch of points $\mathcal{B}$
- we have to choose a minibatch size, e.g. $\|\mathcal{B}\| = 64$
- Stochastic gradient descent SGD: **single sample** batch





2. Supervised learning

More About Deep Neural Networks

……… during the whole week