SI221 – Neural Networks

Chloé Clavel,
chloe.clavel@telecom-paristech.fr,
Sources : Jean-Marie Nicolas, Alain Grumbach

Telecom ParisTech, France

Outline of the course

Objectives of the Lecture

At the end of the lecture...

you will master :

- the underlying principles of the linear perceptron
- the training steps : gradient descent

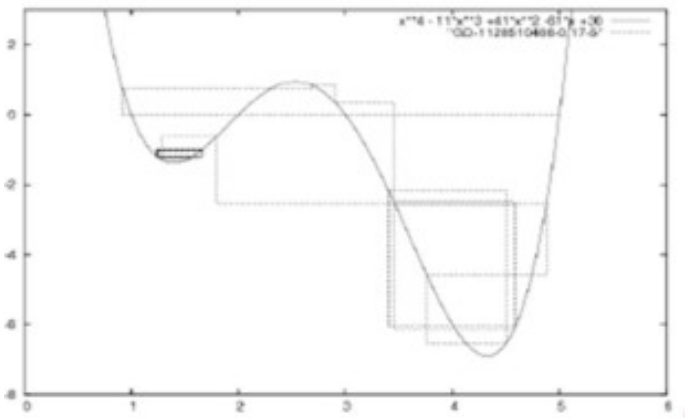
you will be ready to follow lectures :

- for the next steps : involved neural architecture proposed in deep learning

Reminder/prerequisites

Gradient descent

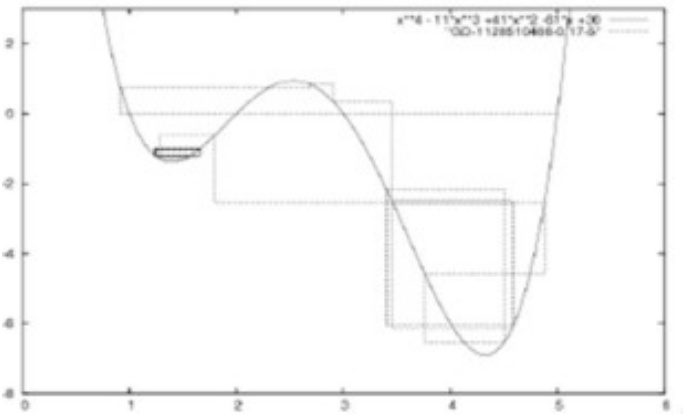
- Method for the minimization of a differentiable function $f : \mathbb{R} \to \mathbb{R}$, when
  - we know the analytical expression of f ;
  - it is not easy to solve the equation $f'(x) = 0$
- Build a numerical sequence x_n so that $\lim_{n \to +\infty} f(x_n) = \min_{x \in \mathbb{R}} f$



Reminder/prerequisites

Gradient descent

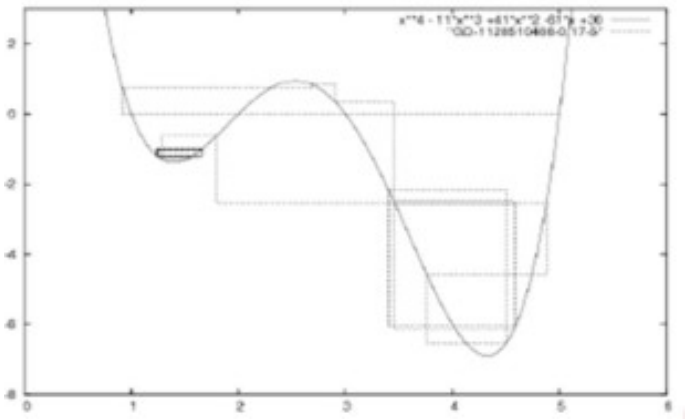
- Choose a x_0 (empirical choice)
- Iterative process while $|x_{n+1} - x_n| \geq \epsilon$, $x_{n+1} = x_n + \Delta x_n$
- $\Delta x_n = -\eta f'(x_n)$ depends on :
  - the derivative function $f'(x_n)$ – direction of the slope of the curve...
  - ...during a distance corresponding to the step size η



Reminder/prerequisites

Gradient descent

- Empirical choice of η :
  - if η is too big, we will not reach the absolute minimum
  - if η is too small, we will stay in a secondary minimum



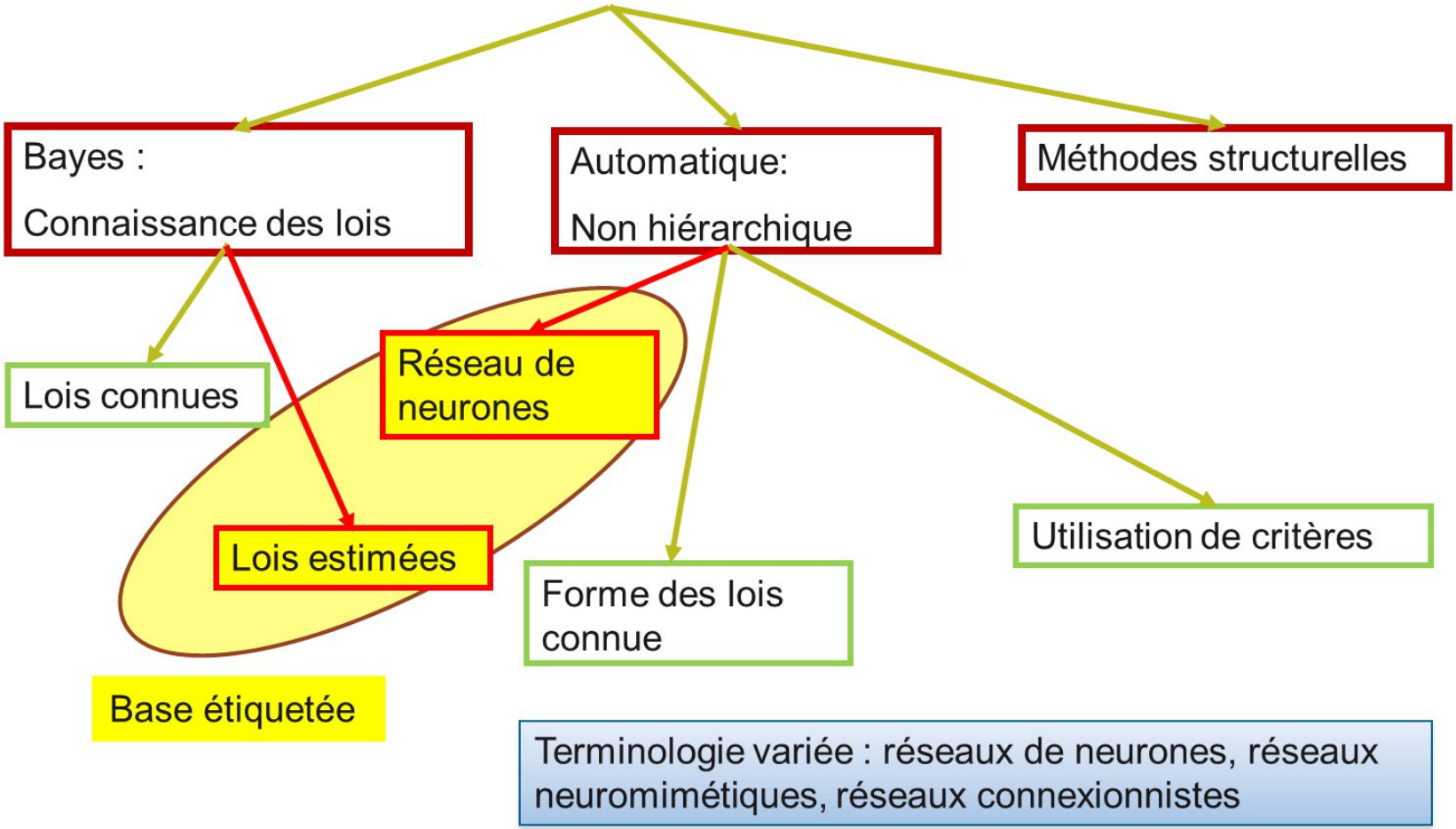
Reminder/prerequisites

Gradient descent for multivariable functions

- Master partial derivatives of functions

$$\frac{\partial J}{\partial x_j}$$

Neural Networks and classification methods



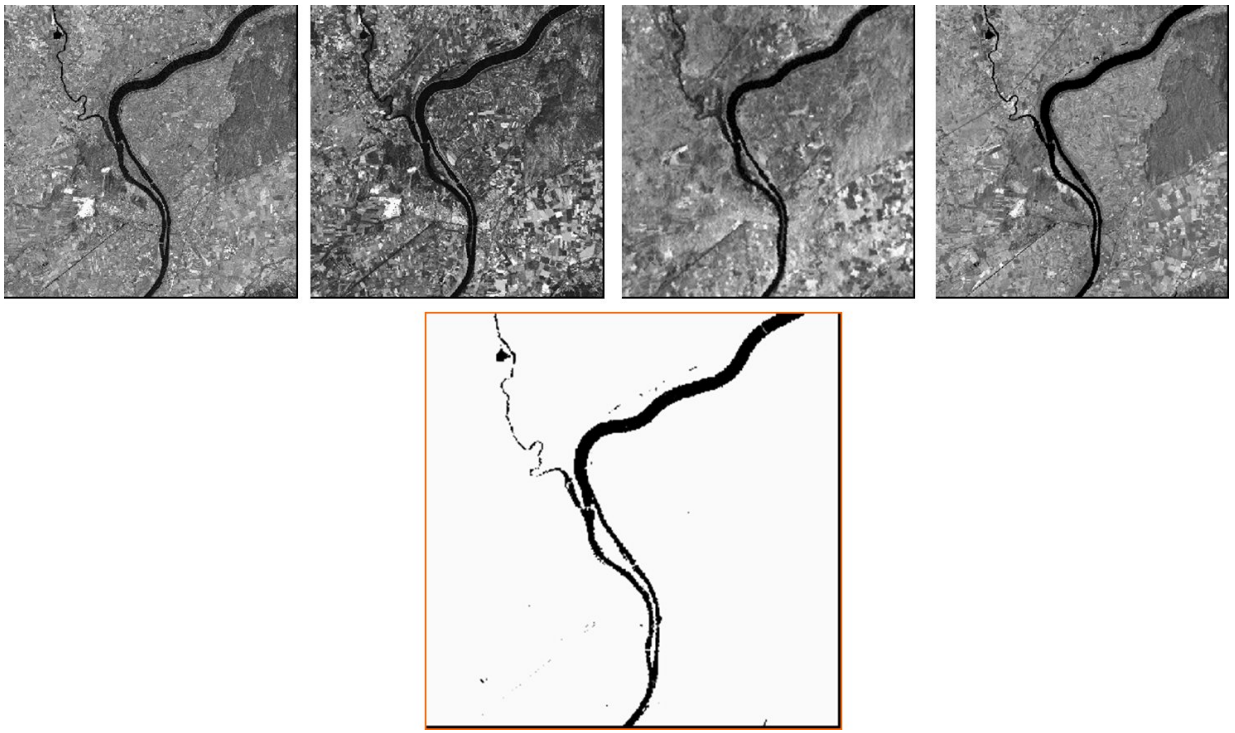
Neural Networks

C'est une méthode de classification supervisée :

- Apprentissage sur la base d'apprentissage (exemples labellisés) : trouver la bonne architecture du système de classification
- Développement (ou généralisation) :juger des performances de l'architecture sur des individus n'ayant pas servi à définir l'architecture

Example of classification problems

Neural Network lab



Example of classification problems

Recognition of digits

Reconnaissance de caractères

Y. Le Cun et al.

AT & T Bell Labs

1990

Le problème

Données: 9298 chiffres extraits de codes postaux manuscrits

3249 chiffres en caractères d'imprimerie (35 polices)

Base d'exemples : 7291 chiffres manuscrits

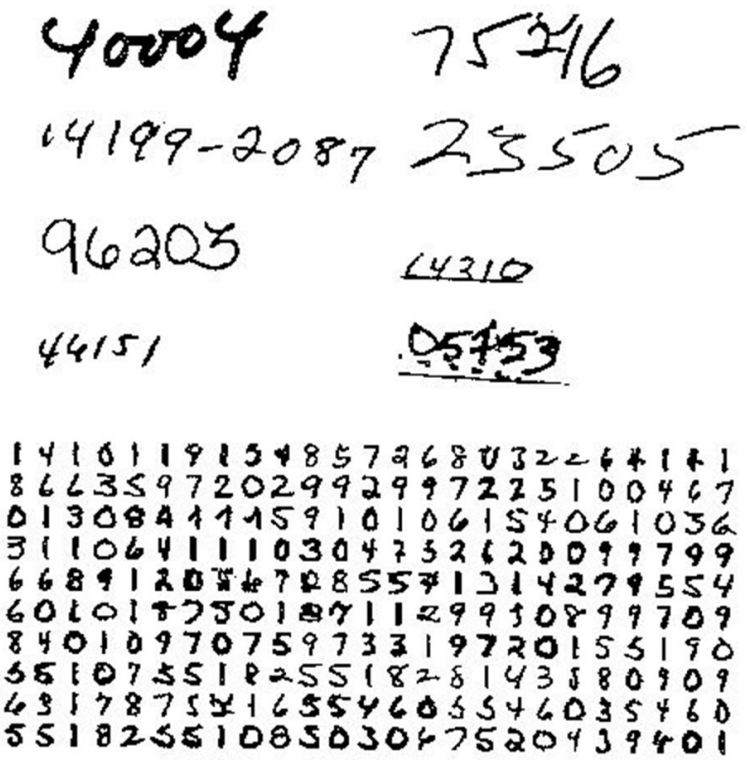
2549 chiffres en caractères d'imprimerie

Base de test : 2007 chiffres manuscrits

700 chiffres en caractères d'imprimerie

(les deux bases contiennent des exemples ambigus (naturels))

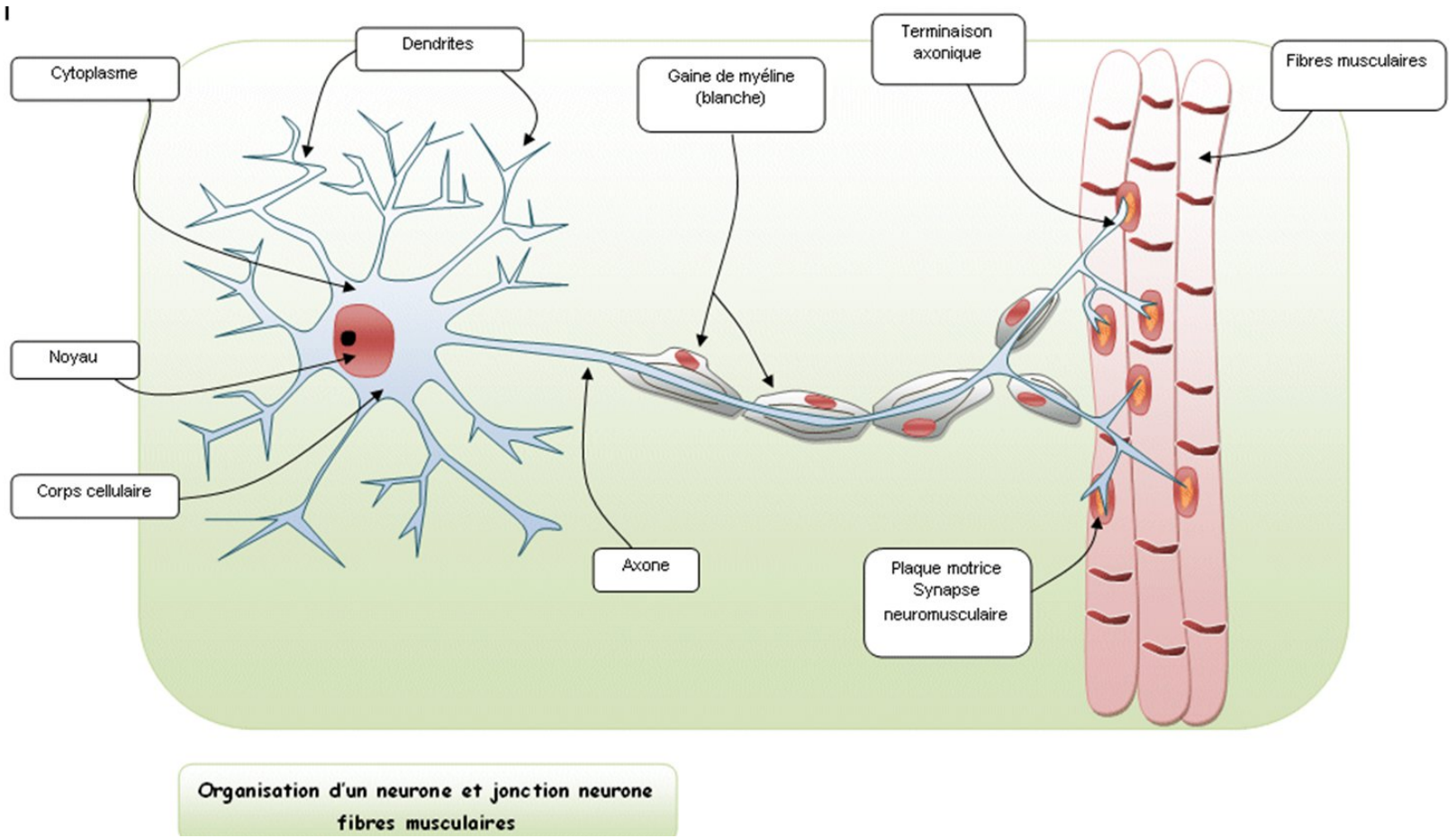
Example of classification problems



Neural network : why neural ?

The idea of neural networks comes from an attempt to model the biological neurons and their connection to simulate intelligence (1943 the neurophysiologist Mc Culloch and the mathematician Pitts)

Biological neurons



Organisation d'un neurone et jonction neurone fibres musculaires

Biological neurons

- cerveau humain : entre un et cent milliards de neurones
- chaque neurone est connecté en moyenne à 10.000 autres par le biais de synapses.
- .neurone activé par stimulations synaptiques
- Le neurone transmet alors un signal électrique aux neurones suivants

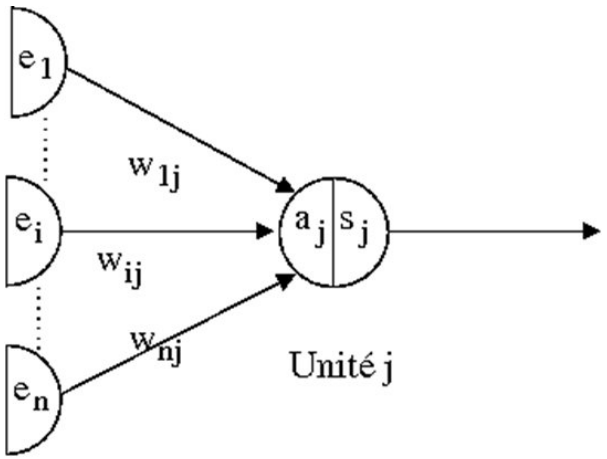
From biological neuron to artificial neurons

Neurone artificiel

Un neurone a pour sortie la somme pondérée des entrées des neurones voisins (McCulloch et Pitts, 1943)

Formal neuron

- Inputs : $e_i \in \{0, 1\}$ (1 if active)
- Activity of the neuron j : $a_j = \sum_{i=1}^{N} w_{i,j} e_i \iff A = W^t * E$ according to the synaptic weights $w_{i,j}$
- Output : $s_j = f(a_j)$, with f the activation function

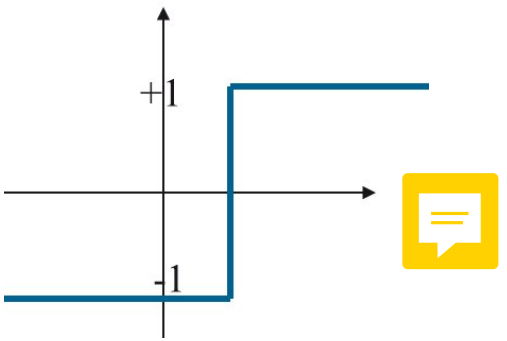


Threshold activation function

Activation function f calculates the output of the neural unit based on its inputs $s_j = f(a_j)$, where $a_j = \sum_{i=1}^{N} w_{i,j} e_i$

Threshold function

- En fonction du signe de $a_j - \sigma$ où σ est le seuil :
  - Cas 1 $s_j = 0$ ou 1
  - Cas 2 $s_j = -1$ ou 1 (cf charge $+e$ ou $-e$: Charges $+$ ou $-$ dans les neurones biologiques)
- Remarque : non dérivable



Binary classification

Principe

- A partir de la base de données étiquetées en deux classes C^+ et C^-
- Apprendre les poids synaptiques w pour que tous les exemples soient bien classés :
  - $\forall E \in C^+$, $f(E) = 1$ et
  - $\forall E \in C^-$, $f(E) = 0$

Binary classification : the linear case

Ecriture Matricielle

$$a_j = \sum_{i=1}^{N} w_{i,j} e_i - \sigma = W^t E - \sigma$$

Trouver l'hyperplan séparateur d'équation $W^t E - \sigma = 0$

Binary classification : the linear case

Principe de Hebb

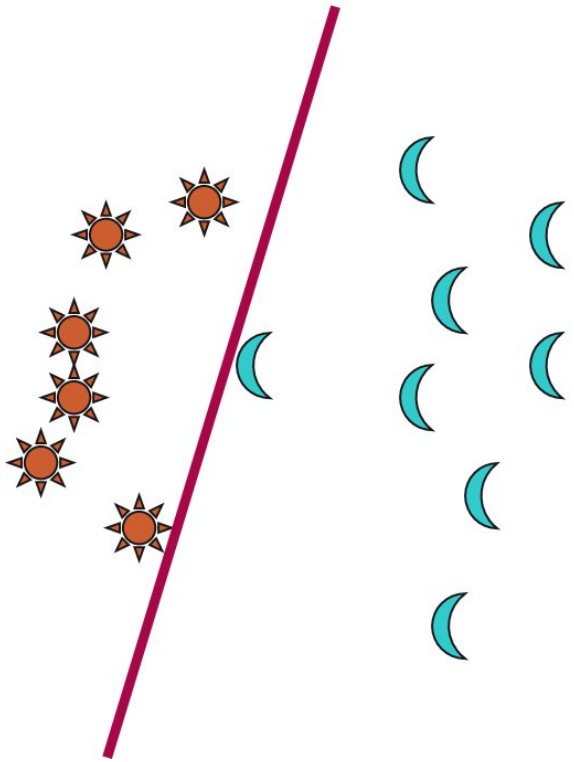
Principe sur lequel repose les algorithmes d'apprentissage des réseaux de neurones :

- neural pathways strengthen over each successive use, especially between neurons that tend to fire (s'exciter) at the same time

Apprentissage

Apprendre le seuil de la fonction f (également appelé **biais**) et les poids wj qui permettent de séparer les exemples d'apprentissage en 2 classes revient à trouver l'équation de la séparatrice entre les deux classes

Binary classification



- **Equation de l'hyperplan**

$$W^t x + b = 0$$

- **Si X appartient à la classe C+**

$$W^t X + b > 0$$

- **Si X appartient à la classe C-**

$$W^t X + b < 0$$

Binary classification

$$b = -\sigma$$

$$W^t x + b = 0$$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \rightarrow \vec{\tilde{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} w_1 & w_2 & \dots & w_n & -\sigma \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix} = 0$$

$$\tilde{W}^t \tilde{X} = 0$$

Règle du Perceptron (Rosenblatt, 1957) :

Pour résoudre l'équation précédente

Essayer différentes séparatrices, en faisant tourner la droite passant par l'origine dans un sens ou dans un autre, jusqu'à trouver la droite qui permet d'avoir les exemples d'apprentissage dans les bonnes classes.

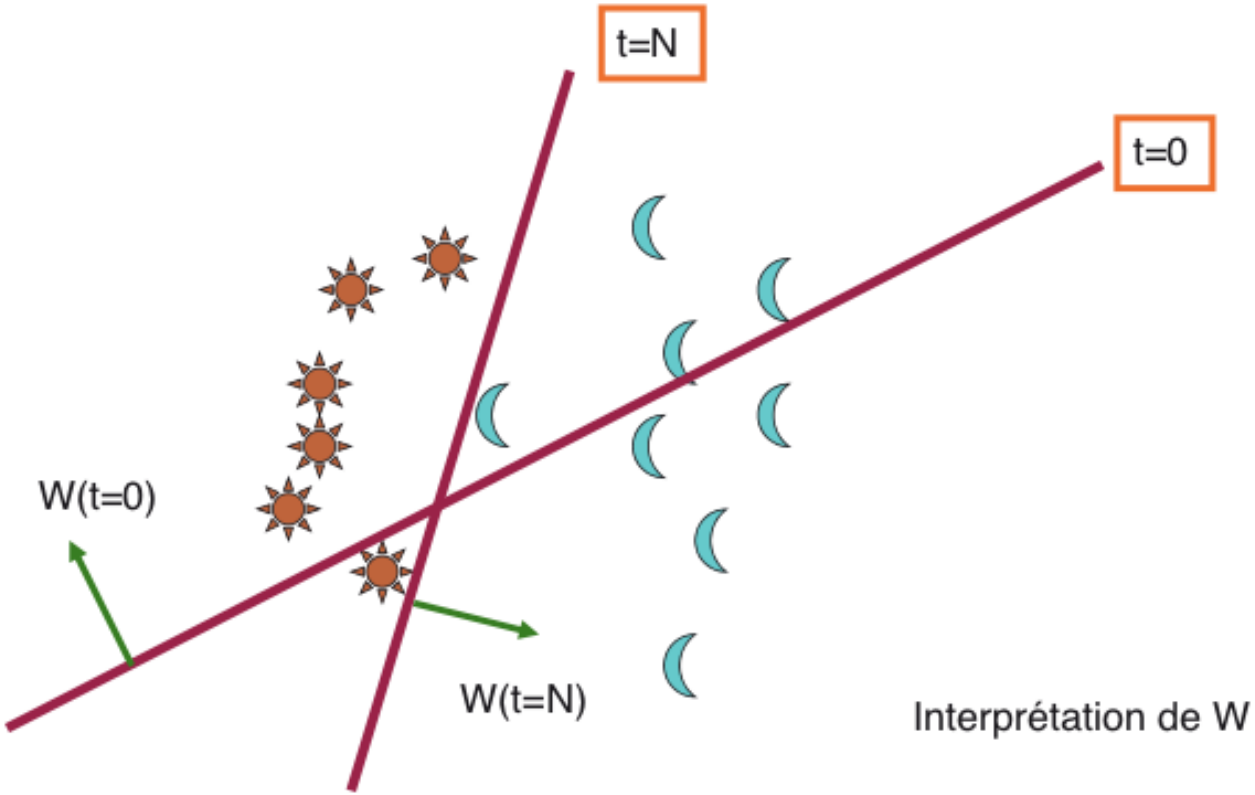
Règle du Perceptron (Rosenblatt, 1957) :

Parcourir la base d'exemples

- Initialiser les poids $\tilde{W}$ [a] des neurones
- Modifier itérativement les poids si la sortie s_j n'est pas égale à la sortie désirée d_j : $\tilde{W}(t+1) = \tilde{W}(t) - \eta(s_j - d_j)\tilde{X}$
- Le paramètre η, théoriquement égal à 1 pour la règle du perceptron, permet d'"adoucir" le processus d'apprentissage comme tout paramètre utilisé par exemple dans une descente de gradient.

a. On peut dérouler l'algo également avec W et x

Binary classification



Linear case

Tests d'arrêt pour l'apprentissage

- Définition d'une **époque** (= 1 parcours de la totalité des exemples disponibles pour l'apprentissage)
- dans le cas linéairement séparable on s'arrête quand l'ensemble des exemples sont tous bien classés

Exo

Soit un perceptron monocouche ayant comme fonction de transition f, la fonction seuil suivante : si $a > seuil$, $f(a) = 1$, sinon $f(a) = -1$. Nous fixons ici le seuil à 0.2. e1 et e2 sont les entrées du réseaux de neurones et d la sortie correspondante, telle qu'étiquetée dans l'ensemble d'apprentissage. Soit la base composée de 4 exemples d'apprentissage :

	e1	e2	d
(1)	1	1	1
(2)	-1	1	-1
(3)	-1	-1	-1
(4)	1	-1	-1

1. En appliquant l'algorithme d'apprentissage du perceptron avec un pas de modification des poids η égal à 0.1, déterminez les poids du perceptron. Vous initialiserez l'algorithme avec les valeurs suivantes , $w1 = -0.2$, $w2 = 0.1$.
2. Représentez les données d'apprentissage dans un espace à 2 dimensions et tracez la droite séparatrice des 2 classes. Donnez son équation.

Problem statement - Non linearity

- Limitations of Rosenblatt's perceptron : can't solve non linearly separable problems
- How to design a perceptron in case of non-linearity ?

Introduce a loss

Erreur-Loss-Fonction de coût

Soit un exemple de la base de données d'apprentissage : X_p, l'erreur locale est égale à :

$$\frac{1}{2}(s_p - d_p)^2$$

L'erreur globale est la moyenne des erreurs locales sur l'ensemble des R exemples d'apprentissage (erreur quadratique moyenne- MSE - Mean Square Error) [1]

$$J = \sum_{p=1}^{R} (s_p - d_p)^2 \qquad (1)$$

1. On omet ici le terme de division pour alléger l'expression $J = \frac{1}{R}\sum_{p=1}^{R}(s_p - d_p)^2$

Erreur-Loss-Fonction de coût

Objectif : trouver les w qui minimise l'erreur

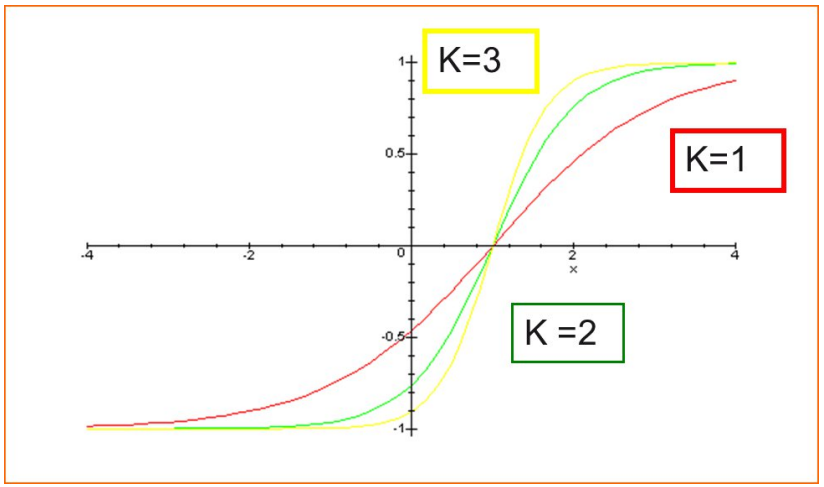
$$J = \sum_{p=1}^{R} (s_p - d_p)^2 \quad = \quad \sum_{p=1}^{R} \left(f \left(\sum_{i=1}^{N} w_i \, e_{i,p} \right) - d_p \right)^2 \qquad (2)$$

Utiliser l'algorithme de descente du gradient avec une fonction f qui soit dérivable (donc pas la fonction seuil)
Other examples of cost functions : 0-1 error, cross-entropy, mean absolute error

Example of differentiable activation function

Sigmoid functions

- $f_{K,\sigma}(x) = \frac{e^{K(x-\sigma)}-1}{e^{K(x-\sigma)}+1}$
- $f(x) \in]-1;1[$
- K : slope on $y = 0$
- differentiable function
- $K \to \infty$: threshold function



Other examples of activiation functions : tanh (hyperbolic tangente function), ReLU (rectified Linear Unit), etc.

Optimization principle for the training

- Find the w_i which minimize the learning error J on the training database

$$\frac{\partial J}{\partial w_i} = 0 \quad \forall i \in [1, N]$$

- Use Stochastic gradient descent for the minimization

$$w_i \rightarrow w_i' = w_i - \eta \nabla_i(J) \tag{3}$$

avec $\nabla_i(J) = \frac{\partial J}{\partial w_i}$

Gradient

$$\begin{aligned}
\frac{\partial J}{\partial w_i} &= 2\sum_{p=1}^{R} \frac{\partial f\left(\sum_{i'=1}^{N} w_{i'}\ e_{i',p}\right)}{\partial w_i} \left(f\left(\sum_{i'=1}^{N} w_{i'}\ e_{i',p}\right) - d_p\right) \\
&= 2\sum_{p=1}^{R} \frac{\partial f\left(\sum_{i'=1}^{N} w_{i'}\ e_{i',p}\right)}{\partial w_i} (s_p - d_p) \\
&= 2\sum_{p=1}^{R} e_{i,p}\ (s_p - d_p)\ \left.\frac{\partial f}{\partial x}\right|_{x=\sum_{i'=1}^{N} w_{i'}\ e_{i',p}} \qquad (4)
\end{aligned}$$

Modalités opératoires pour l'apprentissage

2 possibilités :

- 1) Apprentissage global :
  - on calcule l'erreur quadratique moyenne sur l'ensemble d'apprentissage, on calcule le gradient
  - on corrige les poids avec l'algorithme de descente du gradient
  - on parcourt à nouveau l'ensemble d'apprentissage (inconvénient : débouche sur des minima locaux)

Modalités opératoires pour l'apprentissage

2 possibilités :

- 2) Apprentissage échantillon par échantillon :
  - On choisit un échantillon p dans la base d'apprentissage On évalue son erreur locale et on calcule le gradient
  - on corrige les poids avec l'algorithme de descente du gradient
  - on passe à l'exemple suivant

Ce qu'on a vu jusque maintenant

Case where the dimension of the outputs $d_{out} = 1$ which means that the network's output is a scalar.
Such networks can be used

- for regression (or scoring) by considering the value of the output
- for binary classification by consulting the sign of the output.

Perceptron moncouche avec plusieurs sorties

Networks with $d_{out} = k > 1$ for k-class classification, by

1. associating each dimension with a class
2. looking for the dimension with maximal value.

Perceptron moncouche avec plusieurs sorties $d_{out} = k > 1$

- If the output vector entries are positive and sum to one, the output can be interpreted as a distribution over class assignments
- such output normalization is typically achieved by applying a softmax transformation on the output layer.

Perceptron moncouche avec plusieurs sorties $d_{out} = k > 1$

Outputs normalization via **Softmax function** or normalized exponential function (a generalization of the logistic function for a K-dimensional vector $\mathbf{z}$)

$$softmax : \mathbb{R}^K \rightarrow \left\{ z \in \mathbb{R}^K | z_i > 0, \sum_{i=1}^{K} z_i = 1 \right\}$$

$$softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K$$

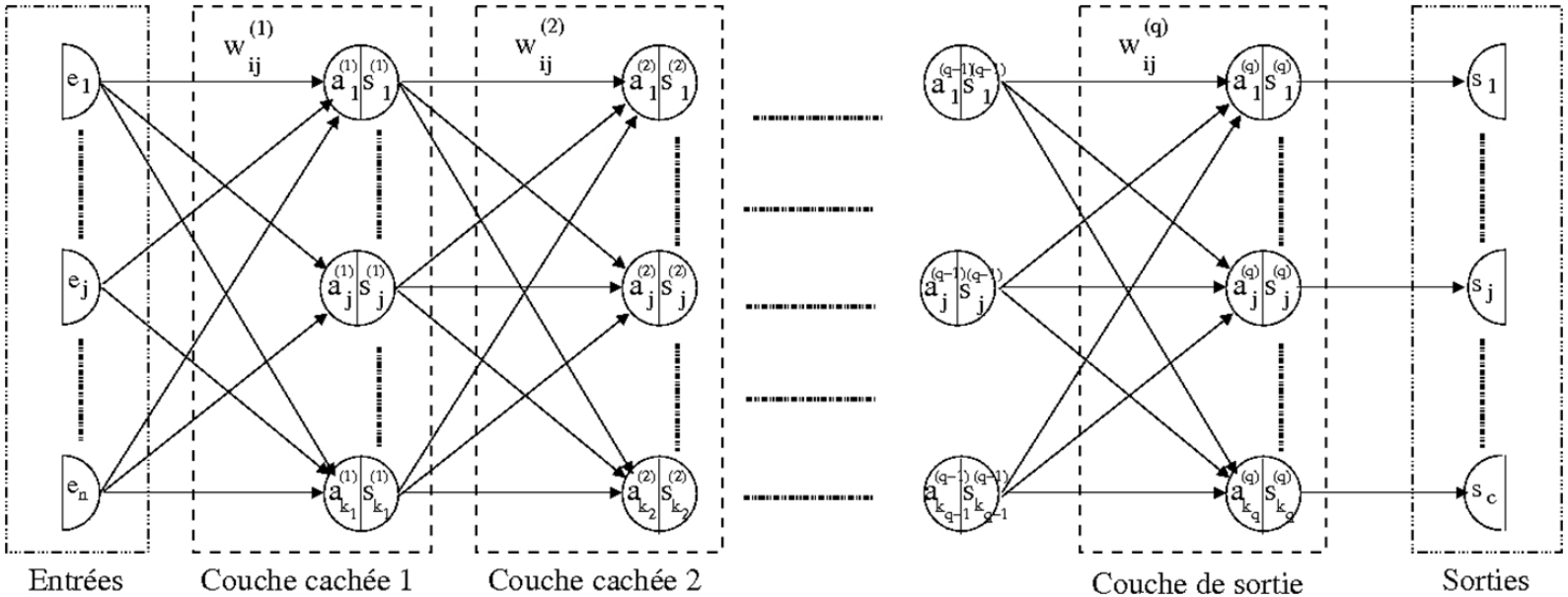
Note : Softmax is invariant to constant offsets in the outputs

$$softmax(z) = softmax(z + c)$$

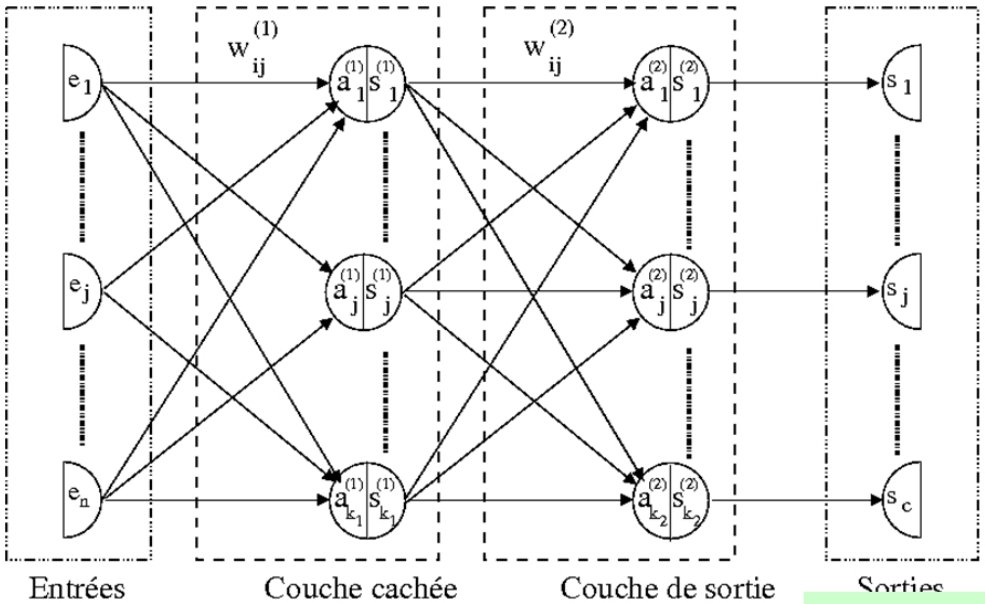
Multilayer NN

Objectif : traiter le cas non linéairement séparable

Principe : regrouper les neurones en couches



Two-layer NN



n

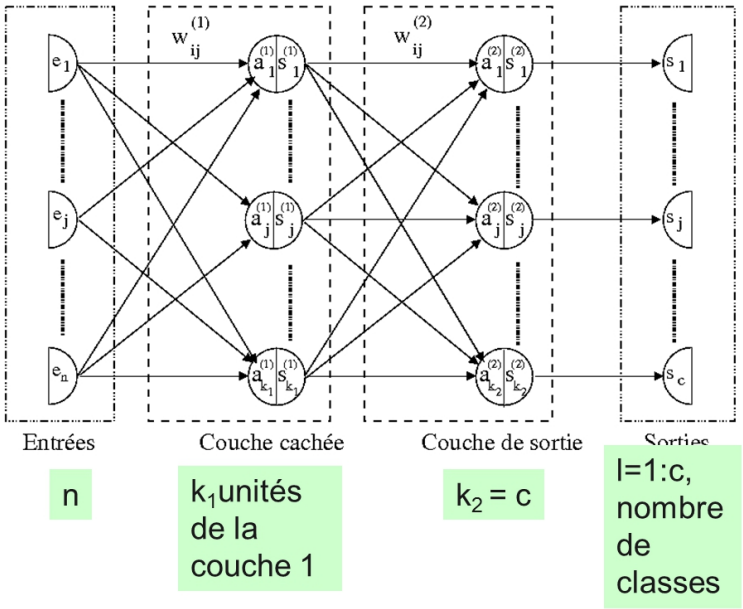
k_1 unités de la couche 1

$k_2 = c$

l=1:c, nombre de classes

ML NN Layers

Notion de « couche cachée » : entre l'entrée et la sortie, couches dont les sorties ne sont pas connues



FORWARD : we need to compute all the outputs of the $m-1$ layer to compute the outputs of the m layer.

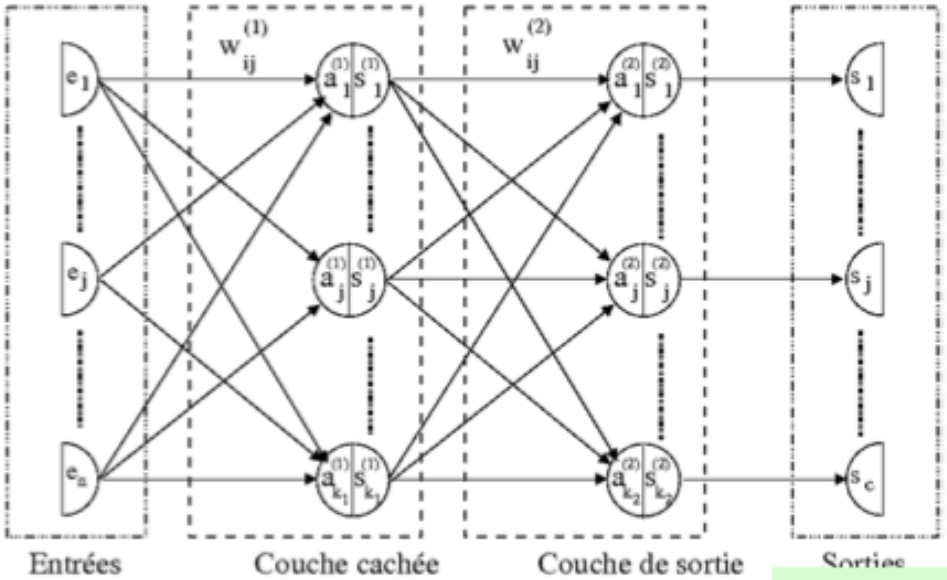
Training and backpropagation algorithm

Principe : L'erreur se propage de droite à gauche dans le réseau

1. define the loss
2. compute partial derivatives
3. apply gradient descent algorithm from output layers to input layers

Training and backpropagation algorithm : 2 layers

$$s_l = ?$$
$$a_l^{(2)} = ?$$
$$s_j^{(1)} = ?$$
$$a_j^{(1)} = ?$$



n

k_1 unités de la couche 1

$k_2 = c$

l=1:c, nombre de classes

Même fonction d'activation pour tous les neurones de toutes les couches

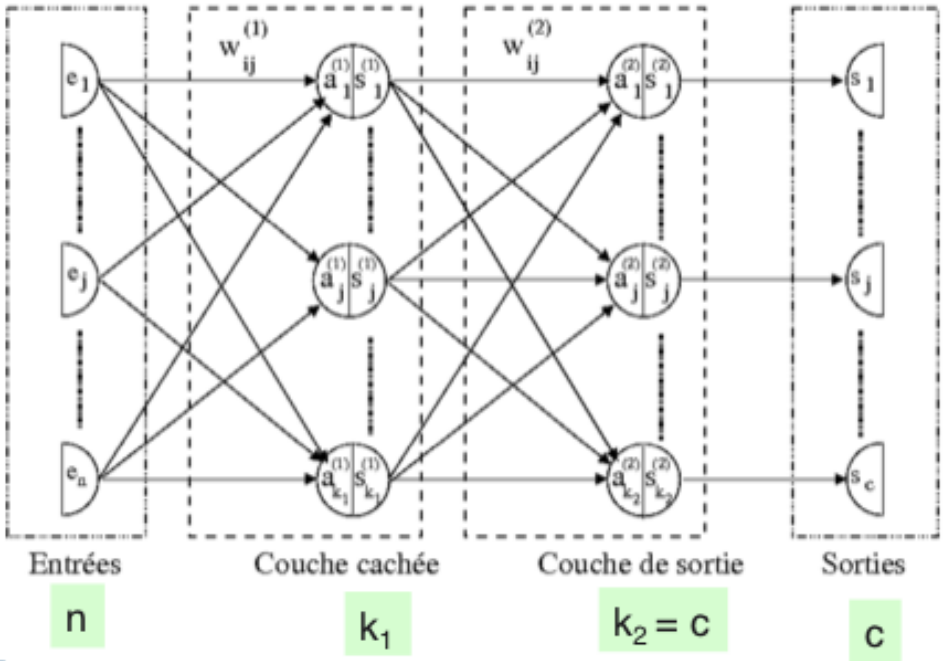
Training and backpropagation algorithm : 2 layers

$$s_l = f\left(a_l^{(2)}\right)$$

$$a_l^{(2)} = \sum_{j=1}^{k_1} w_{jl}^{(2)} s_j^{(1)}$$

$$s_j^{(1)} = f\left(a_j^{(1)}\right)$$

$$a_j^{(1)} = \sum_{k=1}^{n} w_{kj}^{(1)} e_k$$



Même fonction d'activation pour tous les neurones de toutes les couches

Pour la dernière couche

- Dériver en fonction du poids $w_{jl}^{(2)}$

$$\frac{\partial E_i}{\partial w_{jl}^{(2)}} = 2\left(s_l - d_l\right) f'\left(a_l^{(2)}\right) s_j^{(1)}$$

l =1:c sorties (nombre de classes)
j=1:k1 unités de la 1e couche
$w_{jl}^{(2)}$ les poids de la 2e couche

- Modification des poids $w_{jl}^{(2)}$ prenant en compte la direction du gradient pour minimiser l'erreur quadratique
- Calcul de l'erreur quadratique avec ces nouveaux poids et dériver en fonction de $w_{ij}^{(1)}$
- Modification des poids $w_{ij}^{(1)}$ prenant en compte la direction du gradient pour minimiser l'erreur quadratique

Pour la couche cachée

$$s_l = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(a_j^{(1)}\right)\right) = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(\sum_{k=1}^{n} w_{kj}^{(1)} e_k\right)\right)$$

- Dérivation des fonctions composées
- Possible puisque la sigmoïde est dérivable !!

$$\frac{\partial E_i}{\partial w_{kj}^{(1)}} = 2e_k f'\left(a_j^{(1)}\right)\sum_{l=1}^{c}\left(w_{jl}^{(2)}\left(s_l - d_l\right) f'\left(a_l^{(2)}\right)\right)$$

- On rétropropage le gradient !!!

References for this lecture

Le polycopié

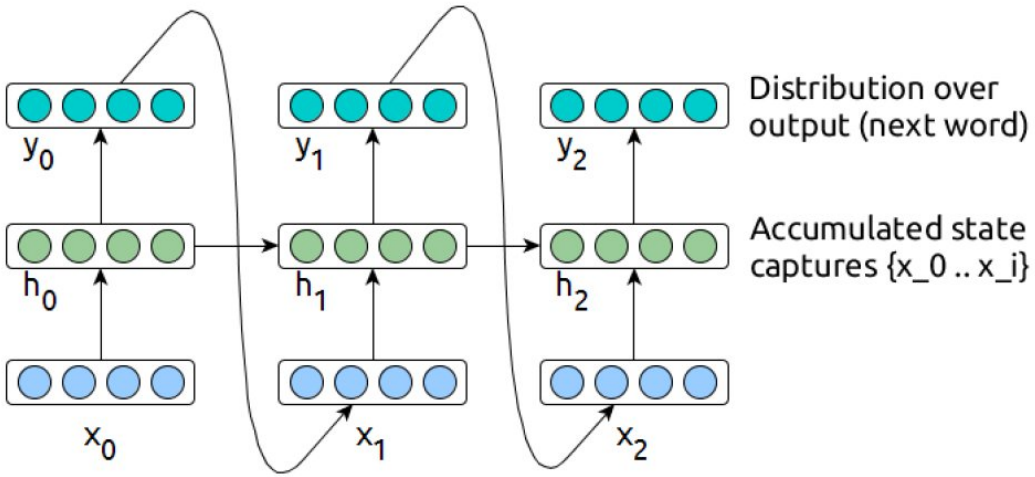
et `http://neuralnetworksanddeeplearning.com/index.html`

To go further

- Les réseaux récurrents (Hopfield, 1982)
  - Idéal pour la reconnaissance de séquences (reconnaissance de la parole , traitement du langage naturel)
  - La sortie à l'instant n-1 est utilisée pour calculer l'activation à l'instant n
- L'apprentissage par renforcement (Sutton et Barto, 1998)
  - Apprentissage dit "qualitatif" par pénalité/récompense.
  - Utilisé par exemple pour l'apprentissage de stratégies de dialogue human-machine

Recurrent Neural Networks

Use for language models



- Reads inputs xi to accumulate state hi and predict outputs yi
- Variants : LSTM networks (Long Short Term Memory Networks), RNN using gating mechanisms such as GRU (Gated Recurrent Units)