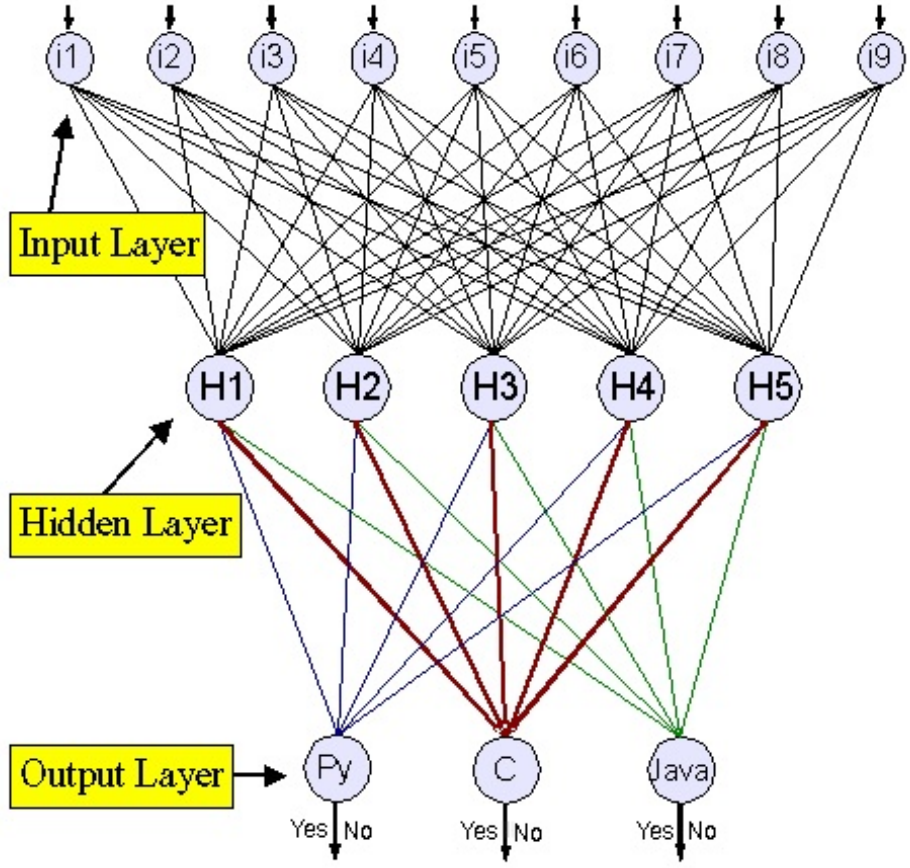
Pattern Recognition :
The Flagship Problem in Artificial Intelligence



Pattern Recognition - The Prototypical Task

- Numerous applications :
  - Handwritten character recognition
  - Medical diagnosis support tools
  - Credit-scoring, churn analysis
  - Speech recognition
  - *etc.*
- Numerous algorithms "off the shelf" (scikit-learn) :
  - Linear Discriminant Analysis, naive Bayes
  - Decision trees, nearest-neighbours
  - (Deep) Neural Networks
  - Support Vector Machines
  - Boosting
  - Random Forests
  - *etc.*
- Concepts and methods will be extended to tackle more complex problems, e.g. biometrics, recommending systems, unsupervised anomaly detection

The generic supervised setup

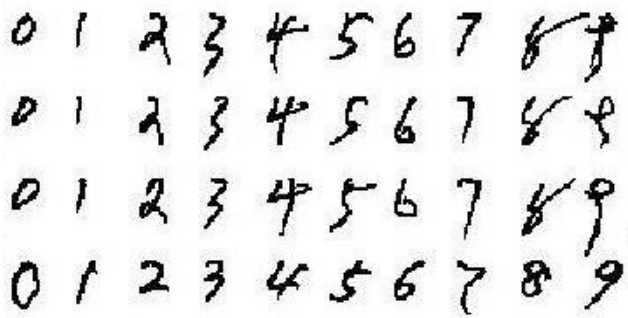
- System is described by a random pair $(X, Y) \sim P$ unknown probability distribution
- $X =$ **input** random vector valued in $\mathbb{R}^d$, in general $d \gg 1$
- $Y =$ label/**output**, encoded and valued in $Y \in \mathbb{R}$
- **The hypothesis :** X models useful information for guessing Y Predictive rule : $g : x \in \mathbb{R}^d \mapsto g(x)$ in some class of programmable rules - Hopefully, $g(X) \sim Y$ for any new pair (X, Y)
- The accuracy of the guess/decision/prediction is evaluated by a **loss function** : $\ell : Y \times Y \to \mathbb{R}^+$
- Risk (unknown !) = generalisation error

$$L(g) = \mathbb{E}\left(\ell(Y, g(X))\right)$$

minimum over $g \in \mathcal{G}$ ideally.

- Training data (labelled examples) $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ $\overset{i.i.d.}{\sim} P$, i.e. copies of the generic pair (X, Y)

First example : handwritten digit recognition



- The input information $X \in \mathbb{R}^d$ is an array describing a pixelated image (*e.g.* $d = 640 \times 480$)
- The label Y is a digit : $Y = \{0, 1, \ldots, 9\}$
- '$0-1$' loss : $\ell(y, g(x)) = \mathbb{I}\{y \neq g(x)\}$
- Database $\mathcal{B}$ with many *labeled examples* are available

$$(X_1, Y_1), \ldots, (X_n, Y_n), \quad n = 60\,000$$

e.g. www.nist.gov

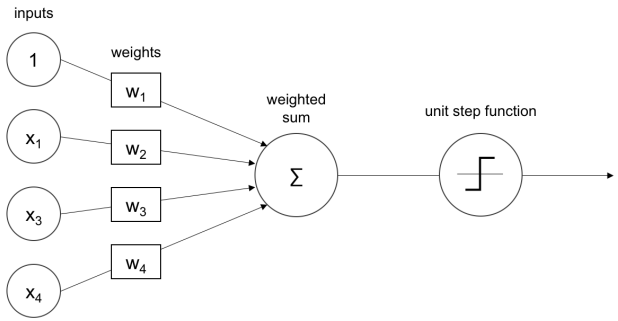
First go : binary labels

- $Y = \{-1, +1\}$ (pure convention, but useful)
- A simple task covering **many situations** :
  - presence/absence of a certain object in an image X
  - state of a system described by physical parameters X : normal *vs* abnormal,
  - medical diagnosis based on physiological parameters X : presence/absence of a given pathology
  - commercial targeting, *etc.*
- The predictive rule may take the form of the sign of a real valued function $f(x)$: $g(x) = sgn(f(x))$. The quantity $f(x)$ ideally indicates **confidence**
- Binary loss

$$\ell(y, g(x)) = \mathbb{I}\{y \neq g(x)\} = \mathbb{I}\{-y \cdot f(x) < 0\}$$

Weights can be introduced, the two types of error may have a **different impact**

The first AI model : artificial neuron



- A mathematical function introduced by McCulloch & Pitts '43
- Mimics the synaptic transmission of the information to a biological neuron
- Given the weight vector w, the function is implemented in a sumplistic fashion :
  1. Compute the inner product $\langle x, w \rangle = {}^t w \cdot x$ (multiplications and summations only)
  2. Activate the neuron if the inner product plus a parameter θ is positive, do not activate it otherwise

$$g(x) = sgn(\langle x, w \rangle + \theta)$$

The first AI model : artificial neuron

- We can collect and store labeled examples $(X_1, Y_1), \ldots, (X_n, Y_n)$
- We need an algorithm to choose θ and the synaptic weights w, so as to reproduce best the examples !
- The first learning algorithm : the Monolayer Perceptron

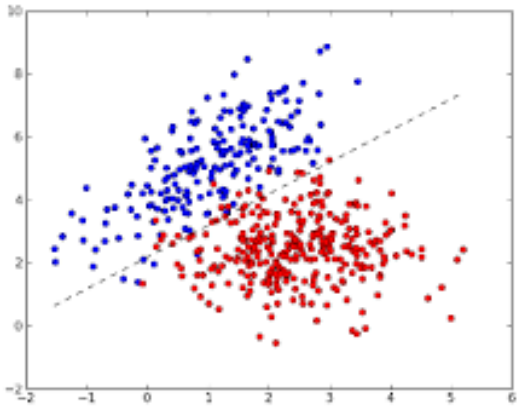
Learning an artificial neuron model
Frank Rosenblatt (1957)

- A probabilistic/statistical view of AI problems
- Motivation : computer vision for aeronautics
- Exploit 'recent' advances in optimization (stochastic approximation for gradient descent, Robbins & Monro '51)
- Exploit the capacities of 'high-speed' calculators in the 50's
- 'On-line' algorithm

The Algorithm - Geometric ideas

- Goal : learn how to split the input space $\mathbb{R}^d$ into two halves, separated by an **affine hyperplane** of eq. $\theta + {}^t w \cdot x = 0$

$$g(x) = sgn({}^t w \cdot X + \theta)$$



Assign positive label to any input x above the hyperplane and negative label when x is below it

- Ideally minimize over (w, θ) in $\mathbb{R} \times \mathbb{R}^d$, the **empirical error**

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{-Y_i({}^t w \cdot X_i + \theta) > 0\}$$

The Algorithm

- Observe that it is the statistical counterpart of the probability of error

$$L(g) = \mathbb{P}\{Y \neq g(X)\}$$

based on the training data, when $g(x) = \theta + {}^t w \cdot x$

- Main barrier to the optimization of the empirical error : $u \mapsto \mathbb{I}\{u > 0\}$ is not differentiable !

- Surrogate problem : minimize

$$-\sum_i Y_i({}^t w \cdot X_i + \theta)$$

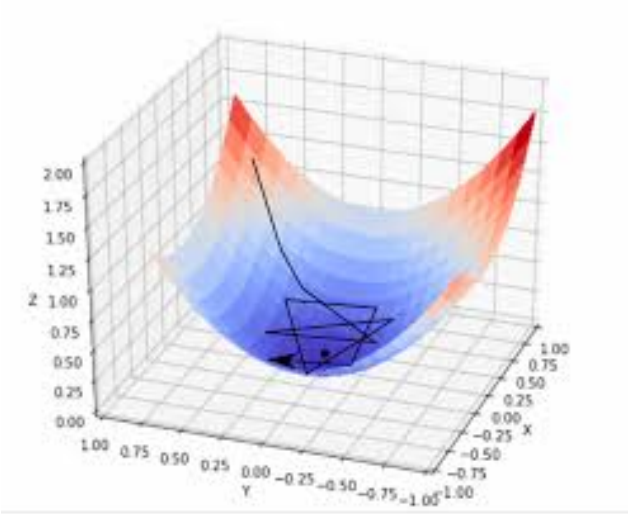
by stochastic gradient descent : $(w, \theta) \mapsto -Y({}^t w \cdot X + \theta)$ is differentiable with gradient (YX, Y)

The Algorithm

- Start with an initial guess for (w, θ)
  1. Choose **at random** a point (X_i, Y_i) misclassified by the current rule (if there is any)
  2. Change the rule parameters (w, θ) by a step into the opposite direction of the gradient computed at (X_i, Y_i) with rate/stepsize ρ

$$\begin{pmatrix} w \\ \theta \end{pmatrix} \leftarrow \begin{pmatrix} w \\ \theta \end{pmatrix} + \rho \begin{pmatrix} Y_i X_i \\ Y_i \end{pmatrix}$$

- Repeat until there is no misclassified observations anymore...



Once the rule is learned, how well does it work ?

- If (w, θ) has been learned by means of the training dataset $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, DO NOT rely on the **training error** to evaluate its future performance !

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{-Y_i(\theta + {}^t w \cdot X_i) > 0\}$$

- The parameters have been precisely chosen to mimic the input-output pairs (X_i, Y_i), the training error is too optimistic to estimate the **true/theoretical** risk

$$\mathbb{P}\{-Y(\theta + {}^t w \cdot X) > 0\}$$

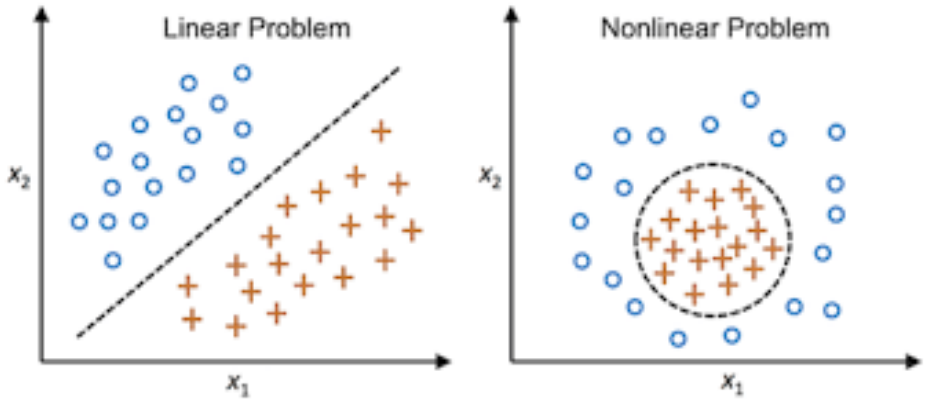
- If possible, use another dataset $\{(X'_1, Y'_1), \ldots, (X'_{n'}, Y'_{n'})\}$, independent from D_n (**test data**) to compute the **test error**

$$\frac{1}{n'} \sum_{i=1}^{n'} \mathbb{I}\{-Y'_i(\theta + {}^t w \cdot X'_i) > 0\},$$

which is a fair (unbiased) estimate of $\mathbb{P}\{-Y(\theta + {}^t w \cdot X) > 0\}$

Only the beginning of the story...

- Easy (and possibly on-line) implementation, that requires no sophisticated library
- If the data are linearly separable, it converges in a finite number of steps
- If not, the hyperplane will oscillate forever...



Only the beginning of the story...
More algorithms are needed !

- Linear Discriminant Analysis, naive Nayes
- Majority vote (local averages) :
  nearest neighbours and decision trees
- Linear SVM
- Neural Networks
- Towards more accuracy and/or stability :
  ensemble learning, deep NN, nonlinear SVM

Some mathematical/statistical concepts are required to understand them...

Basics in Probability - The Frequentist Approach

- The simplest probabilistic model : flip a coin...
- The Bernoulli distribution $\mathcal{B}(\theta)$: only two possible outcomes for the r.v. Y, 0 or 1 say
- The distribution of the binary r.v. Y is fully described by the parameter

$$\theta = \mathbb{P}\{Y = +1\} = 1 - \mathbb{P}\{Y = 0\}$$

- The expectation/mean of a r.v. $Y \sim \mathcal{B}(\theta)$ is

$$\mathbb{E}[Y] = 1 \times \theta + 0 \times (1 - \theta) = \theta$$

- Its variability is described by the variance

$$Var(Y) = \mathbb{E}[(Y - \mathbb{E}[Y])^2] = \theta(1 - \theta)$$

Basics in Probability - The Frequentist Approach

- How to simulate a Bernoulli distribution using a pseudo random number generator (PRNG) ?
- The PRNG chooses at 'random' a number U between 0 and 1 : set $Y = +1$ if $U \leq \theta$ and $Y = 0$ otherwise
- Flip a coin $n \geq 1$ times or, preferably, use the PRNG to produce 'independent' outcomes $Y_1, \ldots, Y_n$
- Test it ! If n is large, the frequency of ones

$$\bar{Y}_n = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

is 'generally' close to $\theta = \mathbb{P}\{Y = +1\}$

Basics in Probability - The Frequentist Approach

- **Law of Large Numbers** : as n tends to ∞,

$$\bar{Y}_n \to \mathbb{E}[Y] = \theta$$

with probability one

- In other words, the empirical mean gets asymptotically closer to the true mean
- **Central Limit Theorem** : convergence occurs at the rate $1/\sqrt{n}$ and the (random) fluctuations of the empirical mean around its limit is described by a Gaussian distribution : as n tends to ∞,

$$\sqrt{n}(\bar{Y}_n - \mathbb{E}[Y]) \Rightarrow \mathcal{N}(0, \theta(1-\theta))$$