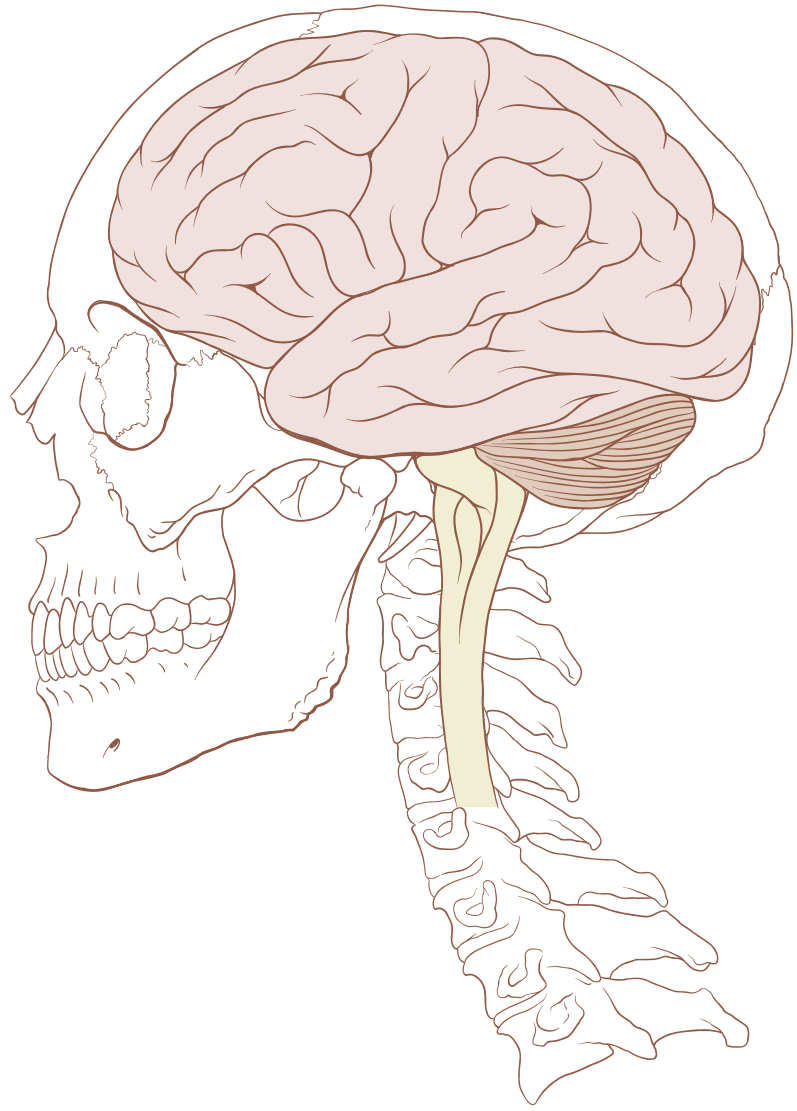
Perceptron simple
Perceptron multi-couches

Nicolas P. Rougier

Master 2 - Sciences Cognitives
Université de Bordeaux

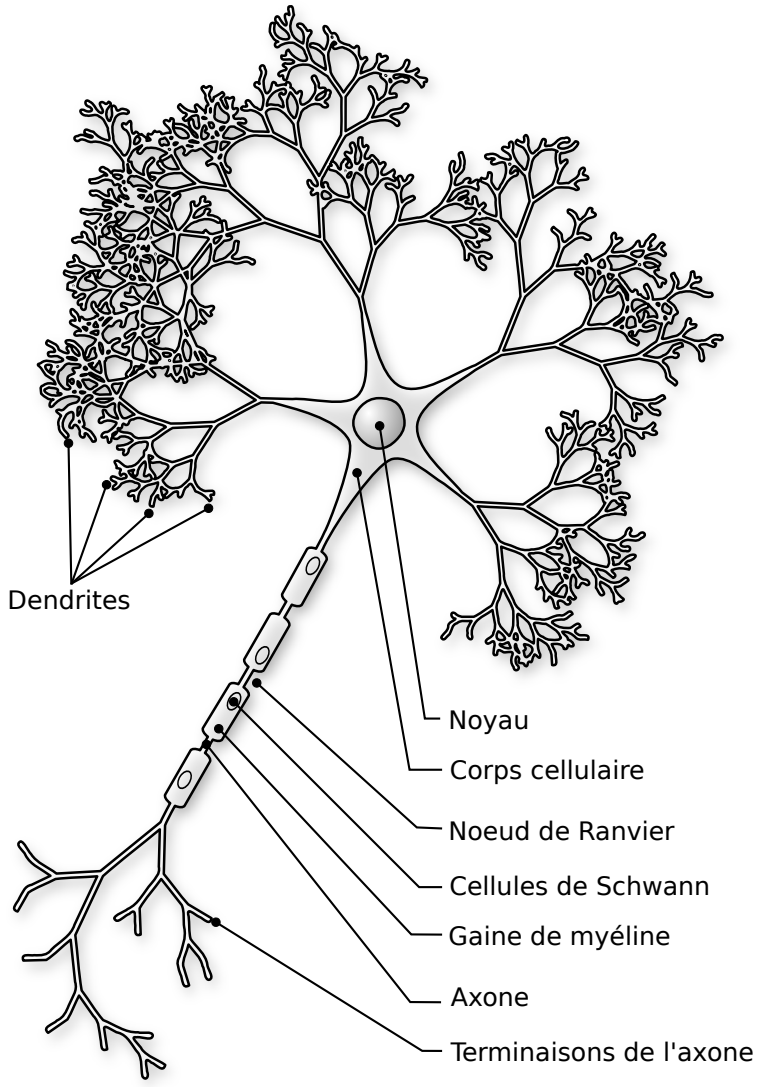
Le cerveau humain

- Nombre de neurones dans le cerveau humain : 100 milliards
- Nombre moyen de connexions par neurone : 10 000
- 1mm^3 de cortex contient un 1 milliard de connexions

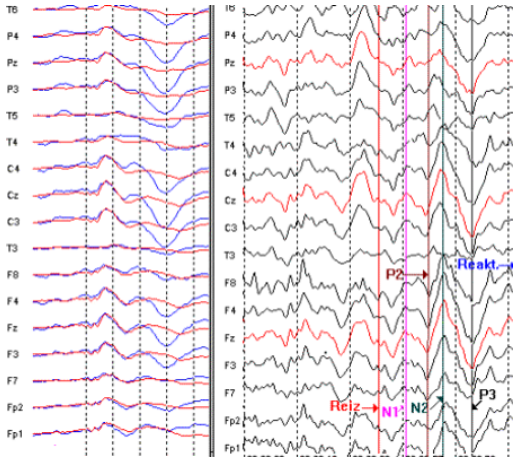


Le neurone biologique

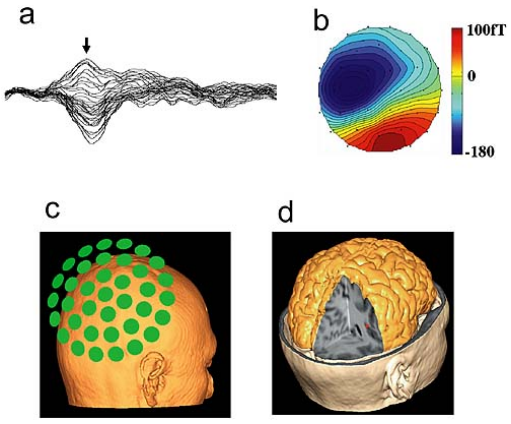
- Un neurone est une cellule capable de transmettre des informations à d'autres neurones au travers de ses différentes connexions (synapses).
- Il existe plusieurs types de neurones (pyramide, panier, Purkinje, etc.) avec des fonctionnements différents (sensoriel, moteur, inter-neurones, etc.)
- Les neurone sont inter-connectés et forment des réseaux



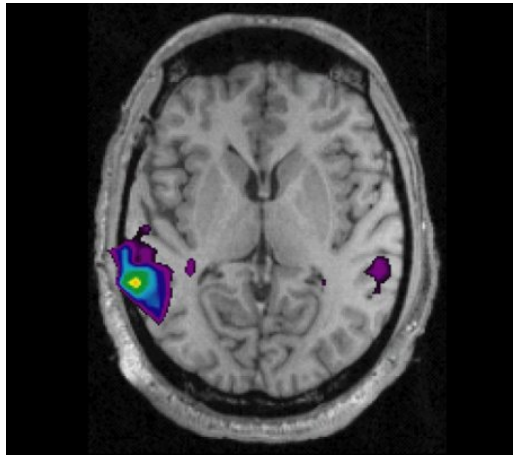
Etudier le cerveau



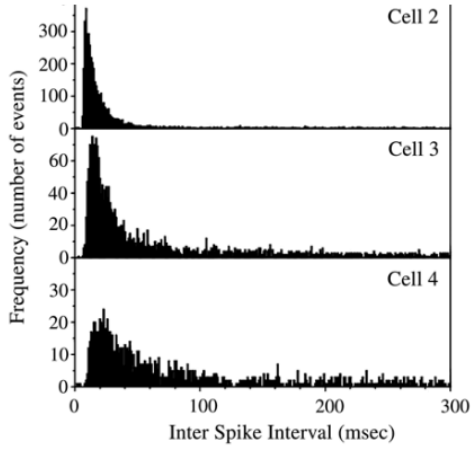
Électro-encéphalographie



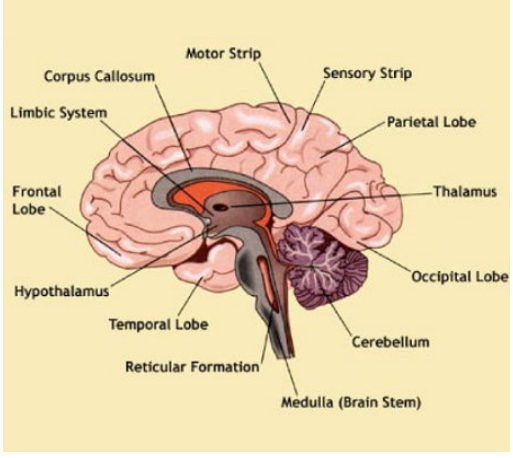
Magnétoencéphalographie (MEG)



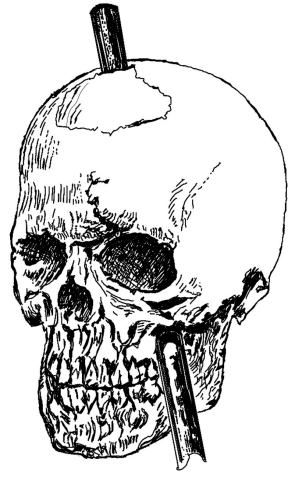
Imagerie par résonance magnétique fonctionnelle (IRMf)



Enregistrement cellulaire



Anatomie



Anomalies/Lésions/Accidents

Modéliser le cerveau

Pourquoi ?

- Pour s'en inspirer
- Pour le comprendre
- Pour le soigner

A quel niveau ?

- Moléculaire ? (neuro-transmetteurs)
- Organitique ? (axones, dendrites, synapses)
- Cellulaire? (neurones, cellules gliales)
- Tissulaire ? (structures, aires fonctionnelles)
- Organique? (cerveau)

Comment ?

- Modéliser un neurone
- Mettre plusieurs neurones en réseau
- Faire apprendre les neurones

Apprendre

Quoi ?

- Généraliser
  → Généraliser une fonction à partir de points déjà connus
- Classifier
  → Prédire si une donnée appartient à telle ou telle classe
- Mémoriser
  → Identifier une information bruitée ou partielle à celles déjà connues
- Regrouper
  → Regrouper des donnés en fonctions de leur similarité

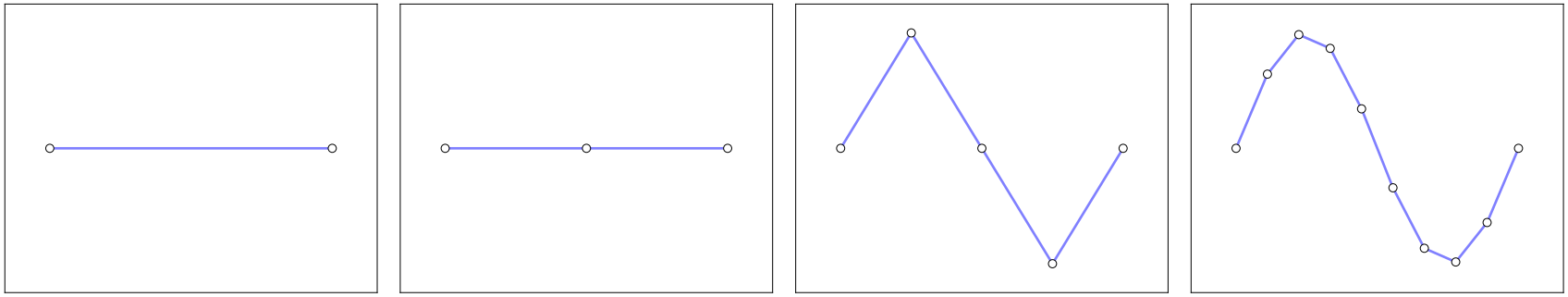
Comment ?

- Supervisé
  → Si la réponse est fausse, on corrige le modèle en donnant la bonne réponse
- Par renforcement
  → Si la réponse est fausse, on communique au modèle que sa réponse est fausse mais sans lui donner la bonne réponse
- Non supervisé
  → On ne dit rien au modèle

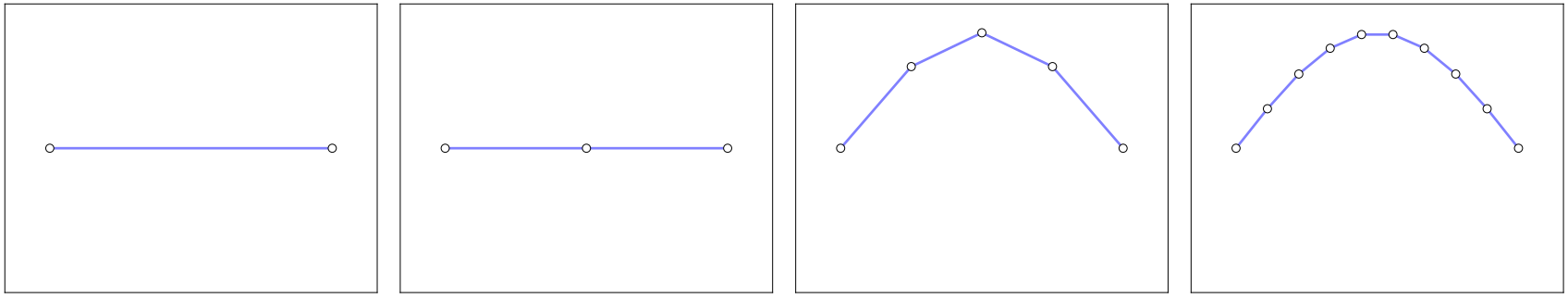
Généraliser une fonction

On donne un ensemble de points (x,y) au modèle et on souhaite que celui-ci puisse prédise la valeur de y pour n'importe quel x donné.

Exemple 1



Exemple 2



Exemples

Série 1

- $(0.1, 0.5) \rightarrow 1$
- $(0.2, 0.9) \rightarrow 1$
- $(0.6, 0.5) \rightarrow 0$
- $(0.7, 0.9) \rightarrow 0$
- $(0.3, 0.7) \rightarrow 1$
- $(0.6, 0.5) \rightarrow$ valeur ?

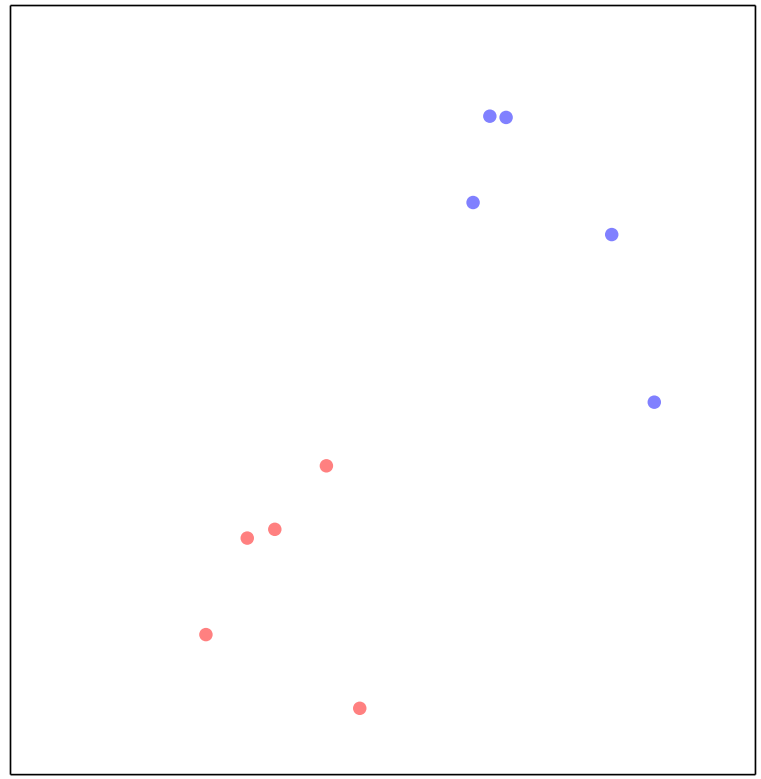
Série 2

- $(0.4, 0.9) \rightarrow 0$
- $(0.0, 0.2) \rightarrow 1$
- $(0.3, 0.6) \rightarrow 0$
- $(0.1, 0.4) \rightarrow 1$
- $(0.2, 0.0) \rightarrow 1$
- $(0.2, 0.7) \rightarrow$ valeur ?

Exemples

Série 3

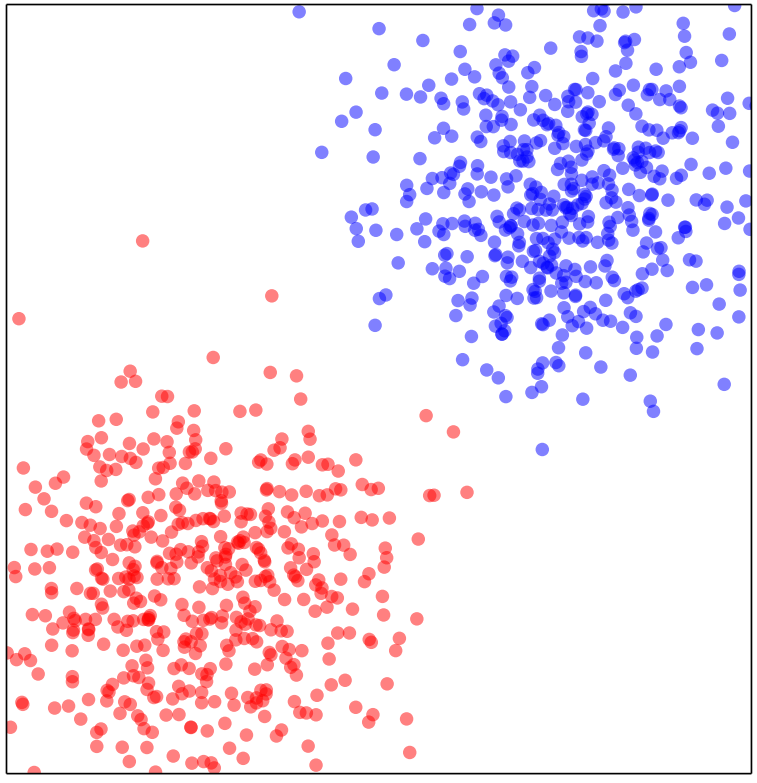
- $(0.134, 0.395) \rightarrow 1$
- $(0.272, 0.989) \rightarrow 1$
- $(0.698, 0.325) \rightarrow 0$
- $(0.701, 0.229) \rightarrow 0$
- $(0.322, 0.773) \rightarrow 1$
- $(0.676, 0.543) \rightarrow$ valeur ?



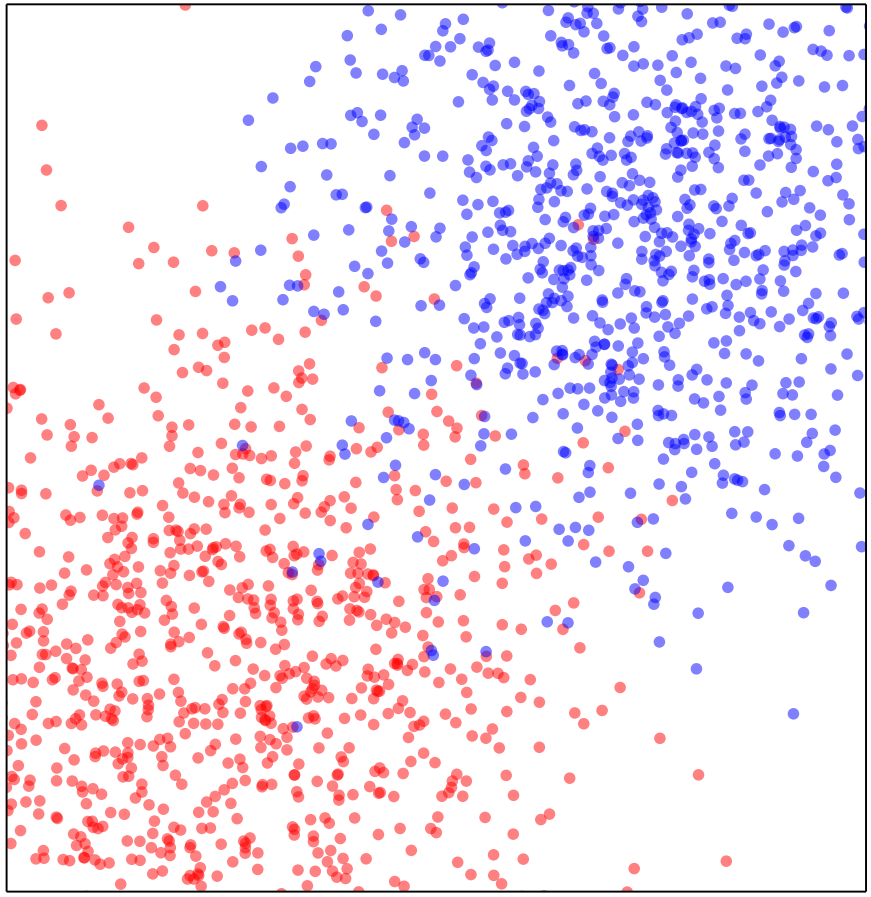
Exemples

Série 3

- $(0.134, 0.395) \rightarrow 1$
- $(0.272, 0.989) \rightarrow 1$
- $(0.698, 0.325) \rightarrow 0$
- $(0.701, 0.229) \rightarrow 0$
- $(0.322, 0.773) \rightarrow 1$
- $(0.676, 0.543) \rightarrow$ valeur ?

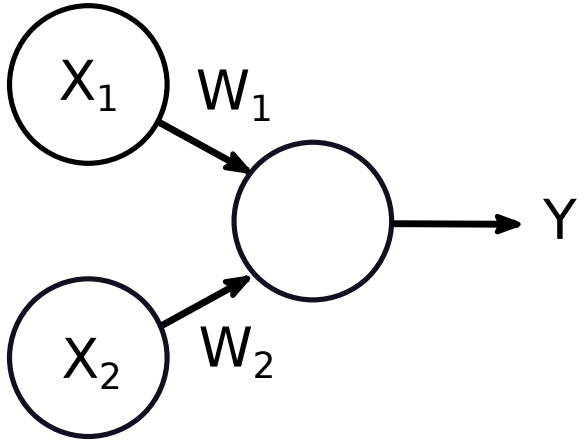


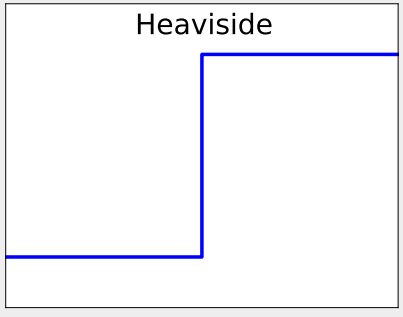
Limites de la méthode

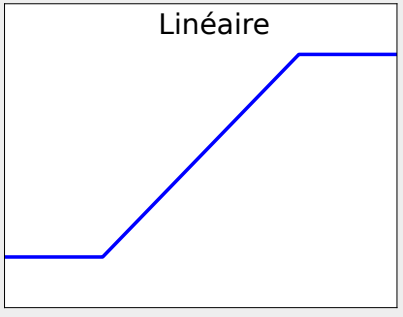


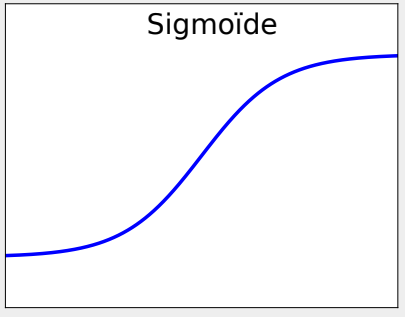
Perceptron simple

- Un neurone possède des entrées
- Chaque entrée possède un poids
- La sortie est une fonction du poids et des entrées
  $Y = f(W_1 * X_1 + W_2 * X_2)$







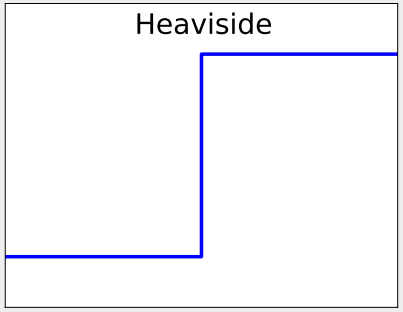


Perceptron simple

Fonctions d'activation (ou fonction de transfert)

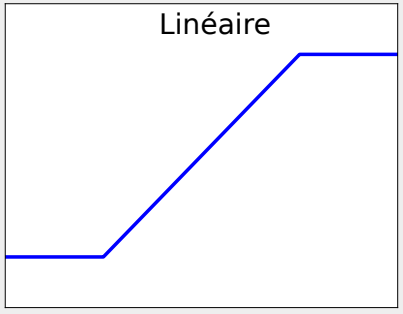
Heaviside (seuil θ)

- Si $x < \theta$ alors f(x) $= 0$
- Si $x \geq \theta$ alors f(x) $= 1$



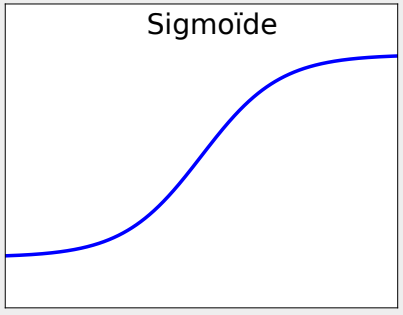
Linéaire (seuil θ_1, θ_2)

- Si $x < \theta_1$ alors f(x) $= 0$
- Si $x > \theta_2$ alors f(x) $= 1$
- Si $\theta_1 \leq x \leq \theta_2$ alors f(x) $=$ x



Sigmoïde / Tangente hyperbolique

- $f(x) = \frac{1}{1+\exp(-x)}$
- $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$



Perceptron simple

Apprentissage supervisé

Calcul de l'erreur

Soit un ensemble de n exemples. On considère la réponse y_k du réseau et la réponse correcte s_k associée à l'exemple k. L'erreur liée à l'exemple k est donc donnée par:

$$E_k = (y_k - s_k)$$

Cette erreur peut être positive $(y_k > s_k)$ ou négative $(y_k < s_k)$

Descente du gradient

Descendre le gradient signifie que l'on cherche à réduire l'erreur dans la direction de l'erreur, en descendant le long du gradient. Si l'on considère les entrées x_i du réseau associés respectivement aux poids w_i, alors:

$$w_i \leftarrow w_i + \alpha(y_k - s_k)x_i$$

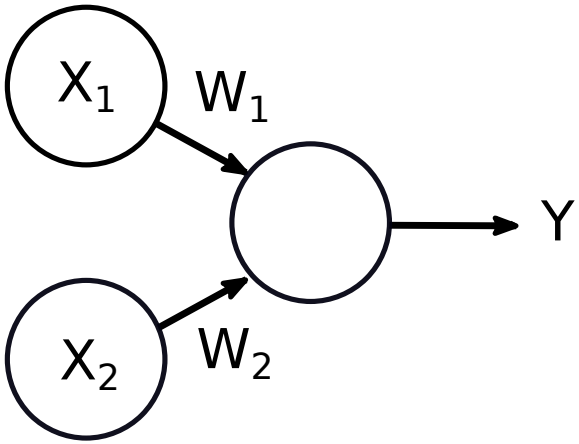
α est le taux d'apprentissage

Perceptron simple

Exercice 1

Soit la fonction logique suivante:

- f(x_1=0,x_2=0) = 0
- f(x_1=1,x_2=0) = 1
- f(x_1=0,x_2=1) = 1
- f(x_1=1,x_2=1) = 1



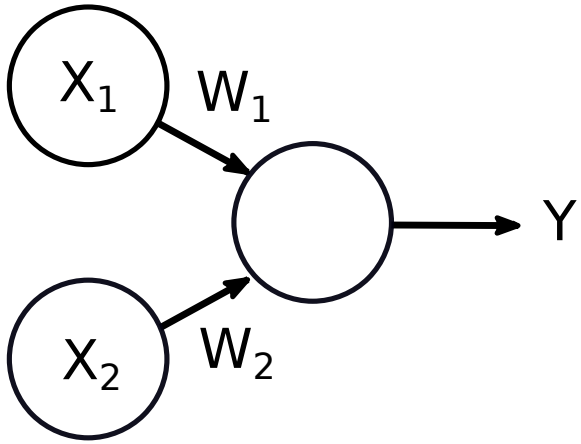
1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil=0).

Perceptron simple

Exercice 2

Soit la fonction logique suivante:

- f(x_1=0,x_2=0) = 0
- f(x_1=0,x_2=1) = 0
- f(x_1=1,x_2=0) = 0
- f(x_1=1,x_2=1) = 1



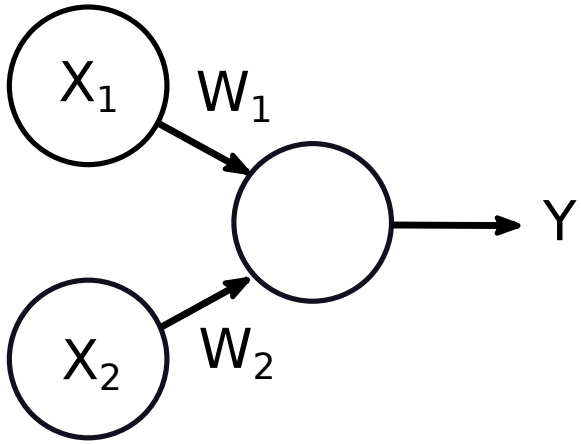
1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil=0).

Perceptron simple

Exercice 2

Soit la fonction logique suivante:

- f(x_1=0,x_2=0) = 0
- f(x_1=0,x_2=1) = 0
- f(x_1=1,x_2=0) = 0
- f(x_1=1,x_2=1) = 1



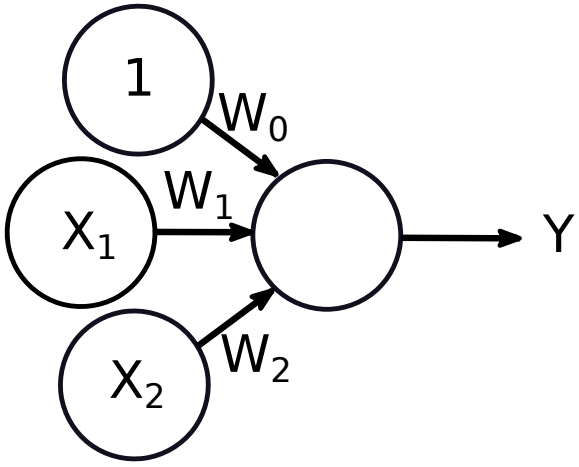
1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil=0).
3. Pourquoi cela ne marche pas ?

Perceptron simple

Exercice 2

Soit la fonction logique suivante:

- f(x_1=0,x_2=0) = 0
- f(x_1=0,x_2=1) = 0
- f(x_1=1,x_2=0) = 0
- f(x_1=1,x_2=1) = 1



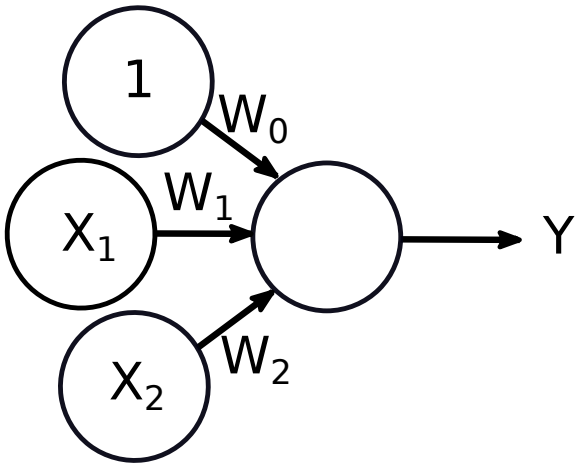
1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil=0).
3. Pourquoi cela ne marche pas ?
4. Réessayer avec la nouvelle architecture

Perceptron simple

Exercice 3

Soit la fonction logique suivante:

- f(x_1=0,x_2=0) = 0
- f(x_1=0,x_2=1) = 1
- f(x_1=1,x_2=0) = 1
- f(x_1=1,x_2=1) = 0



1. De quelle fonction s'agit-il ?
2. A l'aide du réseau donné, trouver les poids w_1 et w_2. On prendra la fonction de Heaviside comme fonction de transfert (seuil=0).

Perceptron simple

Limites

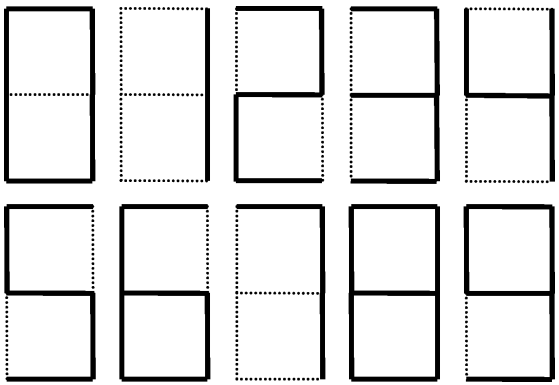
Le perceptron simple ne peut résoudre que des problèmes linéairement séparables. Pour aller plus loin, il est nécessaire d'ajouter des couches.

Et là on a un problème...

Perceptron simple

Exercice 4

Parité



- Proposer un codage binaire de chaque chiffre
- Trouver les poids permettant de décider si le chiffre est pair ou non

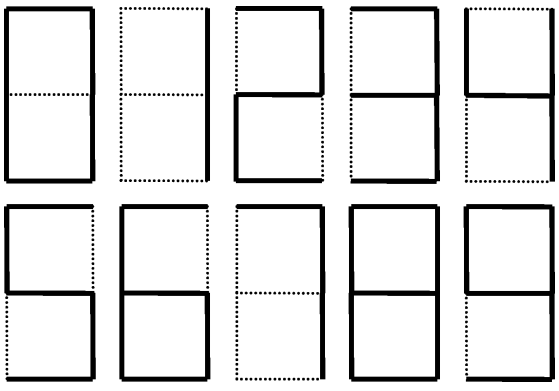
Reconnaissance

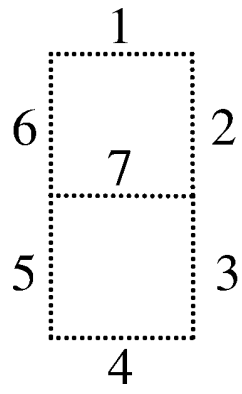
- Proposer une architecture permettant de reconnaitre le chiffre
- Trouver les poids correspondants

Perceptron simple

Exercice 4

Parité





0 se code 1111110,
1 se code 0110000, etc.

- Proposer un codage binaire de chaque chiffre
- Trouver les poids permettant de décider si le chiffre est pair ou non

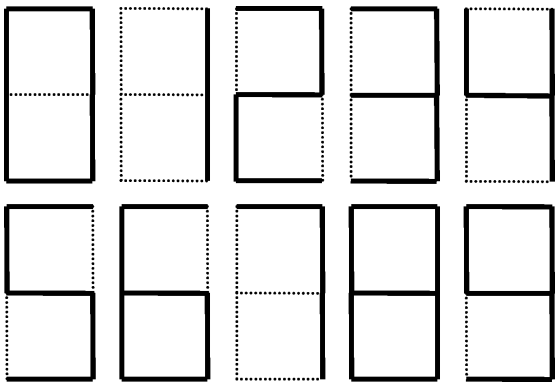
Reconnaissance

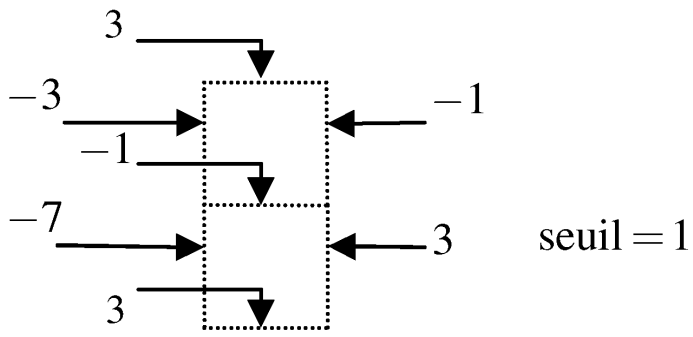
- Proposer une architecture permettant de reconnaitre le chiffre
- Trouver les poids correspondants

Perceptron simple

Exercice 4

Parité





- Proposer un codage binaire de chaque chiffre
- Trouver les poids permettant de décider si le chiffre est pair ou non

Reconnaissance

- Proposer une architecture permettant de reconnaitre le chiffre
- Trouver les poids correspondants

Historique
Sixties

Le neurone formel (McCulloch & Pitts, 1943)

- Automates booléens
- Connexions fixes

Perceptron (Rosenblatt, 1958)

- Modèle linéaire à seuil
- Connexions modifiables

Adaline (Widrow, 1960)

- Modèle linéaire
- Une seule couche
- Connexions modifiables

Historique
Seventies

Champs de neurone (Amari, 1967)

- Continuum neural
- Motifs d'activation

Stabilité/Plasticité (Grossberg, 1968)

- Paramètre de vigilance
- Ré-entrance

Perceptrons (Minsky & Papert, 1969)

- Problème du OU EXCLUSIF
- Arrêt brutal des recherches

Historique
Eigthies

Auto-organisation (Kohonen, 1970)

- Apprentissage non supervisé
- Topologie dans le réseau

Mémoires auto-associatives (Hopfield, 1982)

- Physique statistique
- Apprentissage par coeur

Perceptrons multi-couches

- Rétro-propagation du gradient
- Parker 1982, Le Cun 1985, Rumelhart & McClelland 1986

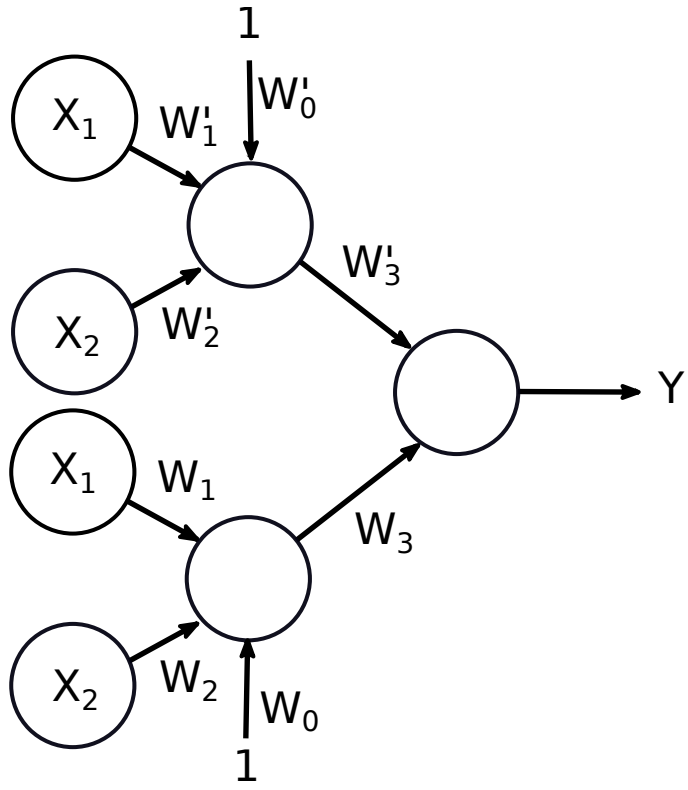
Perceptron multicouche

Exercice 1

Ou exclusif ($\oplus$)

On peut remarquer que
$A \oplus B = (A \vee B) \wedge \neg (A \wedge B)$.

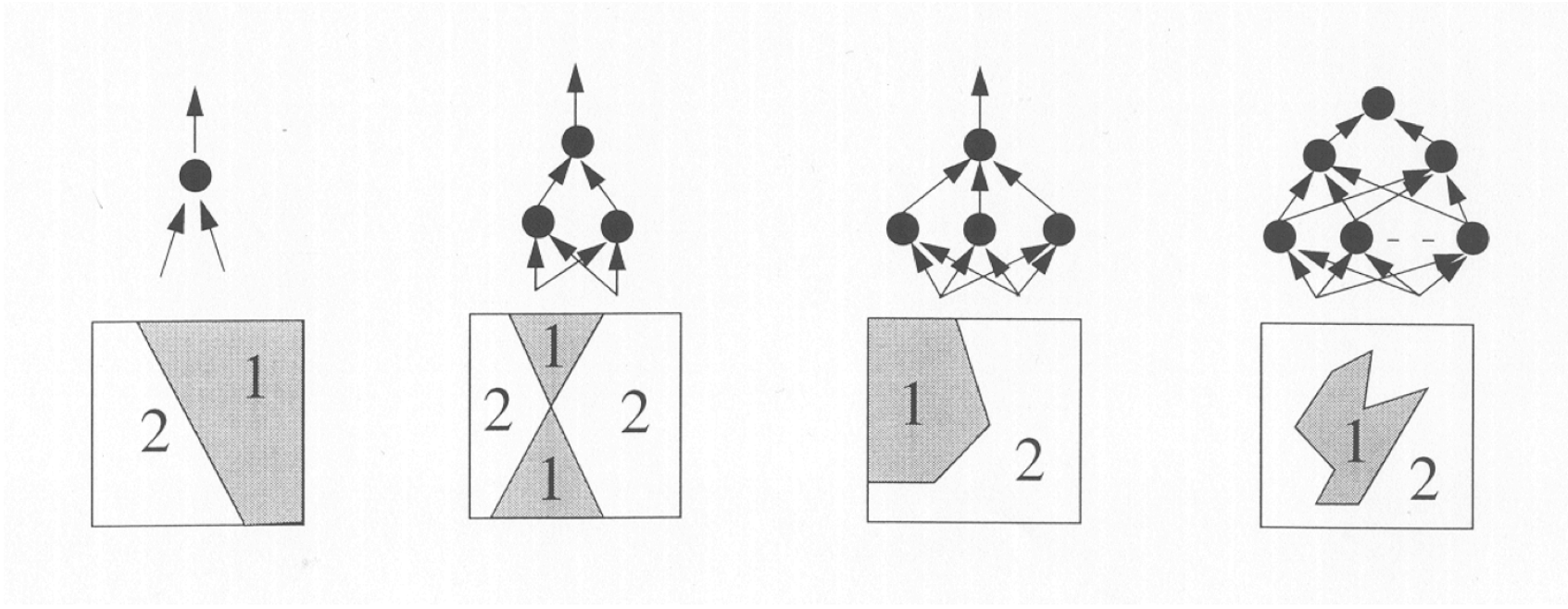
En combinant les deux réseaux (OU et ET), réaliser la fonction ou exclusif



Perceptron multicouche

Approximateur universel de fonctions

Pouvoir séparateur



L'augmentation du nombre de couches et du nombre de neurones accroit le pouvoir de séparation

Perceptron multicouche

Apprentissage

Rétropropagation du gradient

Le problème de l'apprentissage dans les perceptrons multi-couches est de connaitre la contribution de chaque poids dans l'erreur globale du réseau. L'algorothme de rétro-propagation de l'erreur permet de faire cela.

1. Propagation de l'entrée jusqu'à la sortie
2. Calcul de l'erreur en sortie
3. Rétro-propagation de l'erreur jusqu'aux entrées

Conditions

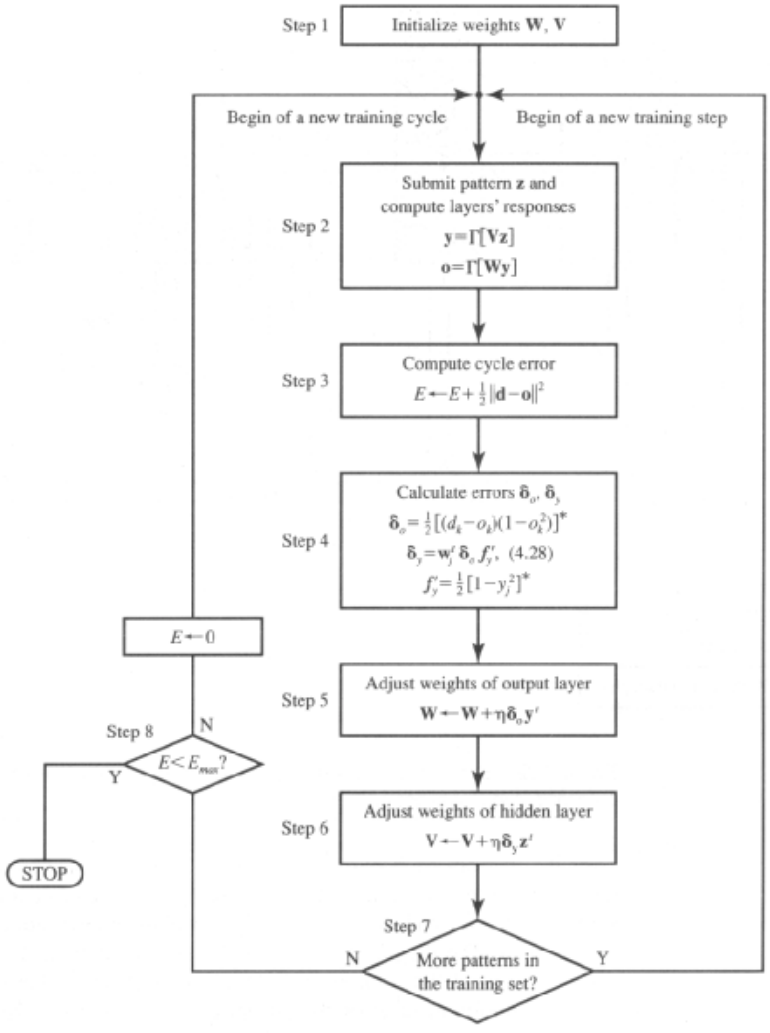
Il faut une fonction d'activation dérivable car on a besoin de la dérivé pour rétro-propager l'erreur.

Détails

Voir "Les réseaux de neurones artificiels", Claude Touzet.

Perceptron multicouche
Algorithme

1. Initialisation
2. Présentation exemple
3. Calcul erreur globale
4. Calcul erreur individuelle
5. Ajustement des poids (couche cachée)
6. Ajustement des poids (couche entrée)
7. Recommencer



Corpus de données

Corpus de données

L'ensemble des données connues et disponibles

Corpus d'apprentissage

Un sous-ensemble du corpus de données qui va servir à l'apprentissage

Corpus de test

Un sous-ensemble du corpus de données qui va servir à vérifier l'apprentissage

Corpus de validation

Un sous-ensemble du corpus de données qui va servir modifier l'apprentissage

Apprentissage

Trop rapide

Un taux d'apprentissage trop rapide peut amener des effets d'sinatabilités dans le réseau.

Trop lent

Un taux d'apprentissage trop lent peut amener le réseau à être bloqué dans un mimumum local.

Inertie (momentum)

On conserve les informations relatifs au dernier apprentissage pour en tenir compte dans l'apprentissage courant. On évite les effets d'oscillations ou bien de rester coincé dans un minimum local.

Mesure de l'erreur

Erreur apparente

L'erreur apparente se mesure sur le corpus d'apprentissage.

Erreur réelle

L'erreur réelle se mesure sur le corpus entier.

Si l'erreur apparente est très faible alors que l'erreur réelle est très forte, le corpus d'apprentissage est très certainement mal échantilloné.

Bibliographie

Livres

- *Les réseaux de neurones : Introduction connexionnisme*
  Claude Touzet, 1992
- *Pattern Recognition And Machine Learning*
  Christopher M. Bishop, Springer, 2006
- *Apprentissage statistique*
  Gérard Dreyfus, Jean-Marc Martinez, Mannuel Samuelides
  Mirta Gordon, Fouad Badran, Sylvie Thiria, Eyrolles, 2008

Concepts

- *Pruning techniques*
- *Deep learning network*
- *Mixture of models*

Perceptron multi-couche

Exercice 2

Nécessite

- Python (`www.python.org`)
- Numpy (`www.numpy.org`)
- Matplotlib (`matplotlib.org`)

Perceptron simple

Tester l'apprentissage du ou exclusif avec un perceptron:
`www.loria.fr/~rougier/downloads/perceptron.py`

Perceptron multi-couches

Tester l'apprentissage du ou exclusif avec un perceptron:
`www.loria.fr/~rougier/downloads/mlp.py`

Perceptron multi-couche

Exercice 3

Réaliser un perceptron multi-couche permettant de détecter si un point (x,y) se trouve à l'intérieur ou à l'extérieur d'un triangle équilatéral centré en (0,0) et inscrit dans le cercle unité.

Choisir d'abord l'architecture du réseau (nombre d'unité, connections, fonctions de transfert, etc) puis générer des exemples positifs (points à l'intérieur) et négatifs (points à l'extérieur) Le plus rapide est d'écrire un programme pour générer autant d'exemple que l'on veut.

Vous pourrez utiliser le logiciel `ginnet.gforge.inria.fr` pour faire apprendre votre perceptron.