Plan du cours

- Algorithmes de recherche
- Algorithmes de recherche et heuristiques
- Problème de satisfaction de contraintes
- Jeux à deux joueurs

  minimax, élagage α-β, MCTS

} Good old fashion AI (GOFAI)

Plan du cours

- Algorithmes de recherche
- Algorithmes de recherche et heuristiques
- Problème de satisfaction de contraintes
- Jeux à deux joueurs

  minimax, élagage α-β, MCTS

} Good old fashion AI (GOFAI)✔

Plan du cours

- Algorithmes de recherche
- Algorithmes de recherche et heuristiques
- Problème de satisfaction de contraintes
- Jeux à deux joueurs

  minimax, élagage α-β, MCTS

} Good old fashion AI (GOFAI)✔

- Réseaux de neurones
- Apprendre un arbre de décision

} Apprentissage automatique

Plan du cours

- Algorithmes de recherche
- Algorithmes de recherche et heuristiques
- Problème de satisfaction de contraintes
- Jeux à deux joueurs

  minimax, élagage α-β, MCTS

} Good old fashion AI (GOFAI)✔

- Réseaux de neurones
- Apprendre un arbre de décision

} Apprentissage automatique

- AlphaGo et AlphaGo-Zéro

Plan du cours

- Algorithmes de recherche
- Algorithmes de recherche et heuristiques
- Problème de satisfaction de contraintes
- Jeux à deux joueurs

  minimax, élagage α-β, MCTS

} Good old fashion AI (GOFAI)✔

- Réseaux de neurones
- Apprendre un arbre de décision

} Apprentissage automatique

- AlphaGo et AlphaGo-Zéro
- Intelligence Artificielle et Ethique

Apprentissage Automatique

- Apprentissage supervisé :
  - on a des exemples (beaucoup !) et une étiquette pour chaque exemple
  - on veut utiliser un algorithme pour faire ensuite de la prédiction pour l'étiquette : on donne un exemple, il faut devinier l'étiquette
- Apprentissage non supervisé :
  - on a des exemples, mais pas d'étiquette
  - clustering ⇨ découvrir une structure cachée dans les données
- apprentissage par renforcement :
  - un agent interagit avec un environnement
  - il reçoit une observation et un signal qui dépendent de l'action exécutée par l'agent

  ⇨ le **but** est de maximiser ce signal

Apprentissage Automatique

- Apprentissage supervisé : cours AFD, puis en **M2-ID**, **M2 ISI**
  - on a des exemples (beaucoup !) et une étiquette pour chaque exemple
  - on veut utiliser un algorithme pour faire ensuite de la prédiction pour l'étiquette : on donne un exemple, il faut devinier l'étiquette
- Apprentissage non supervisé :
  - on a des exemples, mais pas d'étiquette
  - clustering ⇨ découvrir une structure cachée dans les données
- apprentissage par renforcement : cours en **M2 ISI**
  - un agent interagit avec un environnement
  - il reçoit une observation et un signal qui dépendent de l'action exécutée par l'agent
  - ⇨ le **but** est de maximiser ce signal

Dans ce cours, on va voir deux exemples d'apprentissage supervisé

- arbres de décision
- réseaux de neurones

Apprentissage supervisé

- on a un ensemble d'exemples
- on a une étiquette pour chaque exemple
  un exemple : données médicales sur un patient
  étiquette : est-ce que ce patient est atteint par une maladie X ?
- apprendre : pas apprendre "par coeur"

⇨ on voudra utiliser ce qu'on apprend pour de la prédiction
sur un exemple jamais vu, on devra deviner l'étiquette

⇨ l'apprentissage doit permettre de généraliser au delà des exemples donnés

⇨ comment valider une méthode ?

Réseaux de Neurones Artificiels

M1 Informatique 2018–2019 *Intelligence Artificielle*

Stéphane Airiau

LAMSADE

DAUPHINE
UNIVERSITÉ PARIS

- S'inspirer du cerveau humain pour construire un système
- Ici, on s'inspire des neurones
  ⇨ batir des réseaux de neurones artificiels
- vieille idée, McCulloch et Pitts ont présenté un modèle mathématique d'un neurone dès 1943.

Inspiration

- le cerveau : réseau interconnecté très complexe de neurones
- ⇨ Réseau de Neurones Artificiels (RNA) : réseau densémment connecté de simples unités de calcul
- le cerveau
  - $\approx 10^{11}$ neurones
  - chaque neurone est connecté à 10,000 autres neurones
  - le temps de réaction d'un neurone est autour de 10^{-3}s (lent par rapport à un ordinateur)
  - 0.1s pour réaliser une tâche de reconnaissance (ex : montre la photo d'une personnalité ou d'un ami)
    - ⇨ au plus quelques centaines d'étapes pour le calcul.
  - ⇨ calcul vraissemblablement massivement parallèle
- réseaux de neurones pour comprendre mieux le cerveau humain
- réseaux de neurones comme source d'inspiration pour un méchanisme d'apprentissage efficace

Réseau de neurones

- domaine de recherche ancien
- ex dès 1993, un RNA a été utilisé pour gérer le volant d'une voiture roulant sur une autoroute (aux Etats Unis)
  - input : images d'une caméra avec une résolution de 30x32 pixels
    ⇨ chaque pixel est connecté à un neurone d'entrée
  - 4 neurones cachés
  - 30 neurones pour la sortie (un neurone va correspondre à un angle du volant)

Situations pour utiliser un RNA

- entrée est un vecteur de large dimension de valeurs discrètes ou continues (e.g. sortie d'un capteur)
- la sortie est discrète ou continue
- la sortie est un vecteur de valeurs
- les valeurs d'entrées peuvent avoir du bruit (ou contenir des erreurs)
- on n'a pas d'hypothèse sur la fonction à apprendre
- il n'y a pas besoin qu'un humain puisse lire le résultat (par ex pour le vérifier ou comprendre ce qui est appris)
- un temps d'apprentissage long est acceptable
- l'utilisation du RNA est rapide

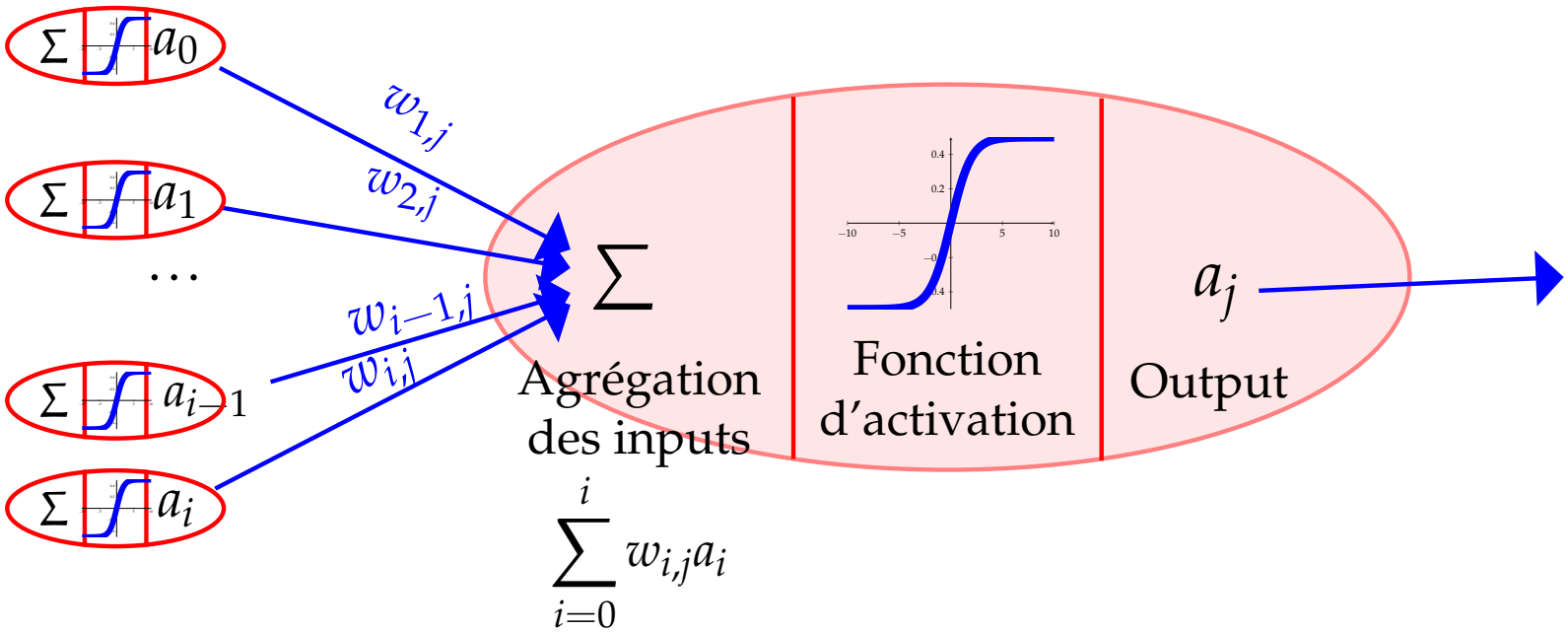
Exemples :

- Apprendre à reconnaître des sons
- Classification d'images
- Prédictions financiaires

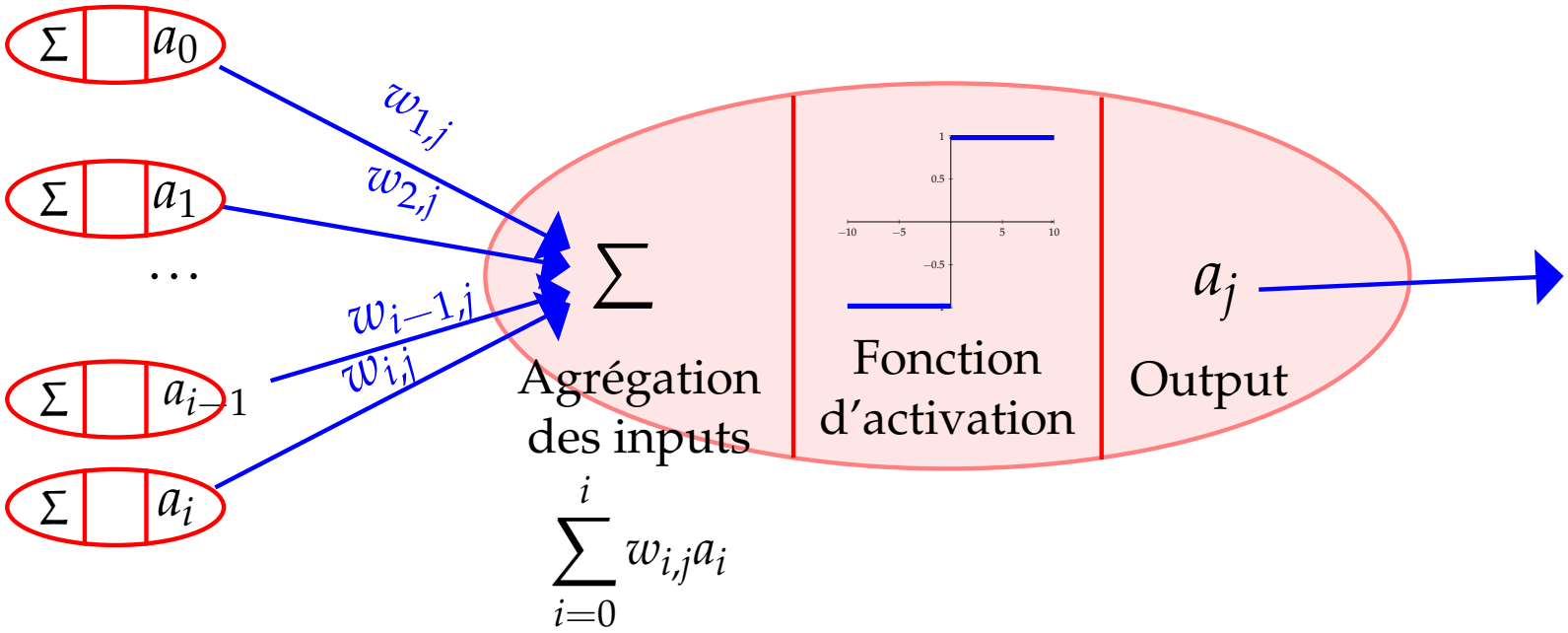
Deep Learning

- Coloration d'images en noir et blanc
- Traduction automatique
- classification d'objets dans des photos
- Game Playing.

Modèle d'un neurone : le perceptron



Modèle d'un neurone : le perceptron



- Fonction d'activation : fonction de Heaviside
- perceptron représente un hyperplan $\overrightarrow{w} \cdot \overrightarrow{x} = 0$ et les instances d'un côté de l'hyperplan seront positives et négatives de l'autre
- on peut représenter des fonction booléennes avec un seul perceptron, mais pas toutes (XOR).
- avec un réseau de perceptrons, on peut représenter toutes les fonctions booléennes

Apprendre pour **un** perceptron

- on doit apprendre les poids w_i entre les entrées $1, 2, \ldots, n$ et le perceptron.
- on peut commencer par des poids pris au hasard
- on présente une série d'instances et on corrige les poids après chaque erreur
  - t est la vraie classification de la sortie
  - o est la sortie générée par le perceptron
  - x_i est l'état de l'entrée numéro i

$$w_i \leftarrow w_i + \Delta w_i$$

où $\Delta w_i = \eta(t-o)x_i$

  - η est le taux d'apprentissage
    (une petite valeur comme 0.1 est habituellement utilisée)
- si les données peuvent être apprises par un perceptron (on dit que les données sont alors linéairement séparables), ce simple algorithme **converge**.

Apprendre pour **un** perceptron **linéaire**

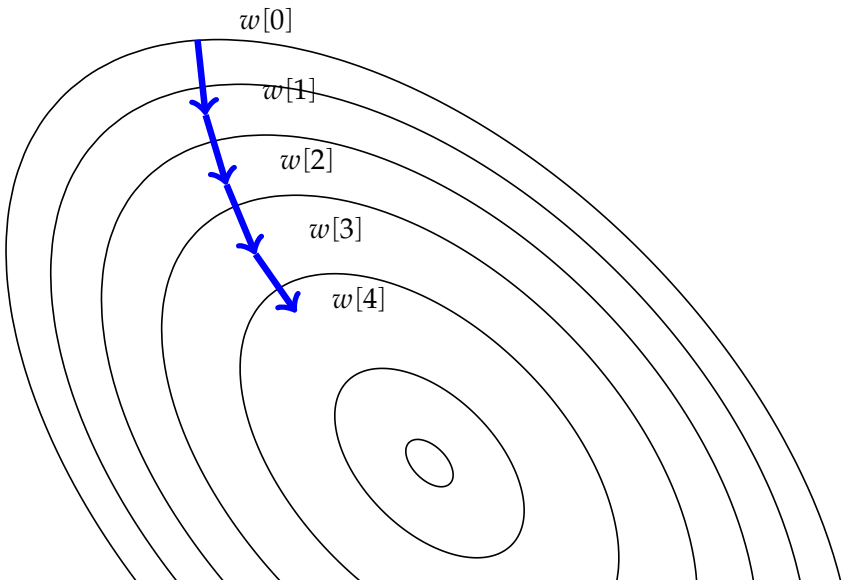
- la fonction d'activation est l'**identité**.
- Si les données sont linéairement séparables, la méthode précédente converge
- et sinon ? ⇨ garantir convergence à la meilleure approximation
- on va utiliser le principe la **descente de gradient** (la plus grande pente)
- L'erreur d'apprentissage est souvent mesurée par

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

où

- D est l'ensemble de exemples d'apprentissage
- t_d est la vraie classification de l'instance d
- o_d est la réponse du peceptron pour l'instance d

Descente de gradient



le gradient indique la direction de la pente la plus forte
La règle de mise à jour sera donc :

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w} \qquad w_i \leftarrow w_i + \Delta w_i$$

$$\text{où } \Delta\vec{w} = -\eta\nabla E(\vec{w}) \qquad \Delta w_i = -\eta\frac{\partial E}{\partial w_i}$$

- η est le taux d'apprentissage qui determine la taille du pas que l'on fait en descendant
- le signe négatif indique que l'on veut descendre

Heureusement, calculer la dérivée est facile ici !

Algorithme de descente de gradient

```
1  initialise chaque w_i avec une valeur au hasard
2  tant que l'erreur est trop grande
3      Pour chaque i ∈ {1,...,n}
4          Δw_i = 0
5      Pour chaque example (x⃗, t) ∈ D
6          calculer la sortie o
7          Pour chaque i ∈ {1,...,n}
8              Δw_i = Δw_i + η(t − o)x_i
9      Pour chaque i ∈ {1,...,n}
10         w_i ← w_i + Δw_i
```

Algorithme pour **une** unité avec une fonction d'activation **linéaire**

Descente de gradient stochastique

- la descente peut être lente
- s'il y a plusieurs minimaux locaux, on n'a pas de garantie de trouver le minimum global

On utilise souvent une variante qui consiste à mettre à jour les poids après l'examen de chaque point (et pas après de les avoir tous examinés)

- c'est souvent une bonne approximation
- demande un peu moins de calculs
- permet parfois d'éviter de tomber dans un minimum local
  ⇨ plus de chances de tomber dans me minimum global