Les réseaux de neurones

Le perceptron

Maria Malek

29 novembre 2019

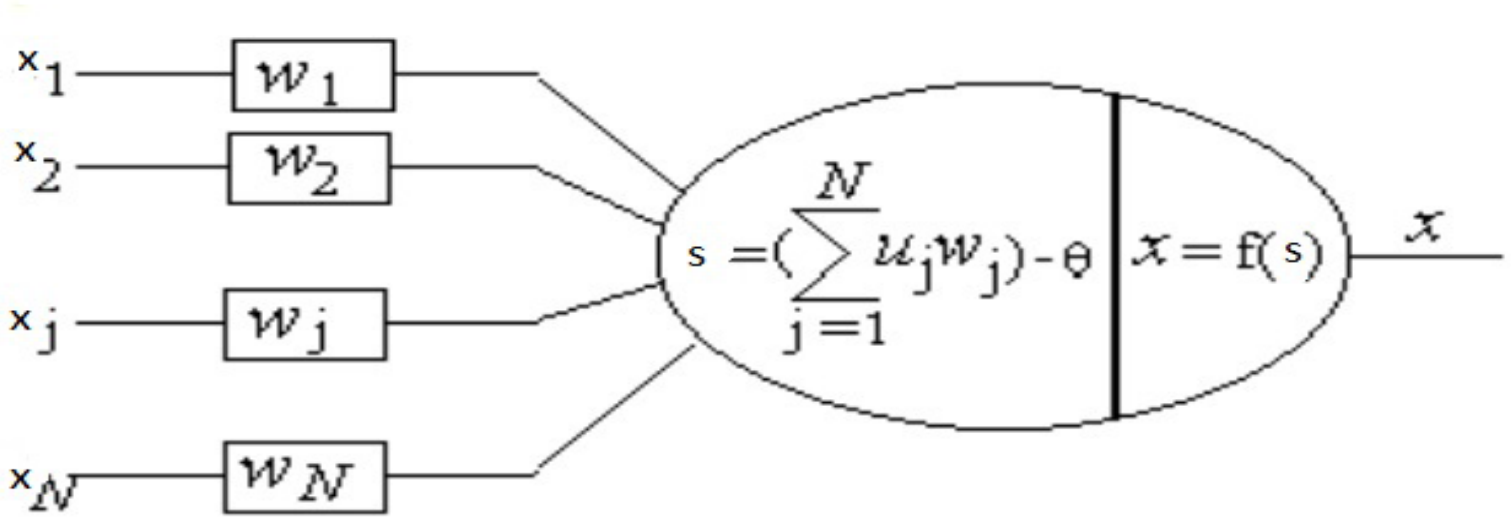
Réseaux de neurones

- Un modèle de réseaux de neurones est constitué d'un ensemble de neurones avec des liens entre eux.
- l'apprentissage se fait par adaptation des poids stockés dans les connections du réseau en fonction des exemples d'apprentissage.
- Certaines méthodes sont supervisées et d'autre sont non supervisées.
- Le réseau de neurone se stabilise (les poids des connections) après un certain nombre de passage de l'ensemble d'apprentissage.

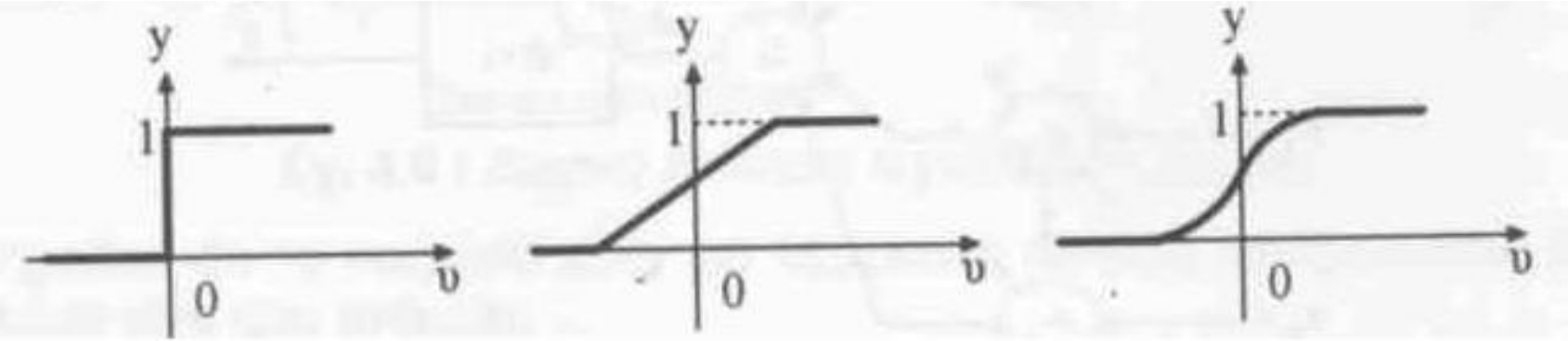
Le neurone de McCullochñPitts

- $x_1, ..x_n$: les entrées du neurone.
- $w_1, ..w_n$: les poids des connections entrantes.
- θ un seuil de comparaison.
- f est la fonction d'activation du neurone.

La sortie du neurone se calcule par : $S = f(\sum_{i}^{n} w_i x_i - \theta)$



Les fonctions d'activation binaires



Les fonctions d'activation binaires

$f(s) \in [0, 1]$

Fonction de seuil (fonction heaviside quand $\delta = 0$)

$$f(s) = \begin{cases} 1 & \text{si } s \geq \delta \\ 0 & \text{sinon.} \end{cases}$$

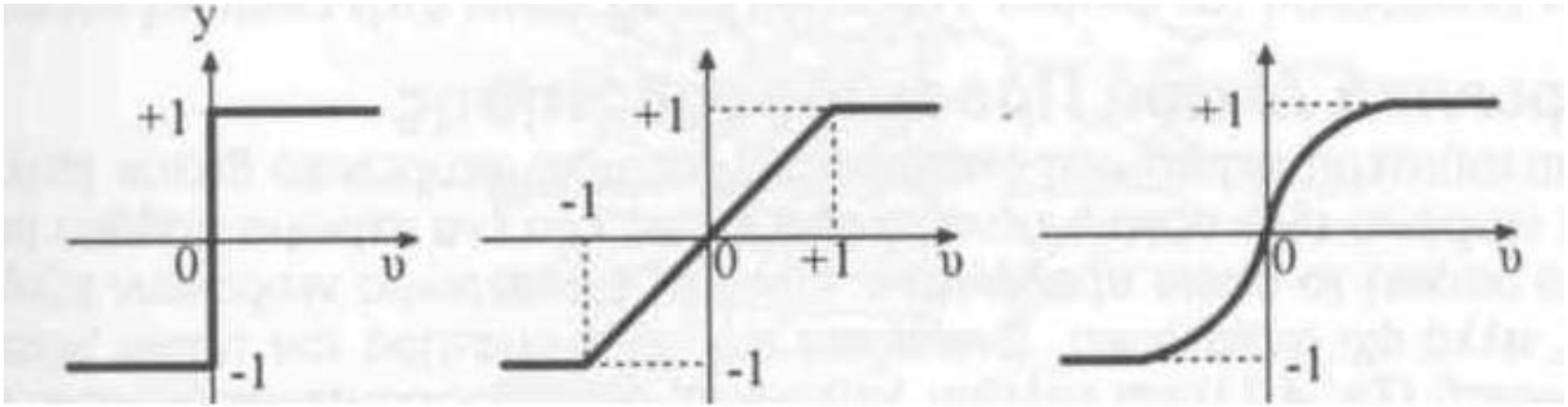
où $\delta \geq 0$

Fonction linéaire par morceaux

$$f(s) = \begin{cases} 0 & \text{si } s \leq -\delta \\ \frac{\delta + s}{2\delta} & \text{si } -\delta < s < \delta \\ 1 & \text{si } s \geq \delta \end{cases}$$

Fonction sigmoïde (Fonction logistique) $f(s) = \frac{1}{1+e^{-\lambda s}}$
où λ est la pente de la fonction

les fonctions d'activation bipolaires



Les fonctions d'activation bipolaires

$f(s) \in [-1, 1]$

Fonction de seuil

$$f(s) = \begin{cases} 1 & \text{si } s \geq \delta \\ -1 & \text{sinon.} \end{cases}$$

où $\delta \geq 0$

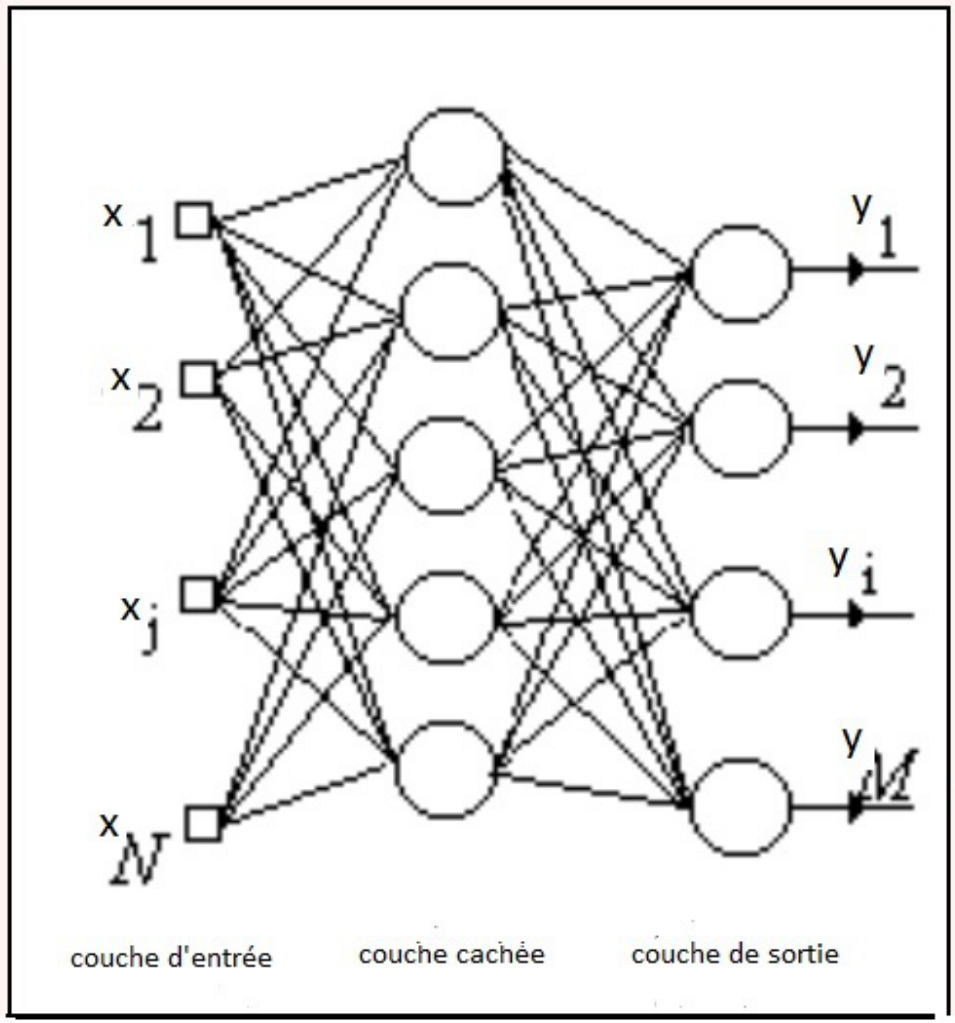
Fonction linéaire par morceaux

$$f(s) = \begin{cases} -1 & \text{si } s \leq -\delta \\ \frac{s}{\delta} & \text{si } -\delta < s < \delta \\ 1 & \text{si } s \geq \delta \end{cases}$$

Fonction sigmoïde $f(s) = \frac{1-e^{-s}}{1+e^{-s}}$

où λ est la pente de la fonction

Réseaux de neurones multicouches



Réseaux de neurones multicouches

- Un réseau de neurones multi-couche est composé d'un ensemble de neurones x_j^l disposé en $L+1$ couches.
- On appelle x_j^l le jème neurone de la couche l.
- La couche 0 étant la couche d'entrée.
- La couche L est la couche de sortie.
- n_l est le nombre de neurones de la couche l. On note $n_0 = n$ et $n_L = m$
- L'état du neurone x_i^l est donné par $s_i^l = -w_{0i}^l + \sum_{j=1}^{n_{l-1}} w_{ji}^l x_j^{l-1}$
- Nous pouvons écrire : $s_i^l = \sum_{j=0}^{n_{l-1}} w_{ji}^l x_j^{l-1}$, avec $X_0^{l-1} = -1$: ?criture matricielle $W^l = [w_{ji}^l]$
- $S^l = W^l x^{l-1}$, $f(S^l) = f(W^l x^{l-1})$

Typologie et fonctionnement des réseaux de neurones

- Par structure
  - Réseaux récurrents.
  - Réseaux multi-couches (ou sans couche cachée)
- Apprentissage Supervisé
  - Pour chaque entrée présentée au RNA on connaît la sortie réelle (désirée).
  - La mise à jour des valeurs des poids et des seuils tient compte de la comparaison entre la sortie réelle et la sortie calculée.
  - Sortie : approximation de la fonction ou bien une regression non-linéaire
  - Exemple : Algorithme de moindres carrées et de rétro-propagation de l'erreur.
- Apprentissage non- supervisé
  - Le réseau de neurones converge vers une représentation synthétique des données.
  - Exemple : les réseaux de neurones de Kohonen.

Tâches principales apprises par les réseaux de neurones-1

Approximation des fonctions par apprentissage supervisé avec RNA non récurrent à une ou plusieurs couches cachées et on utilisera un apprentissage par correction d'erreur.

Auto-association

- Un RNA qui stocke un ensemble des formes et qui sera capable sur présentation d'une de ces formes avec description partielle de la retrouver.
- Apprentissage non supervisé selon la loi de Hebb :
  - La valeur du poids de la connexion entre deux neurones doit être fonction du produit des valeurs d'activation de deux neurones.
  - Le résultat de cette méthode est que les deux neurones ont un comportement synchrone :
  - Le poids de leur connexion augmente et dans le cas contraire, il diminue.

Tâches principales apprises par les réseaux de neurones-2

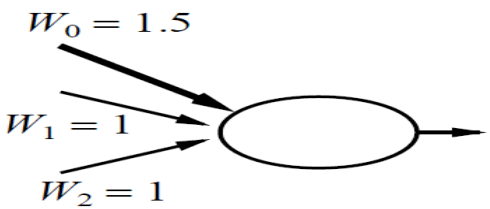
Classification Nous disposons d'un ensemble des classes préexistantes. La tâche du RNA est d'attribuer à un exemple la classe à laquelle il appartient. On peut appliquer un apprentissage supervisé.

Segmentation La tâche du RNA est de représenter les données par des groupes (clusters), des mesures de similarités ou de distance sont utilisés lors du fonctionnement du réseau : Il s'agit d'un apprentissage non supervisé.

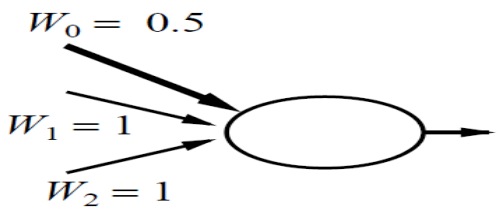
Le perceptron

- Le perceptron a été introduit par Rosenblatt en 1958.
- n entrées et un neurone de sortie dont l'état est une combinaison linéaire des entrées avec les poids.
- Classer une population en deux classes.
- Domaines d'application : la reconnaissance de formes, le traitement de la parole et de l'image, etc.
- Fonction d'activation pour la sortie du perceptron : fonction de seuil binaire, soit bipolaire.

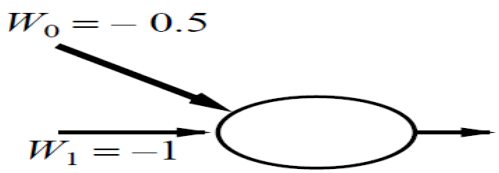
Perceptron et fonctions booléennes



AND

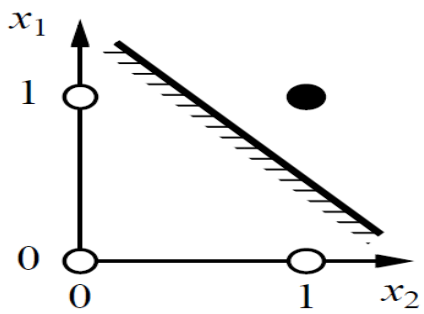


OR

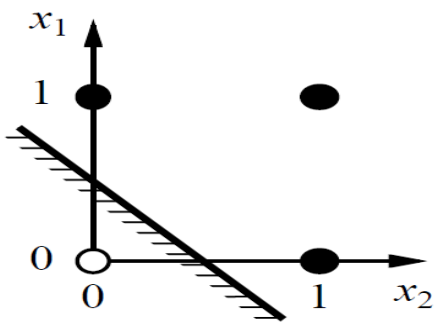


NOT

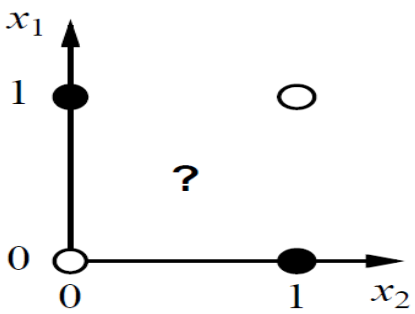
$\sum_j W_j x_j > 0$ or $\mathbf{W} \cdot \mathbf{x} > 0$



(a) x_1 and x_2

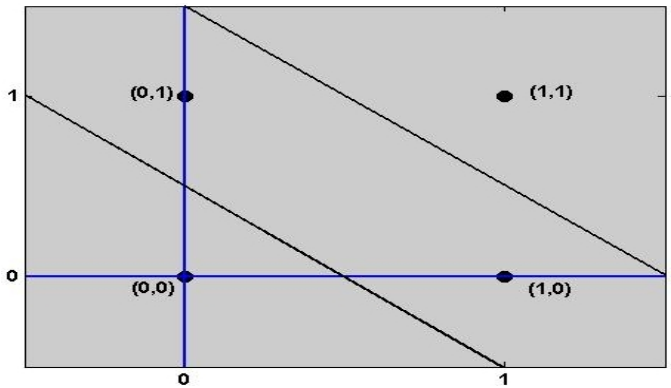


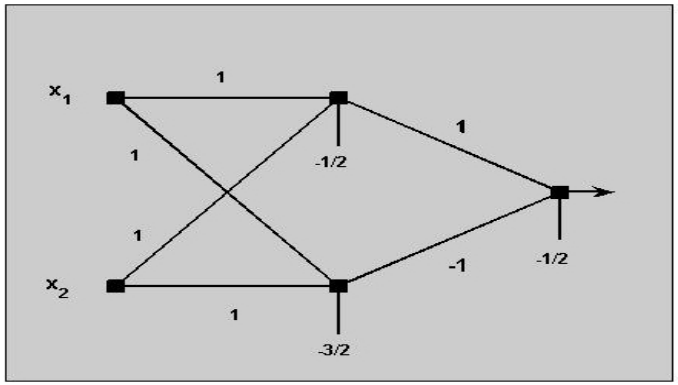
(b) x_1 or x_2



(c) x_1 xor x_2

Le problème XOR





	input 1	input 2	output
object 1	0	0	0
object 2	0	1	1
object 3	1	0	1
object 4	1	1	1

	input 1	input 2	output
object 1	0	0	0
object 2	0	1	0
object 3	1	0	0
object 4	1	1	1

	input 1	input 2	output
object 1	0	0	0
object 2	1	0	1
object 3	1	0	1
object 4	1	1	0

Algorithme du perceptron

Paramètres et Variables

- p entrées-sorties différentes (X_k, Y_k), avec $K = 1..p$, $X_k = (-1, x_1, ..x_n)$,
- Un vecteur de poids $(w_0, w_1, ..w_n)$, avec $w_0 = \theta$
- La sortie calculée Yc_k pour chaque X_k. $Yc_k \in \{0, 1\}$ ou bien $Yc_k \in \{-1, 1\}$
- Un paramètre μ dont le rÙle est de contrÙler la convergence.

Algorithme du perceptron

Initialisation du vecteur W par des valeurs aléatoires.

Apprentissage Répéter (passage de l'ensemble d'apprentissage)

- Répéter selon un ordre donné
  - choisir l'entrée X_k
  - Calculer l'état du neurone $S = W^T X_k$, et sa fonction d'activation $Yc_k = f(S)$
  - Adaptation : $W = W + \mu(Y_k - Yc_k)X_k$
- Test de convergence : Jusqu'à stabilisation du vecteur W.

Algorithme du perceptron

Convergence

- Si Les deux classes sont linéairement séparables alors le perceptron converge, l'hyperplan séparateur des classes est donné par :

$$W^T X = 0$$

- La distance d'un exemple donné par rapport à l'hyperplan est de $d_W(X) = \frac{W^T X}{\sqrt{W^T W}}$

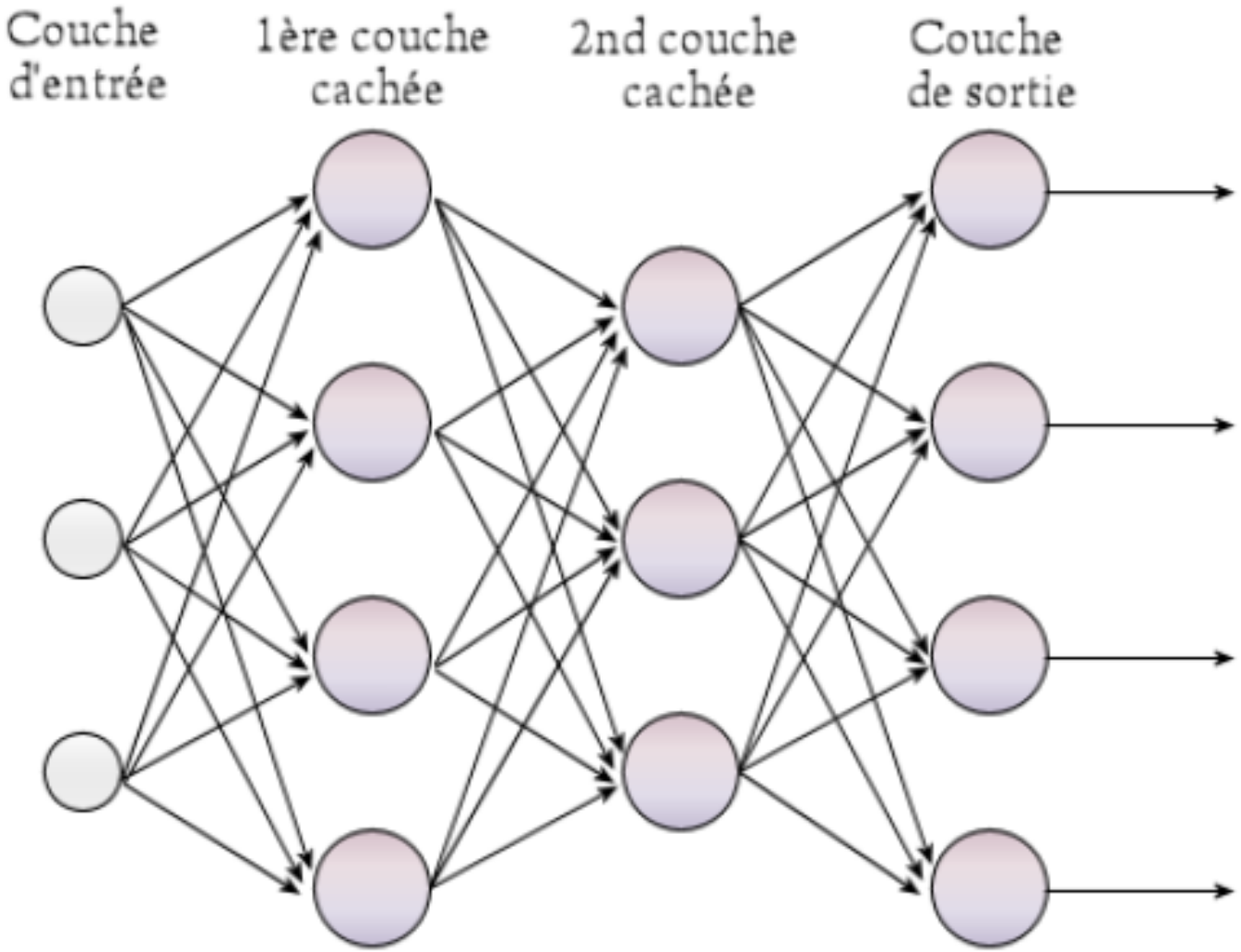
Algorithme du perceptron

Erreur quadratique

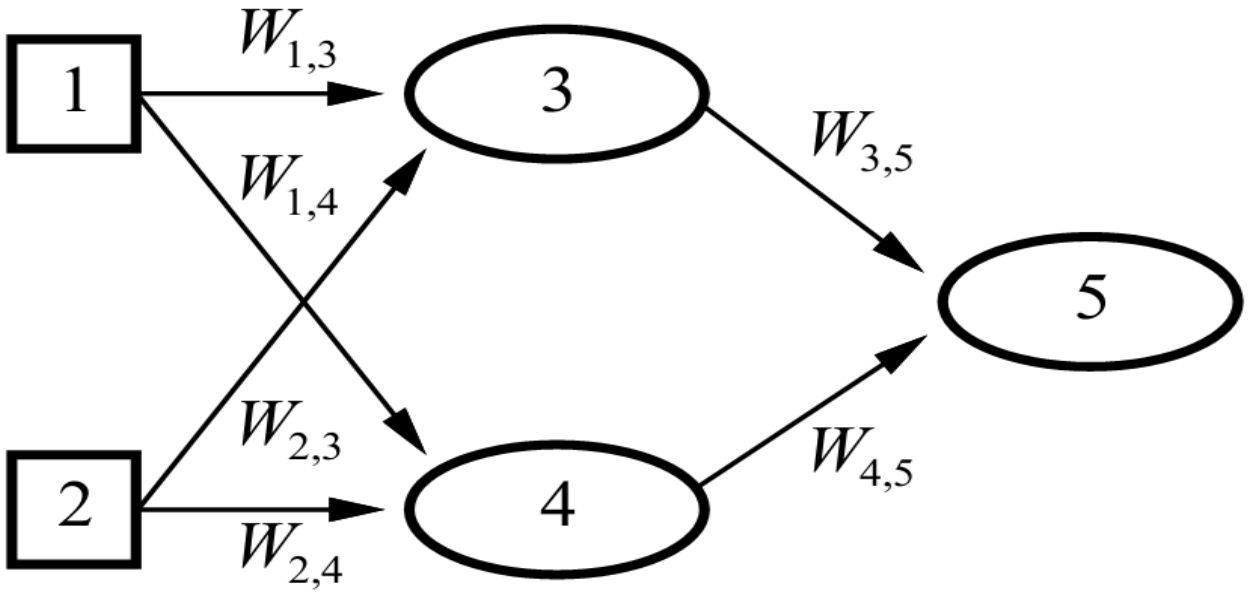
- $Err = (Y - Yc)$ pour l'exemple (X,Y).
- Minimisation de l'erreur quadratique

  $E = \frac{1}{2}Err^2 = \frac{1}{2} \times (Y - Yc)^2$ avec $Yc = f(\sum_i^n w_i x_i - \theta)$ pour l'exemple (X,Y).
- Application de l'optimisation avec l'algorithme de descente du gradient.
- $\frac{\partial E}{\partial w_j} = Err \times \frac{\partial Err}{\partial w_j} = Err \times \frac{\partial (Y - f(\sum_i^n w_i x_i - \theta))}{\partial w_j}$
- $\frac{\partial E}{\partial w_j} = -Err \times f'(X) \times x_j$
- $w_j = w_j + \alpha Err \times f'(X) \times x_j$

Perceptron multi-couches



Perceptron multi-couches



$$x_5 = f(w_{35}x_3 + w_{45}x_4)$$

$$x_5 = f(w_{35} \times f(w_{13}x_1 + w_{23}x_2) + w_{45} \times f(w_{14}x_1 + w_{24}x_2))$$

Perceptron multi-couches

Rétro-propagation de l'erreur

Couche de sortie Comme le perceptron :

$$w_{ji}^L = w_{ji}^L + \alpha \times x_j^{L-1} \times \Delta_i^L \text{ avec}$$

$$\Delta_i^L = Err^L \times f'(X^{L-1})$$

Nous pouvons démontrer que $\frac{\partial E}{\partial w_{ji}^L} = -x_j^{l-1} \times \Delta_i^L$

Couche cachée $\Delta_j^l = f'(X^{l-1}) \times \sum_i^n w_{ji}^l \Delta_i^{l+1}$,

$$w_{kj}^l = w_{kj}^l + \alpha \times x_j^{l-1} \times \Delta_j^l$$

Nous pouvons démontrer que $\frac{\partial E}{\partial w_{kj}^l} = -x_j^{l-1} \times \Delta_j^l$

Exercice-1

On construit un perceptron pour la porte logique AND avec les caractéristiques suivantes :

- les entrées sont binaires et les sorties bipolaires ;
- le taux d'apprentissage est $\mu = 1$;
- le vecteur de poids initaux $W = [3; 2; 3]$ où la première composante correspond au biais.

1. Appliquer l'algorithme du perceptron et tracer pour les différents vecteurs des poids, les droites séparatrices.
2. Donner le plan séparateur de la solution.
3. Calculer pour chaque exemple sa distance au plan séparateur,

Exercice-1-suite

On construit un perceptron pour la porte logique AND avec les caractéristiques suivantes :

- les entrées sont binaires et les sorties bipolaires ;
- le taux d'apprentissage est $\mu = 1$;
- le vecteur de poids initaux $W = [3; 2; 3]$
  où la première composante correspond au biais.

1. ?tudier, sur l'exemple précédent, l'influence du taux d'apprentissage μ sur la convergence. En particulier examiner ce qui se passe si on prend $\mu = 0.5$ avec vecteur des poids $W = [3; 2; 3]$

Exercice-1-suite-2

On construit un perceptron pour la porte logique AND avec les caractéristiques suivantes :

- les entrées sont binaires et les sorties bipolaires ;
- le taux d'apprentissage est $\mu = 1$;
- le seuil $\delta = 0,2$
- le vecteur de poids initaux $W = [3; 2; 3]$
  où la première composante correspond au biais.

1. Appliquer l'algorithme du perceptron et tracer pour les différents vecteurs des poids, les droites séparatrices.
2. Donner le plan séparateur de la solution.
3. Calculer pour chaque exemple sa distance au plan séparateur,

Exercice-2

Soit l'ensemble d'exemples

$$\{([1;1];1);([1;-1];1);([0;1];1);$$

$$([-1;-1];-1);([-1;1];-1);([0;-1];-1)\}$$

1. Vérifier par une représentation graphique que les deux sous-ensembles sont linéairement séparables.
2. En prenant comme vecteur des poids initial $W = [1;1;1]$ et comme $\mu = 1$, appliquer l'algorithme du perceptron jusqu'à la convergence.
3. Calculer pour chaque exemple sa distance du plan séparateur.