# Introduction to Machine Learning

Journées Calcul et Apprentissage

Aurélien Garivier

24-25 avril 2019

# Table of contents

# What is Machine Learning?

# Why Machine Learning?

Actualité

## Yann LeCun, Geoffrey Hinton et Yoshua Bengio reçoivent le prix Turing

Par **Stephane Nachez** - 27 mars 2019


## LE MACHINE LEARNING PROVOQUE UNE CRISE DANS LE DOMAINE DE LA SCIENCE

Bastien L · 19 février 2019 · Analytics, Data Analytics, Intelligence artificielle · 1 commentaire

***Le Machine Learning est en train de provoquer une grave crise de reproductibilité dans le domaine de la science. C'est ce qu'affirme la statisticienne Genevera Allen de la Rice University dans le cadre de la conférence AAAS Annual Meeting.***

De plus en plus de chercheurs utilisent le Machine Learning pour analyser des données et y détecter des tendances. Cependant, dans le cadre de la conférence scientifique AAAS Annual Meeting, la statisticienne Genevera Allen de la Rice University a tenu à tirer la sonnette d'alarme. Selon elle, le Machine Learning est en passe de provoquer **une crise de reproductibilité dans le domaine de la science**.

SHARE

SPECIAL VIEWPOINTS

## Machine Learning for Science: State of the Art and Future Prospects


Eric Mjolsness*, Dennis DeCoste
+ See all authors and affiliations



*Science* 14 Sep 2001:
Vol. 293, Issue 5537, pp. 2051-2055
DOI: 10.1126/science.293.5537.2051


Article | Figures & Data | Info & Metrics | eLetters | PDF


### Abstract

Recent advances in machine learning methods, along with successful applications across a wide variety of fields such as planetary science and bioinformatics, promise powerful new tools for practicing scientists. This viewpoint highlights some useful characteristics of modern machine learning methods and their relevance to scientific applications. We conclude with some speculations on near-term progress and promising directions.

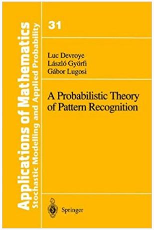

PUBLIC RELEASE: 15-FEB-2019

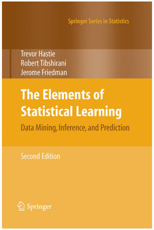
## Can we trust scientific discoveries made using machine learning?

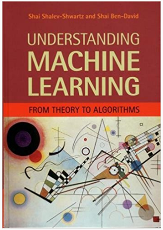
*Rice U. expert: Key is creating ML systems that question their own predictions*

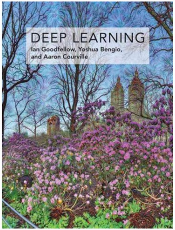
RICE UNIVERSITY

# Where to learn more?



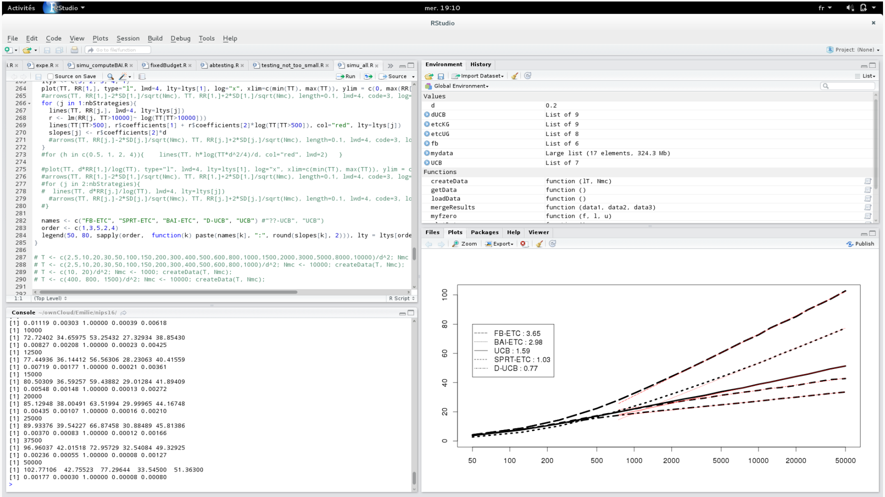



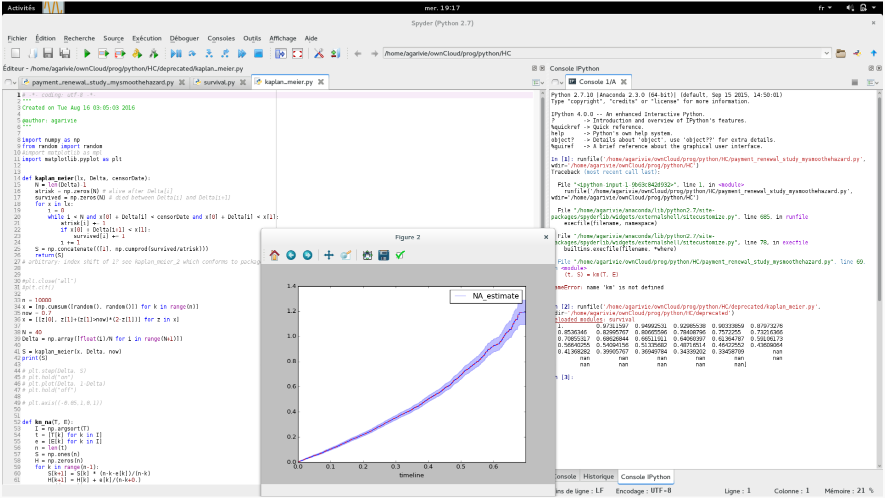

WikiStat

scikit learn

# What do I need to practice ML? R



# What do I need to practice ML? python



# What do I need to practice ML? scikit-learn

# Outline

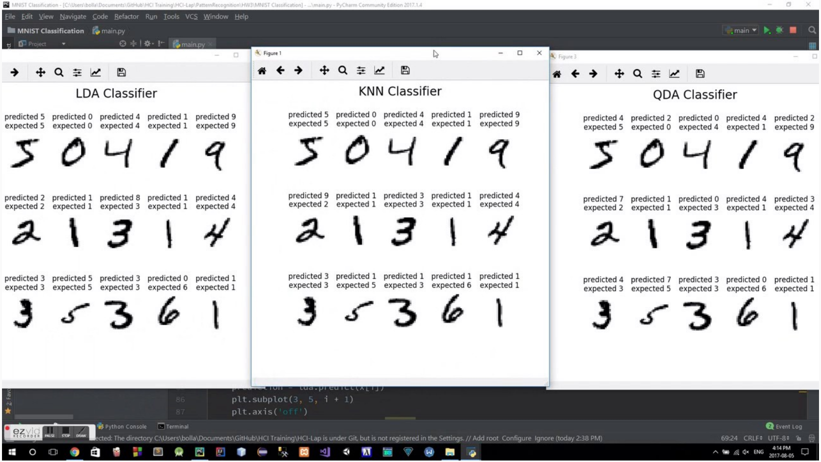
# What is Machine Learning?

- Algorithms operate by building a model from **example** inputs in order to make data-driven **predictions or decisions**...
- ...rather than following strictly static program instructions: useful when designing and programming explicit algorithms is unfeasible or poorly efficient.

## Within Artificial Intelligence

- evolved from the study of pattern recognition and computational learning theory in artificial intelligence.
- AI: emulate cognitive capabilities of humans
  (big data: humans learn from abundant and diverse sources of data).
- a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving".

# Example: MNIST dataset



# Definition

**Arthur Samuel (1959)**

Field of study that gives computers the ability to learn without being explicitly programmed

**Tom M. Mitchell (1997)**

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

# Machine Learning: Typical Problems

- spam filtering, text classification
- optical character recognition (OCR)
- search engines
- recommendation platforms
- speach recognition software
- computer vision
- bio-informatics, DNA analysis, medicine
- ...

For each of this task, it is possible but very inefficient to write an explicit program reaching the prescribed goal.

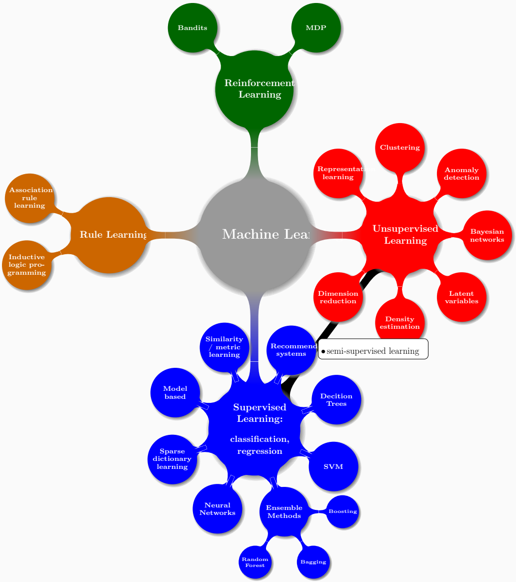
It proves much more succesful to have a machine infer what the good decision rules are.

# What is Statistical Learning?

= Machine Learning using statistics-inspired tools and guarantees

- Importance of **probability**- and **statistics**-based methods
  → **Data Science** (Michael Jordan)
- **Computational Statistics**: focuses in prediction-making through the use of computers together with statistical models (ex: Bayesian methods).
- **Data Mining** (unsupervised learning) focuses more on exploratory data analysis: discovery of (previously) unknown properties in the data. This is the analysis step of Knowledge Discovery in Databases.
- Machine Learning has more **operational** goals
  Ex: ~~consistency~~ → oracle inequalities
  Models (if any) are *instrumental*.
  ML more focused on *correlation*, less on *causality* (now changing).
- Strong ties to **Mathematical Optimization**, which furnishes methods, theory and application domains to the field

# What is ML composed of?



# Outline

What is Machine Learning?

Data and Learning Algorithms

Classification Framework

First Algorithms: fitting versus generalizing

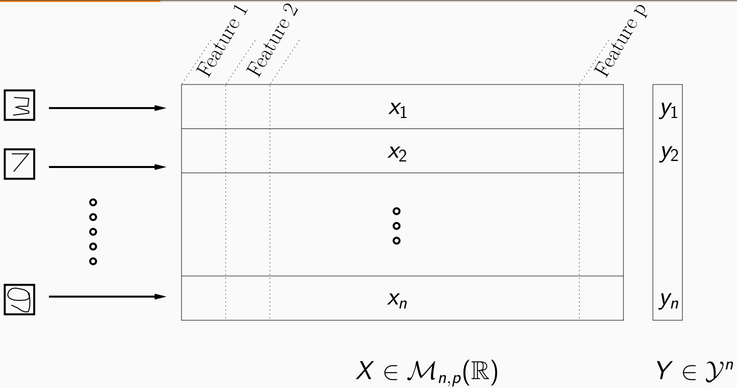
Nearest-Neighbor Classification

Empirical Risk Minimization

Support Vector Machines

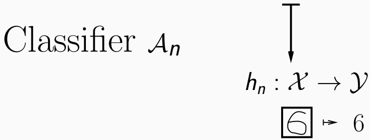
Neural Networks

# What is a classifier?




Data: $n$-by-$p$ matrix $X$

- $n$ examples = points of observations
- $p$ features = characteristics measured for each example



# Data repositories

- Inside R: package `datasets`
- Inside python/scikitlearn: package `sklearn.datasets`
- UCI Machine Learning Repository
- Challenges: Kaggle, etc.

# Outline

# Statistical Learning Hypothesis

**Assumption**

- The examples $(X_i, Y_i)_{1 \le i \le n}$ are iid samples of an unknown joint distribution $\mathcal{D}$;
- The points to classify later are also independent draws of the *same* distribution $\mathcal{D}$.

Hence, for every *decision rule* $h : \mathcal{X} \to \mathcal{Y}$ we can define the *risk*

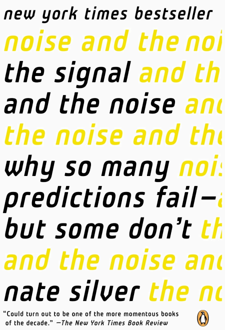
$$L_{\mathcal{D}}(h) = \mathbb{P}_{(X,Y)\sim\mathcal{D}}\big(h(X) \neq Y\big) = \mathcal{D}\Big(\big\{(x, y) : h(x) \neq y\big\}\Big) .$$

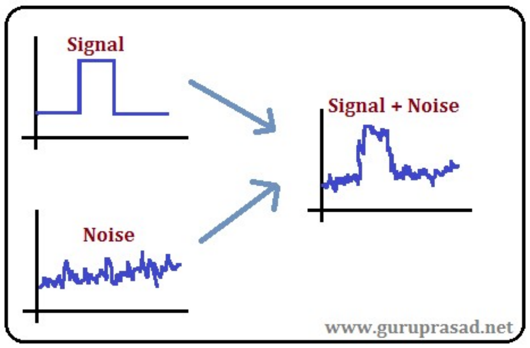
The goal of the learning algorithm is to *minimize the expected risk*:

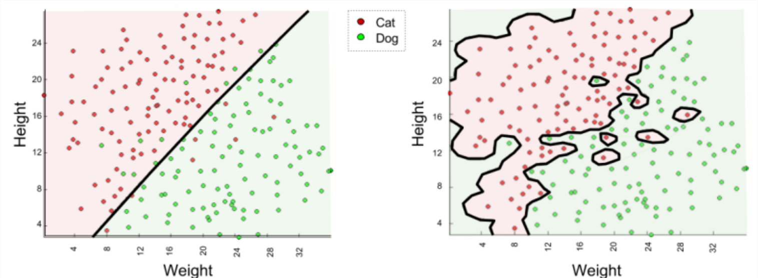
$$R_n(\mathcal{A}_n) = \mathbb{E}_{\mathcal{D}^{\otimes n}}\left[L_{\mathcal{D}}\Big(\underbrace{\mathcal{A}_n\big((X_1, Y_1), \ldots, (X_n, Y_n)\big)}_{\hat{h}_n}\Big)\right]$$

for *every* distribution $\mathcal{D}$, using only the examples.

# Signal and Noise



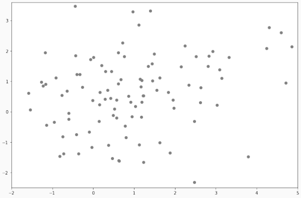
## Example: Character Recognition

| | |
|---|---|
| Domain set $\mathcal{X}$<br>Label set $\mathcal{Y}$<br>Joint distribution $\mathcal{D}$ | $28 \times 28$ images<br>$\{0, 1, \ldots, 9\}$<br>? |
| Prediction function $h \in \mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$<br>Risk $R(h) = P_{\mathcal{D}}\big(h(X) \neq Y\big)$ | |
| Sample $S_n = \big\{(X_i, Y_i)\big\}_{i=1}^{n}$<br>Empirical risk<br>$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(X_i) \neq Y_i\}$ | MNIST dataset |
| Learning algorithm<br>$\mathcal{A} = (\mathcal{A}_n)_n$, $\mathcal{A}_n : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}$<br>Expected risk $R_n(\mathcal{A}) = \mathbb{E}_n\Big[L_{\mathcal{D}}\big(\mathcal{A}_n(S_n)\big)\Big]$ | neural nets, boosting... |

# Two visions of $\mathcal{D}$

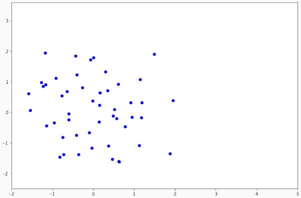
**As a pair** $(\mathcal{D}_x, k)$, where

- for $A \subset \mathcal{X}$, $\mathcal{D}_x(A) = \mathcal{D}\big(A \times \mathcal{Y}\big)$ is the marginal distribution of $X$,
- and for $x \in \mathcal{X}$ and $B \subset \mathcal{Y}$, $k(B|x) = \mathcal{D}\big(Y \in B|X = x)$ is (a version of) the conditional distribution of $Y$ given $X$.



**As a pair** $\Big(\mathcal{D}_y, \left(\mathcal{D}(\cdot|y)\right)_y\Big)$, where

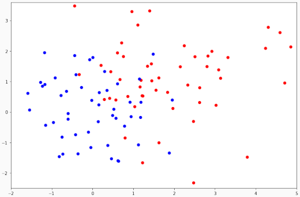
- for $y \in \mathcal{Y}$, $\mathcal{D}_Y(y) = \mathcal{D}\big(\mathcal{X} \times y\big)$ is the marginal distribution of $Y$,
- and for $A \subset \mathcal{X}$ and $y \in \mathcal{Y}$, $\mathcal{D}(A|y) = \mathcal{D}\big(X \in A|Y = y)$ is the conditional distribution of $X$ given $Y = y$.



# Two visions of $\mathcal{D}$

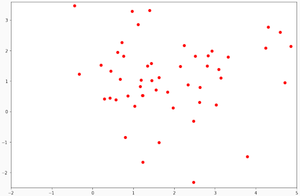
**As a pair** $(\mathcal{D}_x, k)$, where

- for $A \subset \mathcal{X}$, $\mathcal{D}_x(A) = \mathcal{D}\big(A \times \mathcal{Y}\big)$ is the marginal distribution of $X$,
- and for $x \in \mathcal{X}$ and $B \subset \mathcal{Y}$, $k(B|x) = \mathcal{D}\big(Y \in B|X = x\big)$ is (a version of) the conditional distribution of $Y$ given $X$.



**As a pair** $\Big(\mathcal{D}_y, \big(\mathcal{D}(\cdot|y)\big)_y\Big)$, where

- for $y \in \mathcal{Y}$, $\mathcal{D}_Y(y) = \mathcal{D}\big(\mathcal{X} \times y\big)$ is the marginal distribution of $Y$,
- and for $A \subset \mathcal{X}$ and $y \in \mathcal{Y}$, $\mathcal{D}(A|y) = \mathcal{D}\big(X \in A|Y = y\big)$ is the conditional distribution of $X$ given $Y = y$.



# Performance Limit: Bayes Classifier

Consider binary classification $\mathcal{Y} = \{0, 1\}$, $\eta(x) := \mathcal{D}(Y = 1 | X = x)$.

**Theorem**

*The Bayes classifier is defined by*
$h^*(x) = \mathbb{1}\big\{\eta(x) \geq 1/2\big\} = \mathbb{1}\big\{\eta(x) \geq 1 - \eta(x)\big\} = \mathbb{1}\big\{2\eta(x) - 1 \geq 0\big\}$.
*For every classifier* $h : \mathcal{X} \to \mathcal{Y} = \{0, 1\}$,

$$L_{\mathcal{D}}(h) \geq L_{\mathcal{D}}(h^*) = \mathbb{E}\Big[\min\big(\eta(X), 1 - \eta(X)\big)\Big] .$$

*The Bayes risk* $L^*_{\mathcal{D}} = L_{\mathcal{D}}(h^*)$ *is called the* **noise** *of the problem.*

*More precisely,*

$$L_{\mathcal{D}}(h) - L_{\mathcal{D}}(h^*) = \mathbb{E}\Big[\big|2\eta(X) - 1\big| \, \mathbb{1}\big\{h(X) \neq h^*(X)\big\}\Big] .$$

Extends to $|\mathcal{Y}| > 2$.

# First Algorithms: fitting versus generalizing

# Nearest-Neighbor Classification

# The Nearest-Neighbor Classifier

We assume that $\mathcal{X}$ is a metric space with distance $d$.

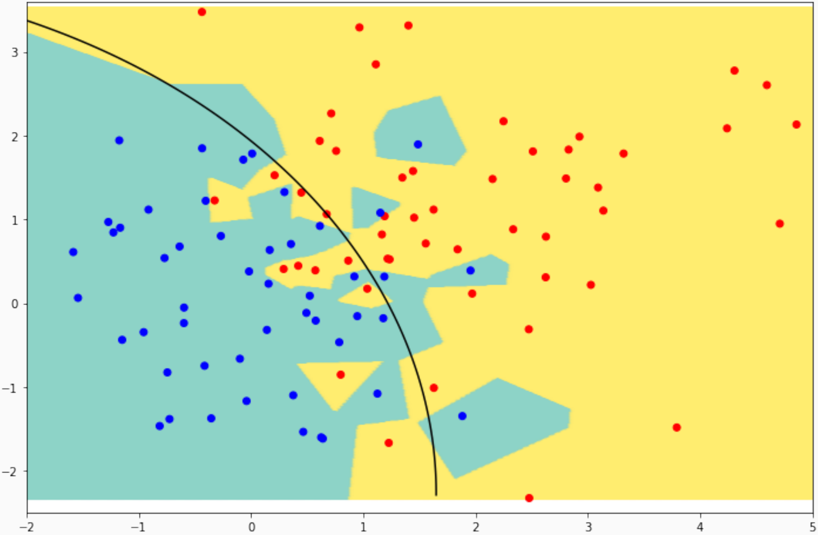
The nearest-neighbor classifier $\hat{h}_n^{NN} : \mathcal{X} \to \mathcal{Y}$ is defined as

$$\hat{h}_n^{NN}(x) = Y_I \quad \text{where} \quad I \in \underset{1 \le i \le n}{\arg\min}\; d(x - X_i)\ .$$

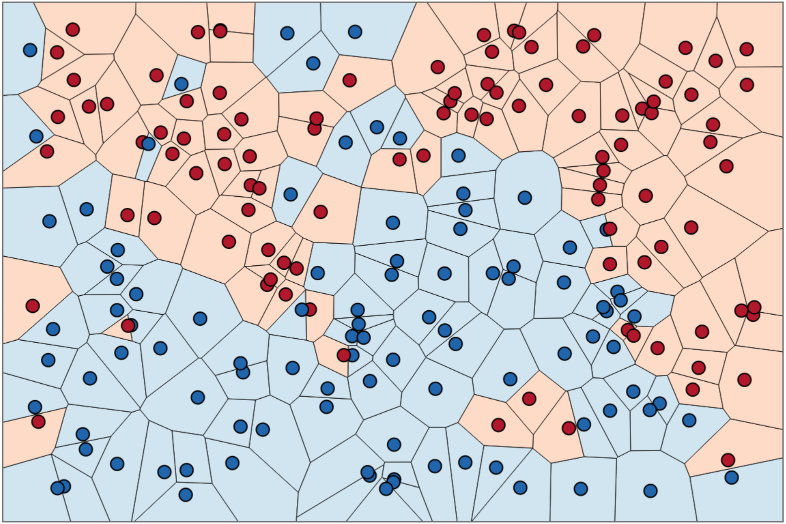
Typical distance: $L^2$ norm on $\mathbb{R}^d$: $\|x - x'\| = \sqrt{\sum_{j=1}^d (x_i - x_i')^2}\ .$

Buts many other possibilities: Hamming distance on $\{0,1\}^d$, etc.

# Numerically



# Numerically



# The most simple analysis of the most simple algorithm

**A1.** $\mathcal{Y} = \{0, 1\}$.

**A2.** $\mathcal{X} = [0, 1[^d$.

**A3.** $\eta$ is $c$-Lipschitz continuous:

$$\forall x, x' \in \mathcal{X}, \left|\eta(x) - \eta(x')\right| \leq c\left\|x - x'\right\| .$$

**Theorem**

*Under the previous assumptions, for all distributions $\mathcal{D}$ and all $m \geq 1$*

$$L_{\mathcal{D}}\left(\hat{h}_n^{NN}\right) \leq 2L_{\mathcal{D}}^* + \frac{3c\sqrt{d}}{n^{1/(d+1)}} .$$

# Proof Outline

- Conditioning: as $I(x) = \arg\min_{1\leq i\leq n} \|x - X_i\|$,

$$L_D(\hat{h}_n^{NN}) = \mathbb{E}\Big[\mathbb{E}\big[\mathbb{1}\{Y \neq Y_{I(X)}\}\big|X, X_1, \dots, X_n\big]\Big] \,.$$

- $Y \sim \mathcal{B}(p),\ Y' \sim \mathcal{B}(q) \implies \mathbb{P}(Y \neq Y') \leq 2\min(p, 1-p) + |p-q|$,

$$\mathbb{E}\Big[\mathbb{1}\{Y \neq Y_{I(X)}\}|X, X_1, \dots, X_n\Big] \leq 2\min\big(\eta(X), 1-\eta(X)\big) + c\,\big\|X - X_{I(X)}\big\| \,.$$

- Partition $\mathcal{X}$ into $|\mathcal{C}| = T^d$ cells of diameter $\sqrt{d}/T$:

$$\mathcal{C} = \left\{\left[\frac{j_1 - 1}{T}, \frac{j_1}{T}\right[ \times \cdots \times \left[\frac{j_d - 1}{T}, \frac{j_d}{T}\right[, \quad 1 \leq j_1, \dots, j_d \leq T\right\} \,.$$

- 2 cases: either the cell of $X$ is occupied by a sample point, or not:

$$\big\|X - X_{I(X)}\big\| \leq \sum_{c\in\mathcal{C}} \mathbb{1}\{X \in c\} \left(\frac{\sqrt{d}}{T} \mathbb{1}\bigcup_{i=1}^{n} \{X_i \in c\} + \sqrt{d}\,\mathbb{1}\bigcap_{i=1}^{n} \{X_i \notin c\}\right) \,.$$

- $\implies \mathbb{E}\big[\|X - X_{I(X)}\|\big] \leq \frac{\sqrt{d}}{T} + \frac{\sqrt{d}\,T^d}{e\,n}$ and choose $T = \Big\lfloor n^{\frac{1}{d+1}} \Big\rfloor$.

# What does the analysis say?

- Is it loose? (sanity check: uniform $\mathcal{D}_X$)
- *Non-asympototic* (finite sample bound)
- The second term $\frac{3c\sqrt{d}}{n^{1/(d+1)}}$ is *distribution independent*
- Does not give the trajectorial decrease of risk
- Exponential bound $d$ (cannot be avoided...)
  $\implies$ *curse of dimensionality*
- How to improve the classifier?

Let $\mathcal{X}$ be a (pre-compact) metric space with distance $d$.

**k-NN classifier**

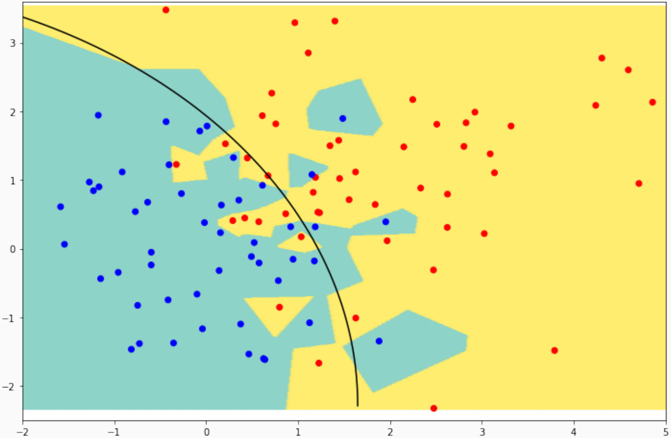
$h^{kNN} : x \mapsto \mathbb{1}\{\hat{\eta}(x) \geq 1/2\}$ = plugin for Bayes classifier with estimator

$$\hat{\eta}(x) = \frac{1}{k} \sum_{j=1}^{k} Y_{(j)}(X)$$

where

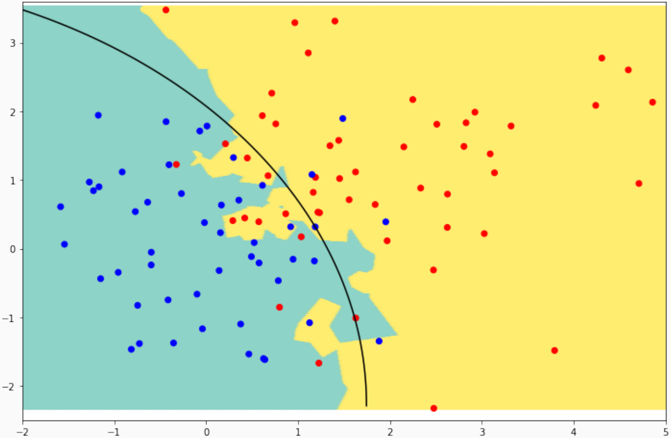
$$d\big(X_{(1)}(X), X\big) \leq d\big(X_{(2)}(X), X\big) \leq \cdots \leq d\big(X_{(n)}(X), X\big) .$$

# More neighbors are better?



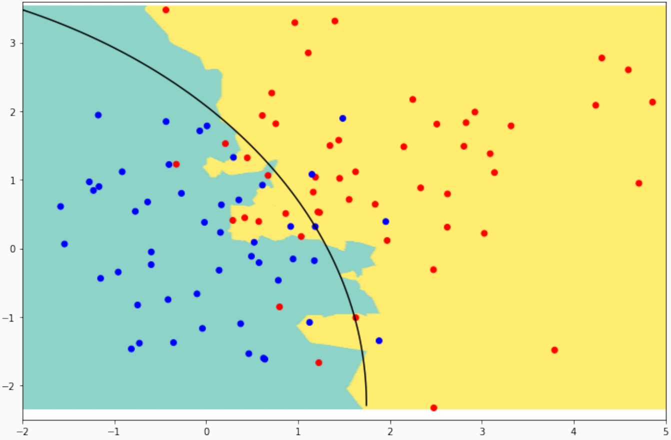

$k = 1$

# More neighbors are better?



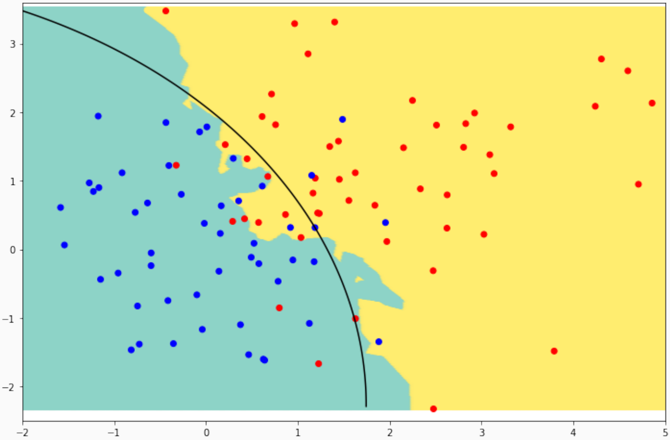

$k = 3$

# More neighbors are better?



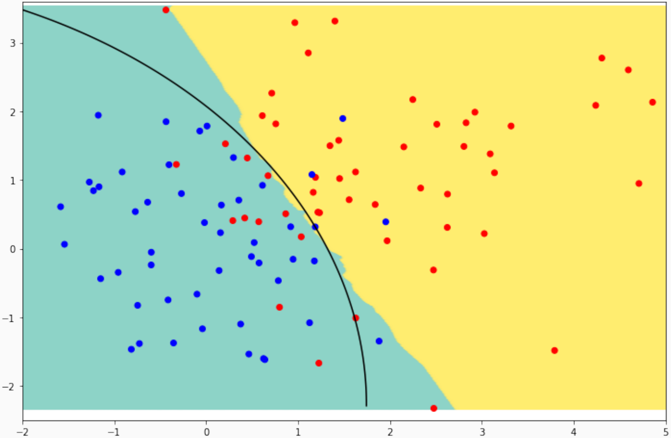

$k = 5$

# More neighbors are better?



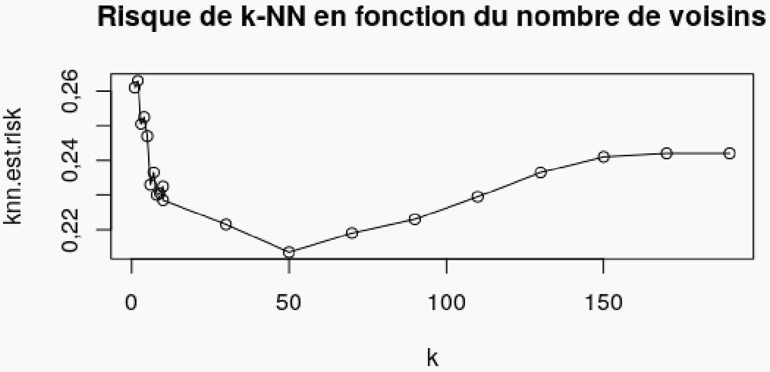

$k = 7$

# More neighbors are better?




$k = 75$

# Bias-Variance tradeoff



## Risk bound

Let $\mathcal{C}_\epsilon$ be an $\epsilon$-covering of $\mathcal{X}$:

$$\forall x \in X, \exists x' \in \mathcal{C}_\epsilon : d(x, x') \leq \epsilon \,.$$

### Excess risk for k-nearest-neighbours

If $\eta$ is $c$-Lipschitz continuous: $\forall x, x' \in \mathcal{X}, \big|\eta(x) - \eta(x')\big| \leq c\, d\big(x, x'\big)$, then for all $k \geq 2$ and all $n \geq 1$:

$$\begin{aligned} L\big(\hat{h}_n^{kNN}\big) - L\big(h^*\big) &\leq \frac{1}{\sqrt{k\, e}} + \frac{2k|\mathcal{C}_\epsilon|}{n} + 4c\epsilon \\ &\leq \frac{1}{\sqrt{k\, e}} + (2+4c)\left(\frac{\alpha k}{n}\right)^{\frac{1}{d+1}} \quad \begin{cases} \text{for } \epsilon = \left(\frac{\alpha k}{n}\right)^{\frac{1}{d+1}} \\ \text{if } |\mathcal{C}_\epsilon| \leq \alpha \epsilon^{-d} \end{cases}, \\ &\leq (3+4c)\left(\frac{\alpha}{n}\right)^{\frac{1}{d+3}} \quad \text{for } k = \left(\frac{n}{\alpha}\right)^{\frac{2}{d+3}} \,. \end{aligned}$$

# Room for improvement

- Lower bound? in $n^{-\frac{1}{d}}$.
- Margin conditions
  $\implies$ fast rates
- More regularity?
  $\implies$ weighted nearest neighbors
- Is regularity required everywhere?
  $\implies$ What matters are the balls of mass $\approx k/n$ near the decision boundary.
- 2 "parameters":
  - obvious: the number of neighbors $k$ (bias-variance tradeoff)
  - hidden: the distance $d$ (real problem)

# Curse of dimensionality: No free lunch theorem

**Theorem**

Let $c > 1$ be a Lipschitz constant. Let $A$ be any learning algorithm for binary classification over a domain $\mathcal{X} = [0, 1]^d$. If the training set size is $n \leq (c+1)^d/2$, then there exists a distribution $\mathcal{D}$ over $[0, 1]^d \times \{0, 1\}$ such that:

- $\eta(x)$ is $c$-Lipschitz;
- the Bayes error of the distribution is 0;
- with probability at least $1/7$ over the choice of $S_n \sim \mathcal{D}^{\otimes n}$,

$$L_{\mathcal{D}}\big(A(S_n)\big) \geq \frac{1}{8} .$$

# Empirical Risk Minimization

# Going empirical

Idea for every candidate rule $h$ in an *hypothesis class* $\mathcal{H}$, replace the unknown risk

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(X,Y)\sim\mathcal{D}}\big(h(X) \neq Y\big)$$

by the computable *empirical risk*

$$L_{S_n}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{h(X_i) \neq Y_i\}$$

and use some *uniform law of large numbers:*

$$\mathbb{P}_D\left(\sup_{h\in\mathcal{H}} \big|L_{S_n}(h) - L_{\mathcal{D}}(h)\big| > c\sqrt{\frac{D_{\mathcal{H}}\log(n) + \log\frac{1}{\delta}}{n}}\right) \leq \delta$$

where $D_{\mathcal{H}}$ is the *Vapnik-Chervonenkis dimension* of $\mathcal{H}$.

# Empirical Risk minimization

*Uniform law of large numbers:*

$$\mathbb{P}_D \left( \sup_{h \in \mathcal{H}} \left| L_{S_n}(h) - L_{\mathcal{D}}(h) \right| > c \sqrt{\frac{D_{\mathcal{H}} \log(n) + \log \frac{1}{\delta}}{n}} \right) \leq \delta \, .$$

$\rightarrow$ *Empirical Risk Minimizer*:

$$\hat{h}_n = \arg\min_{h \in \mathcal{H}} L_{S_n}(h) \, .$$

Good if

- the class $\mathcal{H}$ is not too large
- the number $n$ of examples is large enough

so as to ensure that $c \sqrt{\frac{D_{\mathcal{H}} \log(n) + \log \frac{1}{\delta}}{n}} \leq \epsilon$.

$\rightarrow$ *Sample complexity* = number of examples required to have an $\epsilon$-optimal rule in the hypothesis class $\mathcal{H} = O\left(\frac{D_{\mathcal{H}}}{\epsilon^2}\right)$ .

# The class of halfspaces

**Definition**

The class of linear (affine) functions on $\mathcal{X} = \mathbb{R}^d$ is defined as

$$L_d = \left\{ h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R} \right\}, \quad \text{where } h_{w,b}(x) = \langle w, x \rangle + b \,.$$

The hypothesis class of halfspaces for binary classification is defined as

$$\mathcal{HS}_d = \operatorname{sign} \circ L_d = \left\{ x \mapsto \operatorname{sign}\left( h_{w,b}(x) \right) : h_{w,b} \in L_d \right\}$$

where $\operatorname{sign}(u) = \mathbb{1}\{u \geq 0\} - \mathbb{1}\{u < 0\}$. *Depth 1 neural networks.*

By taking $\mathcal{X}' = \mathcal{X} \times \{1\}$ and $d' = d + 1$, we may omit the bias $b$ and focus on functions $h_w(x) = \langle w, x \rangle$.

**Property**

The VC-dimension of $\mathcal{HS}_d$ is equal to $d + 1$.

Corollary: the class of halfspaces is learnable with *sample complexity* $O\left(\frac{d+1+\log(1/\delta)}{\epsilon^2}\right)$.

# Realizable case: Learning halfspaces with a linear program solver

*Realizable case*: there exists $w^*$ such that $\forall i \in \{1, \dots, n\}$, $y_i \langle w^*, x_i \rangle > 0$.

Then there exists $\bar{w} \in \mathbb{R}^d$ such that $\forall i \in \{1, \dots, n\}$, $y_i \langle \bar{w}, x_i \rangle \geq 1$: if we can find one, we have an ERM.

Let $A \in \mathcal{M}_{n,d}(\mathbb{R})$ be defined by $A_{i,j} = y_i \, x_{i,j}$, and let $v = (1, \dots, 1) \in \mathbb{R}^m$. Then any solution of the linear program

$$\max_{w \in \mathbb{R}^d} \langle 0, w \rangle \quad \text{subject to} \quad Aw \geq v$$

is an ERM. It can thus be computed in polynomial time.

# Rosenblatt's Perceptron algorithm

**Algorithm:** Batch Perceptron

**Data:** training set $(x_1, y_1), \dots, (x_n, y_n)$

$w_0 \leftarrow (0, \dots, 0)$

$t \geq 0$

**while** $\exists i_t : y_{i_t} \langle w_t, x_{i_t} \rangle \leq 0$ **do**

  $w_{t+1} \leftarrow w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}$

  $t \leftarrow t + 1$

**return** $w_t$

Each updates helps reaching the solution, since

$$y_{i_t} \langle w_{t+1}, x_{i_t} \rangle = y_{i_t} \left\langle w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}, x_{i_t} \right\rangle = y_{i_t} \langle w_t, x_{i_t} \rangle + \|x_{i_t}\| \,.$$

Relates to a coordinate descent (stepsize does not matter).

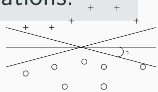
# Convergence of the Perceptron algorithm

**Theorem**

Assume that the dataset $S_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ is linearly separable and let the *separation margin* $\gamma$ be defined as:

$$\gamma = \max_{w \in \mathbb{R}^d : \|w\|=1} \min_{1 \le i \le n} \frac{y_i \langle w, x_i \rangle}{\|x_i\|} .$$

Then the perceptron algorithm stops after at most $1/\gamma^2$ iterations.



**Proof:** Let $w^*$ be such that $\forall 1 \le i \le n$, $\quad \dfrac{y_i \langle w^*, x_i \rangle}{\|x_i\|} \ge \gamma$ .

- If iteration $t$ is necessary, then

$$\langle w^*, w_{t+1} - w_t \rangle = y_{i_t} \left\langle w^*, \frac{x_{i_t}}{\|x_{i_t}\|} \right\rangle \ge \gamma \quad \text{and hence } \langle w^*, w_t \rangle \ge \gamma t \ .$$

- If iteration $t$ is necessary, then

$$\|w_{t+1}\|^2 = \left\| w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|} \right\|^2 = \|w_t\|^2 + \underbrace{\frac{2 y_{i_t} \langle w_t, x_{i_t} \rangle}{\|x_{i_t}\|}}_{\le 0} + y_{i_t}^2 \le \|w_t\|^2 + 1$$

  and hence $\|w_t\|^2 \le t$, or $\|w_t\| \le \sqrt{t}$.

- As a consequence, the algorithm iterates at least $t$ times if

$$\gamma t \le \langle w^*, w_t \rangle \le \|w_t\| \le \sqrt{t} \quad \Longrightarrow \quad t \le \frac{1}{\gamma^2} \ .$$

In the worst case, the number of iterations can be exponentially large in the dimension $d$. Usually, it converges quite fast. If $\forall i$, $\|x_i\| = 1$, $\gamma = d(S, D)$ where $D = \{x : \langle w^*, x \rangle = 0\}$.

# Computational difficulty of agnostic learning, and surrogates

**NP-hardness of computing the ERM for halfspaces**

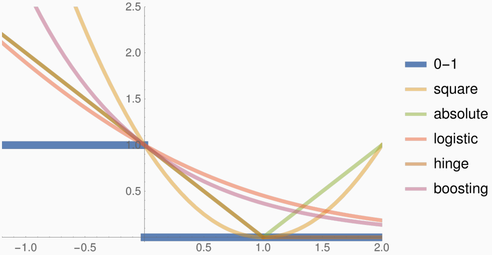
Computing an ERM in the agnostic case is NP-hard.

See *On the difficulty of approximately maximizing agreements*, by Ben-David, Eiron and Long.

Since the 0-1 loss

$$L_{S_n}(h_w) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \langle w, x_i \rangle < 0\}$$

is intractable to minimize in the agnostic case, one may consider *surrogate* loss functions

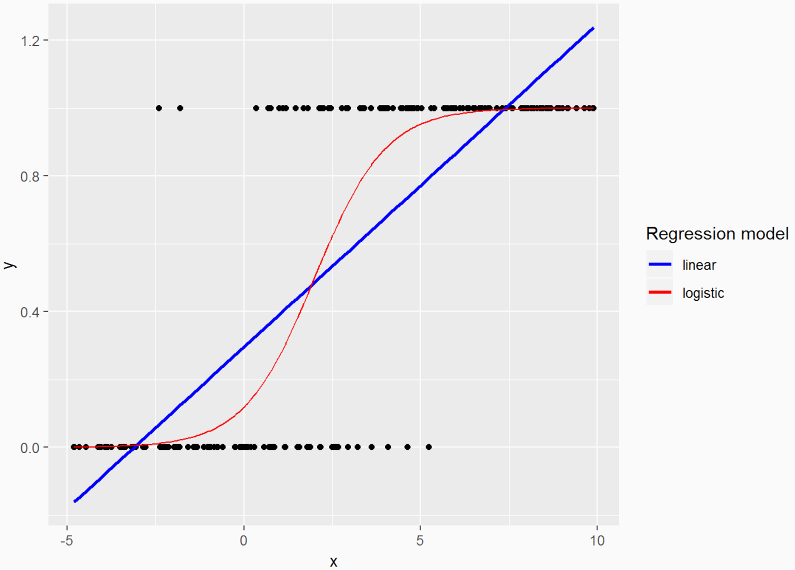



$$L_{S_n}(h_w) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i \langle w, x_i \rangle\big) \ ,$$

where the loss function $\ell : \mathbb{R} \to \mathbb{R}^+$

- dominates the function $\mathbb{1}\{u < 0\}$,
- and leads to a "simple" optimization problem (e.g. *convex*).

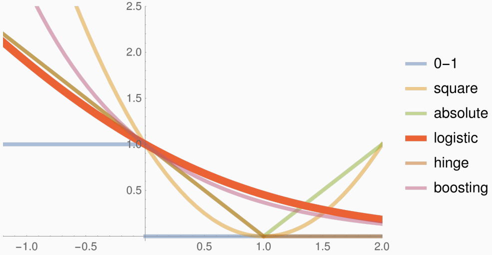
# Logistic Regression



# Logistic loss $\mathcal{Y} = \{-1, 1\}$

Statistics: "logistic regression":

$$P_w\big(Y = y | X = x\big) = \frac{1}{1 + \exp\big(-y\,\langle w, x\rangle\big)}$$




log with base 2 here so that $\ell(0) = 1$

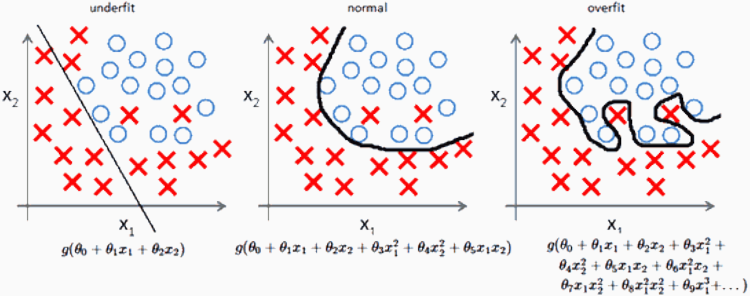
$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \log\big(1 + \exp(-y_i\langle w, x_i\rangle)\big) \ ,$$

Convex minimization problem, can be solved by Newton's algorithm (in small dimension) or stochastic gradient descent (in higher dimension).

# Structural Risk minimization

What if $\mathcal{H} = \bigcup_{d=1}^{\infty} \mathcal{H}_d$, with $\mathcal{H}_d \subset \mathcal{H}_{d+1}$?

$\rightarrow$ empirical risk minimization fails




$\rightarrow$ *structural* risk minimization:

$$\hat{h}_n = \underset{d \geq 1, h \in \mathcal{H}_d}{\arg\min} \; L_{S_n}(h) + D_{\mathcal{H}_d} \log(n) \, .$$

# Support Vector Machines

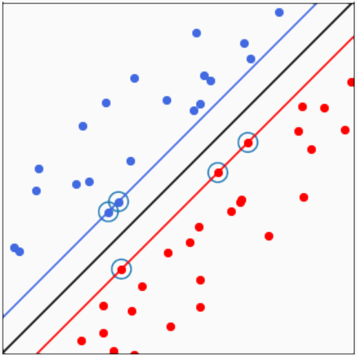
# Margin for linear separation

- Training sample $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$.
- Linearly separable if there exists a halfspace $h = (w, b)$ such that $\forall i, y_i = \text{sign}\left(\langle w, x_i \rangle + b\right)$.
- What is the best separating hyperplane for generalization?

**Distance to hyperplane**

If $\|w\| = 1$, then the distance from $x$ to the hyperplane $h = (w, b)$ is $d(x, \mathcal{H}) = \left|\langle w, x \rangle + b\right|$.

**Proof:** Check that $\min\left\{\|x - v\|^2 : v \in h\right\}$ is reached at $v = x - \left(\langle w, x \rangle + b\right) w$.



# Hard-SVM

Formulation 1:

$$\underset{(w,b):\|w\|=1}{\arg\max} \min_{1\le i\le m} \left|\langle w, x_i\rangle + b\right| \quad \text{such that } \forall i, y_i\big(\langle w, x_i\rangle + b\big) > 0 \ .$$

Formulation 2:

$$\min_{w,b} \|w\|^2 \quad \text{such that } \forall i, y_i\big(\langle w, x_i\rangle + b\big) \ge 1 \ .$$

Remark: $b$ is not penalized.

**Proposition**

The two formulations are equivalent.

**Proof:** if $(w_0, b_0)$ is the solution of Formulation 2, then $\hat{w} = \frac{w_0}{\|w_0\|}$ , $\hat{b} = \frac{b_0}{|w|}$ is a solution of Formulation 1: if $(w^*, b^*)$ is another solution, then letting $\gamma^* = \min_{1\le i\le m} y_i\big(\langle w, x_i\rangle + b\big)$ we see that $\left(\frac{w^*}{\gamma^*}, \frac{b^*}{\gamma^*}\right)$ satisfies the constraint of Formulation 2, hence $\|w_0\| \le \frac{\|w^*\|}{\gamma^*} = \frac{1}{\gamma^*}$ and thus $\min_{1\le i\le m} \left|\langle \hat{w}, x_i\rangle + \hat{b}\right| = \frac{1}{\|w_0\|} \ge \gamma^*$.

# Sample Complexity

**Definition**

A distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$ is *separable with a* $(\gamma, \rho)$*-margin* if there exists $(w^*, b^*)$ such that $\|w^*\| = 1$ and with probability 1 on a pair $(X, Y) \sim \mathcal{D}$, it holds that $\|X\| \leq \rho$ and $Y\big(\langle w^*, X\rangle + b\big) \geq \gamma$.

Remark: by multiplying the $x_i$ by $\alpha$, the margin is mutliplied by $\alpha$.

**Theorem**

For any distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{\pm 1\}$ that satisfies the $(\gamma, \rho)$-separability with margin assumption using a homogenous halfspace, with probability at least $1 - \delta$ over the training set of size $n$ the $0 - 1$ loss of the output of Hard-SVM is at most

$$\sqrt{\frac{4(\rho/\gamma)^2}{n}} + \sqrt{\frac{2\log(2/\delta)]}{n}} \, .$$

Remark: depends on dimension $d$ only thru $\rho$ and $\gamma$.

# Soft-SVM

When the data is not linearly separable, allow *slack variables* $\xi_i$:

$$\min_{w,b,\xi} \lambda \|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \quad \text{such that } \forall i, y_i\big(\langle w, x_i\rangle + b\big) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

$$= \min_{w,b} \lambda \|w\|^2 + L_{S_n}^{\mathrm{hinge}}(w, b) \quad \text{where } \ell^{\mathrm{hinge}}(u) = \max(0, 1-u)\ .$$

**Theorem**

Let $D$ be a distribution over $B(0,\rho) \times \{\pm 1\}$. If $\mathcal{A}_n(S_n)$ is the output of the soft-SVM algorithm on the sample $S$ of $D$ of size $n$,

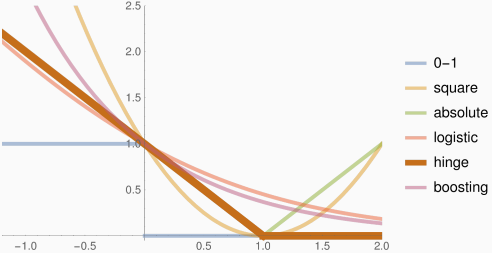
$$\mathbb{E}\Big[L_D^{0-1}\big(\mathcal{A}_n(S_n)\big)\Big] \leq \mathbb{E}\Big[L_D^{\mathrm{hinge}}\big(\mathcal{A}_n(S_n)\big)\Big] \leq \inf_u L_D^{\mathrm{hinge}}(u) + \lambda\|u\|^2 + \frac{2\rho^2}{\lambda n}\ .$$

For every $B > 0$, setting $\lambda = \sqrt{\frac{2\rho^2}{B^2 n}}$ yields:

$$\mathbb{E}\Big[L_D^{0-1}\big(\mathcal{A}_n(S_n)\big)\Big] \leq \mathbb{E}\Big[L_D^{\mathrm{hinge}}\big(\mathcal{A}_n(S_n)\big)\Big] \leq \inf_{w:\|w\|\leq B} L_D^{\mathrm{hinge}}(w) + \sqrt{\frac{8\rho^2 B^2}{n}}\ .$$

# SVM as a Penalized Empirical Risk Minimizer

Margin maximization leads to




$$L_{S_n}^{\mathrm{hinge}}(h_w) = \frac{1}{n}\sum_{i=1}^{n} \max\left\{0, 1 - y_i\langle w, x_i\rangle\right\} ,$$

convex but non-smooth minimization problem, used with a penalization term $\lambda\|w\|^2$.

# Dual Form of the SVM Optimization Problem

To simplify, we consider only the homogeneous case of hard-SVM. Let

$$g(w) = \max_{\alpha \in [0,+\infty)^n} \sum_{i=1}^{n} \alpha_i \big(1 - y_i \langle w, x_i \rangle\big) = \begin{cases} 0 & \text{if } \forall i, y_i \langle w, x_i \rangle \geq 1, \\ +\infty & \text{otherwise .} \end{cases}$$

Then the hard-SVM problem is equivalent to

$$\begin{aligned} \min_{w : \forall i, y_i \langle w, x_i \rangle \geq 1} \frac{1}{2}\|w\|^2 &= \min_{w} \frac{1}{2}\|w\|^2 + g(w) \\ &= \min_{w} \max_{\alpha \in [0,+\infty)^m} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \alpha_i \big(1 - y_i \langle w, x_i \rangle\big) \\ &\overset{\text{min}-\text{max thm}}{=} \max_{\alpha \in [0,+\infty)^n} \min_{w} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \alpha_i \big(1 - y_i \langle w, x_i \rangle\big) \ . \end{aligned}$$

The inner min is reached at $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and can thus be written as

$$\max_{\alpha \in \mathbb{R}^n, \alpha \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{1 \leq i,j \leq n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \ .$$

# Support vectors

Still for the homogeneous case of hard-SVM:

**Property**

Let $w_0$ be a solution of and let $I = \{i : \left|\langle w_0, x_i \rangle\right| = 1\}$. There exist $\alpha_1, \ldots, \alpha_n$ such that

$$w_0 = \sum_{i \in I} \alpha_i x_i \ .$$

The dual problem involves the $x_i$ only thru scalar products $\langle x_i, x_j \rangle$.

It is of size $n$ (independent of the dimension $d$).

These computations can be extended to the non-homogeneous soft-SVM

$\rightarrow$ **Kernel trick**.

# Numerically solving Soft-SVM

$f(w) = \frac{\lambda}{2}\|w\|^2 + L_S^{\mathrm{hinge}}(w)$ is $\lambda$-strongly convex.
$\rightarrow$ Stochastic Gradient Descent with learning rate $1/(\lambda t)$. Stochastic subgradient of $L_S^{\mathrm{hinge}}(w)$ : $v_t = -y_{I_t} x_{I_t} \mathbb{1}\{y_{I_t}\langle w, x_{I_t}\rangle < 1\}$.

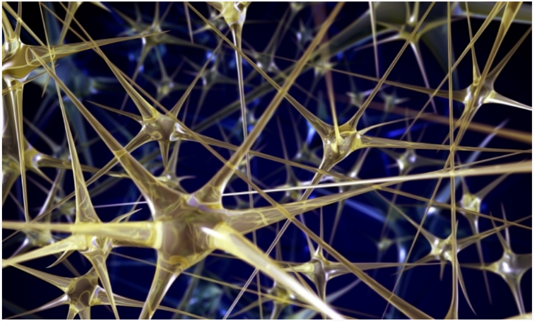
$$w_{t+1} = w_t - \frac{1}{\lambda t}(\lambda w_t + v_t) = \frac{t-1}{t} w_t - \frac{1}{\lambda t} v_t = -\frac{1}{\lambda t}\sum_{i=1}^{t} v_t \ .$$

**Algorithm:** SGD for Soft-SVM

Set $\theta_0 = 0$
**for** $t = 0 \dots T-1$ **do**
  Let $w_t = \frac{1}{\lambda t}\theta_t$
  Pick $I_t \sim \mathcal{U}\big(\{1, \dots, n\}\big)$
  **if** $y_{I_t}\langle w_t, x_{I_t}\rangle < 1$ **then**
    $\theta_{t+1} \leftarrow \theta_t + y_{I_t} x_{I_t}$
  **else**
    $\theta_{t+1} \leftarrow \theta_t$
**return** $\bar{w}_T = \frac{1}{T}\sum_{t=0}^{T-1} w_t$

# Neural Networks

# One-layer network



Src: `http://insanedev.co.uk/open-cranium/`

# One-layer network

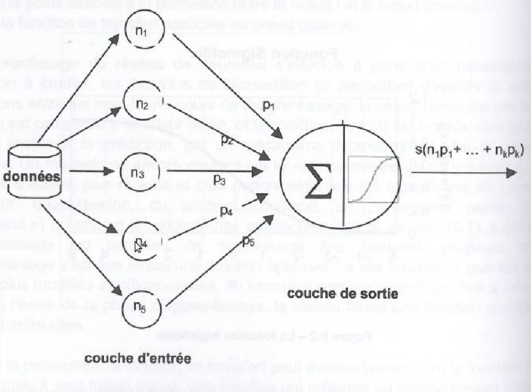
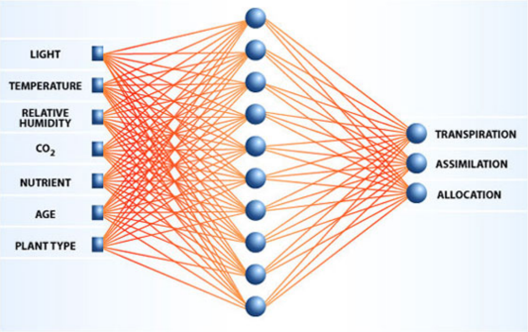


Figure 8.3 – Réseau de neurones sans couche cachée

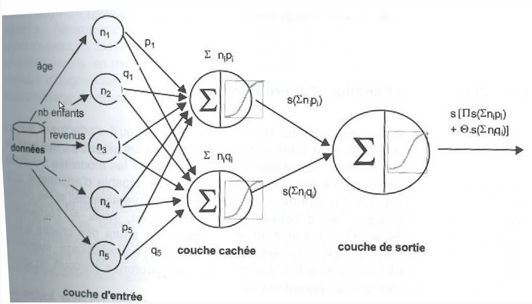
Src: [Tufféry, Data Mining et Informatique Dcisionnelle]

# One-layer network




Src: `http://www.makhfi.com`

# Two-layer network




Src: [Tufféry, Data Mining et Informatique Dcisionnelle]

# Profound ideas and tricks

- Convolutional networks
- Max-pooling
- Dropout
- Data augmentation
- GANs
- Representation learning
- Self-learning (ex: classify against rotations)

# The three main theoretical challenges of deep learning

- **Expressive power of DNN**: why are the function we are interested in so well approximated by (deep convolutive) neural networks?
- **Success of nave optimisation**: why does gradient descent lead to a good local minimum?
- **Generalization miracle** why is there no overfitting with so many parameters?