Le Perceptron

M1 MIAGE *Machine Learning & Applications*

Stéphane Airiau

LAMSADE

DAUPHINE
UNIVERSITÉ PARIS

- S'inspirer du cerveau humain pour construire un système
- Ici, on s'inspire des neurones
  ⇒ bâtir 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 :

- analyse de textes, traductions, résumés
- reconnaissances d'objets dans des images, reconnaissance faciale, colorisation d'images ou de vidéos en noir et blanc
- …

Deep Learning

"deep" réfère à un réseau de neurones qui possède beaucoup de couches de neurones ⇒ nécessite un long temps pour l'apprentissage.

spécificité à voir en M2.

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

Aujourd'hui : modèle **d'un seul** neurone : le perceptron

Principe :

- un neurone reçoit des signaux d'autres neurones ou de capteurs
- le neurone combine ces signaux
- si la combinaison dépasse un seuil, le neurone envoie un signal (influx nerveux)

Entrées d'un neurone :

- soit c'est une sortie d'un autre neurone
- soit c'est une entrée provenant d'un capteur
  (par exemple valeur d'un pixel)

Dans un neurone, on attribue un poids $w_i \in \mathbb{R}$ à la $i^{\text{ième}}$ entrée.

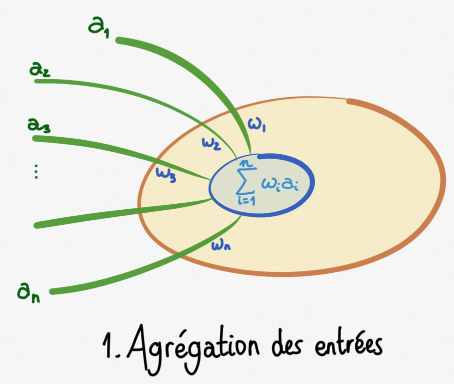
Trois étapes (le perceptron possède n entrées) :

1- on agrège toutes les entrées en calculant une somme pondérée

2- on applique une fonction d'activation

3- on retourne le résultat

Exécution du perceptron



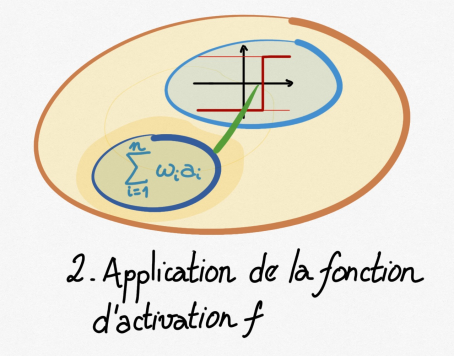
Trois étapes (le perceptron possède n entrées) :

1- **on agrège toutes les entrées en calculant une somme pondérée**

2- on applique une fonction d'activation

3- on retourne le résultat

Exécution du perceptron



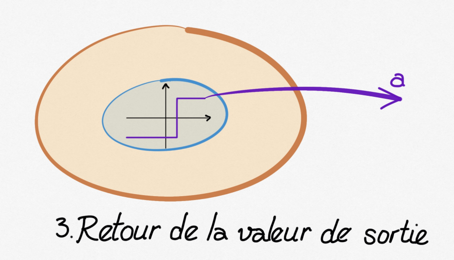
Trois étapes (le perceptron possède n entrées) :

1- on agrège toutes les entrées en calculant une somme pondérée

2- **on applique une fonction d'activation** ⇨ on calcule $f\left(\sum_{i=1}^{n} w_i a_i\right)$

3- on retourne le résultat

Exécution du perceptron



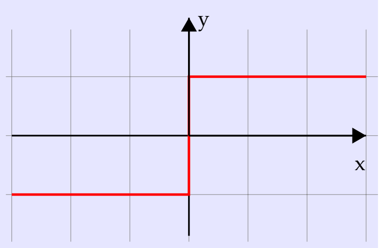
Trois étapes (le perceptron possède n entrées) :

1- on agrège toutes les entrées en calculant une somme pondérée

2- on applique une fonction d'activation

3- **on retourne le résultat**

Fonction d'activation



- pour conserver l'idée d'un seuil d'activation, on va utiliser une fonction de seuil (ou Heaviside) où la valeur seuil est 0.

$$f : \begin{array}{l} \mathbb{R} \to [-1,1] \\ x \mapsto \begin{cases} x < 0 : -1 \\ x \geqslant 0 : 1 \end{cases} \end{array}$$

- on pourra utiliser d'autres fonctions plus tard (*tanh*, fonction logistique, ReLu)

Règle de décision du perceptron

Avec la fonction seuil, notre perceptron encode la fonction suivante :

- si $\sum_{i=1}^{n} w_i a_i < 0$ le perceptron retourne -1
- si $\sum_{i=1}^{n} w_i a_i \geqslant 0$ le perceptron retourne +1

Le comportement de notre perceptron va donc dépendre des poids w_i.

Petite astuce : Au lieu de choisir 0 comme valeur seuil, on peut choisir la valeur à -1, et lui donner un poids w_0.
Le comportement du perceptron sera alors donnée par

$$\sum_{i=0}^{n} w_i a_i$$

où a_0 a pour valeur 1.

Exemple

$$w_0 = 1 \text{ rappel : } a_0 = 1$$

$$w_1 = 1$$

$$w_2 = 1$$

a_1	a_2	$w_0a_0 + w_1a_1 + w_2a_2$	sortie
-1	-1	-1	-1
-1	1	1	+1
1	-1	1	+ 1
1	1	3	+1

Ca vous rappelle quelque chose ?

Exemple

$$w_0 = 1 \text{ rappel : } a_0 = 1$$

$$w_1 = 1$$

$$w_2 = 1$$

a_1	a_2	sortie
-1	-1	-1
-1	1	+1
1	-1	+ 1
1	1	+1

- -1 correspond à faux
- +1 correspond à vrai

a_1	a_2	$a_1 \vee a_2$
faux	faux	faux
faux	vrai	vrai
vrai	faux	vrai
vrai	vrai	vrai

Table de vérité de l'opérateur logique OU

Pouvez vous trouvez les poids pour l'opérateur ET ?

a_1	a_2	$a_1 \wedge a_2$
faux	faux	faux
faux	vrai	faux
vrai	faux	faux
vrai	vrai	vrai

Table de vérité de l'opérateur logique ET

$w_0 =$? rappel : $a_0 = 1$

$w_1 =$?

$w_2 =$?

a_1	a_2	$w_0a_0 + w_1a_1 + w_2a_2$	sortie
-1	-1		-1
-1	1		-1
1	-1		-1
1	1		+1

Pouvez vous trouvez les poids pour l'opérateur XOU OU EXCLUSIF ?

a_1	a_2	$a_1 \oplus a_2$
faux	faux	faux
faux	vrai	vrai
vrai	faux	vrai
vrai	vrai	faux

Table de vérité de l'opérateur logique XOU OU EXCLUSIF

$$w_0 = ? \text{ rappel : } a_0 = 1$$

$$w_1 = ?$$

$$w_2 = ?$$

a_1	a_2	$w_0a_0 + w_1a_1 + w_2a_2$	sortie
-1	-1		-1
-1	1		+1
1	-1		+1
1	1		-1

Règle de décision du perceptron

Avec la fonction seuil, notre perceptron encode la fonction suivante :

- si $\sum_{i=1}^{n} w_i a_i < 0$ le perceptron retourne -1 ➩ **classe comme OUI**
- si $\sum_{i=1}^{n} w_i a_i \geqslant 0$ le perceptron retourne +1 ➩ **classe comme NON**

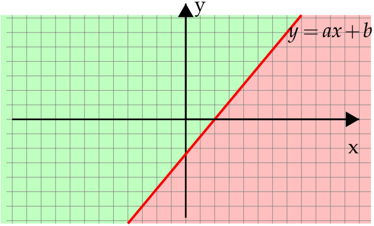
Avec des vecteurs de dimension n, on pourra donc utiliser le perceptron pour faire de la classification binaire !

Règle de décision du perceptron – dimension 2

- 2 dimensions, donc 2 entrées a_1 et a_2
- ➯ changeons de noms : $x = a_1$ et $y = a_2$
- Les règles du perceptron deviennent
  - si $w_0 + w_1 x + w_2 y < 0$ le perceptron retourne -1
  - si $w_0 + w_1 x + w_2 y \geqslant 0$ le perceptron retourne +1
- En réécrivant un peu ($a = -\frac{w_1}{w_2}$ et $b = -\frac{w_0}{w_2}$)
  - si $y < ax + b$ le perceptron retourne -1
  - si $y \geqslant ax + b$ le perceptron retourne +1

la droite $y = ax + b$ sépare l'espace en deux : tous les points (x,y) tels que

- $y < ax + b$ seront classés -1,
- $y \geqslant ax + b$ seront classé +1.



Règle de décision du perceptron – dimension n

- En dimension 3, on aurait quelque chose comme $z = ax + by + c$, et on a un plan qui sépare l'espace en trois dimensions en deux : tous les points "au dessus" ont la valeur $+1$, ceux "en dessous" ont la valeur -1.
- En dimension n, c'est plus compliqué à représenter, mais l'intuition est similaire : un "hyperplan" sépare l'espace en deux : d'un côté les $+1$, de l'autre les -1.
- Le perceptron représente des données "linéairement séparables"

Perceptron et apprentissage

Avec les poids fixés $w_0, w_1, \ldots, w_n$, le perceptron représente un hyperplan avec

- d'un côté des points qui ont pour valeur $+1$
- et de l'autre les points qui ont pour valeur -1

On veut faire de l'apprentissage !

On a des données, on veut les représenter par un perceptron :
➩ il va falloir trouver les poids $w_0, w_1, \ldots, w_n$ du perceptron

Perceptron et apprentissage

Avec les poids fixés $w_0, w_1, \ldots, w_n$, le perceptron représente un hyperplan avec

- d'un côté des points qui ont pour valeur $+1$
- et de l'autre les points qui ont pour valeur -1

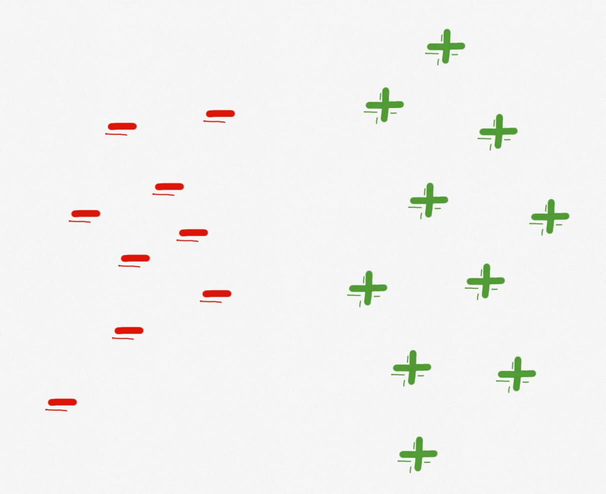
On veut faire de l'apprentissage !

On a des données, on veut les représenter par un perceptron :
⇨ il va falloir trouver les poids $w_0, w_1, \ldots, w_n$ du perceptron

- si les données sont linéairement séparables, on va pouvoir trouver plusieurs perceptrons qui représentent ces données ✔
- si les données ne sont pas linéairement séparables, la réponse n'est pas claire !
  - on peut dire qu'il y a un échec et s'arrêter !

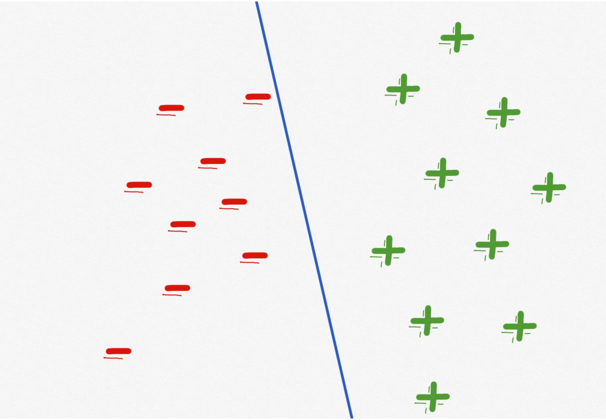
  ⇨ il faudra utiliser une autre méthode !
  - on peut essayer de trouver le perceptron qui fait le moins d'erreur !

Séparabilité linéaire des données



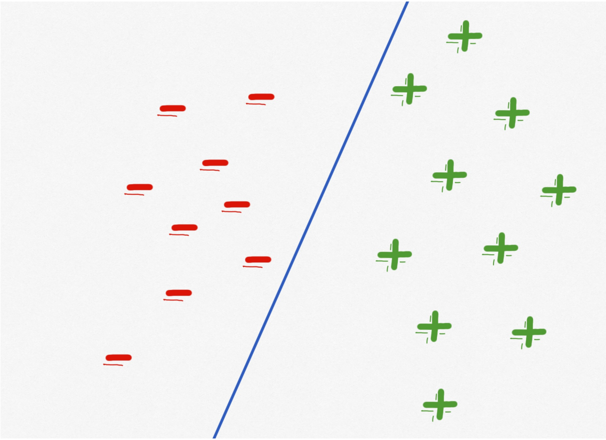
Peut-on séparer linéairement ces données ?

Séparabilité linéaire des données



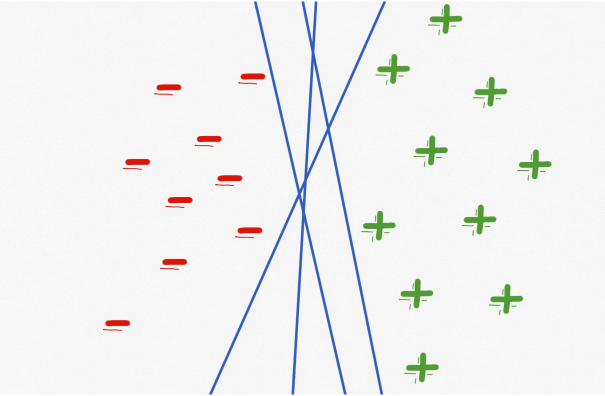
oui, un exemple.

Séparabilité linéaire des données



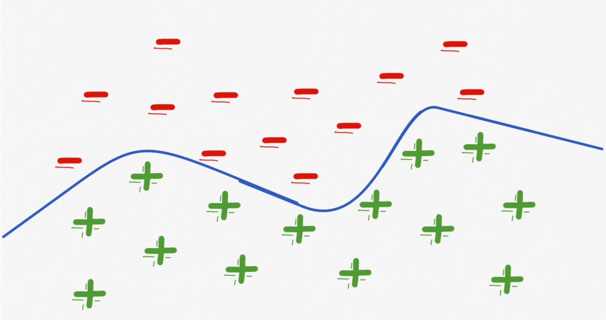
oui, un autre exemple !

Séparabilité linéaire des données



une infinité de perceptrons peuvent donc représenter ces données.

Séparabilité linéaire des données



ces données ne peuvent pas être séparées linéairement !

Apprentissage

But de l'apprentissage : trouver le vecteur $\overrightarrow{w}$ de poids $\overrightarrow{w} = \langle w_0, w_1, \ldots, w_n \rangle$ d'un perceptron qui représente les données.

idée :

- initialement, on tire au sort des poids aléatoires
- on prend chaque donnée une à une et on itère
  - si notre perceptron donne la bonne classification ➫ on ne change rien !
  - si notre perceptron se trompe ➫ on change les poids
  - ➫ on veut déplacer l'hyperplan de séparation pour corriger l'erreur
- si les données sont linéairement séparables, on n'aura plus de corrections à effectuer et on peut arrêter l'algorithme
- si les données ne sont pas linéairement séparables, l'algorithme ne va pas s'arrêter
  ➫ on peut mettre un nombre maximum d'itérations et décréter un échec.

Apprentissage - algorithme

// η hyperparamètre "pas d'apprentissage"
// perceptron initialisé avec le vecteur $\mathbf{w} = \langle \mathbf{w_0}, \mathbf{w_1}, \ldots, \mathbf{w_n} \rangle$ aléatoire
// on note f_{seuil} la fonction de seuil

tant que il y a des changements de poids
    On prend chaque point $\mathbf{x} = \langle \mathbf{x_1}, \ldots, \mathbf{x_n} \rangle$ des données
    la classe de $\mathbf{x}$ est $t_\mathbf{x}$
    on calcule la valeur $o_\mathbf{x}$ de $\mathbf{x}$ du perceptron $o_\mathbf{x} \leftarrow f_{seuil}\left(\sum_{i=0}^{n} w_i x_i\right)$
    // mise à jour des poids
    pour chaque attribut j
        $w_j \leftarrow w_j + \eta(t_\mathbf{x} - o_\mathbf{x})x_j$

- si le perceptron prédit la bonne classification, $t_\mathbf{x} - o_\mathbf{x} = 0$
  ➥ pas de changement de poids.
- sinon il y a un changement de poids : on pousse le poids vers la direction d'une classe ou l'autre selon l'erreur
  voir exemple page suivante
- le changement du poids est proportionnel à x_j

Apprentissage - algorithme

Règle de mise à jour des poids

$$w_j \leftarrow w_j + \eta(t_{\mathbf{x}} - o_{\mathbf{x}})x_j$$

- ex : $\eta = 1$, $x_j = 1$ et le perceptron **se trompe** et retourne **-1**.
  Donc $\eta(t_{\mathbf{x}} - o_{\mathbf{x}})x_j = 1 - (-1) = 2$: on augmente le poids de 2 pour donner plus de chance de dépasser le seuil
- ex : $\eta = 1$, $x_j = 1$ et le perceptron **se trompe** et retourne $+1$.
  Donc $\eta(t_{\mathbf{x}} - o_{\mathbf{x}})x_j = -1 - (1) = -2$: on réduit le poids de -2 pour éviter qu'il passe le seuil.
- le changement du poids est proportionnel à x_j :
  si $x_j = 2$, on augmenterait (ou réduirait) plus le poids.

Apprentissage - résultat

- un théorème nous **garantie** convergence **si** les données sont linéairement séparable !
- il n'y a pas convergence sinon !

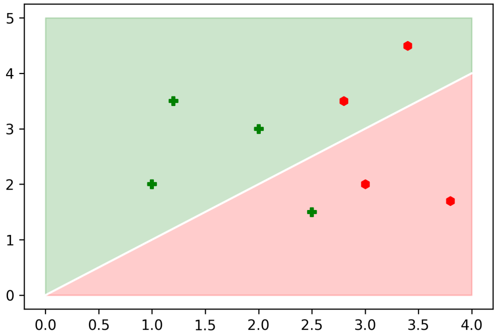
en pratique :

- la convergence est lente
- l'ordre dans lequel on utilise les données est important dans la vitesse de convergence !

Exemple simpl(issim)e

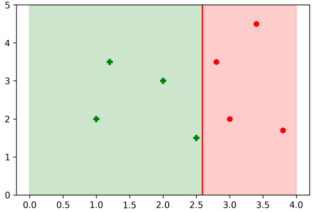
Problème initial :

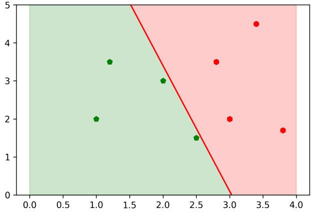
- 8 points en 2 dimensions
- poids initial $\langle 0, 1, -1 \rangle$ et $\eta = 0.1$



Exemple simpl(issim)e

Des solutions trouvées

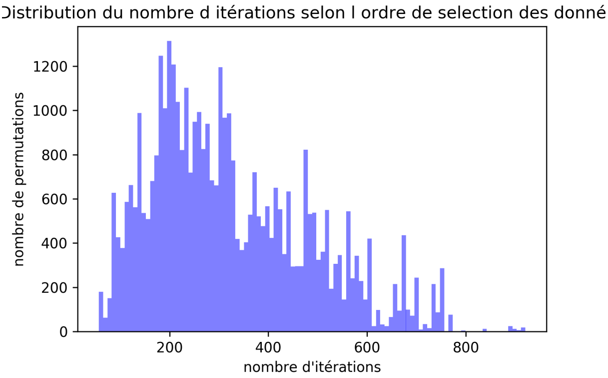




Exemple simpl(issim)e - ordre d'examen

On a essayé tous les ordres pour examiner les points

- pour chaque ordre, on compte le nombre d'itérations jusqu'à convergence
- on trace la distribution du nombre d'itérations
- minimum 57 itérations



Bilan du Perceptron

le perceptron peut représenter un problème

- de classification binaire
- avec des données linéairement séparables

problème :

- et si les données ne sont pas séparables ?
- peut-on l'utiliser pour de la regression (rappel : prédire une valeur) ?