Apprentissage Automatique

M1 IAFA-SECIL
Université Paul Sabatier

Contacts :
Sandrine.Mouysset@irit.fr
thomas.pellegrini@irit.fr

Apprentissage supervisé

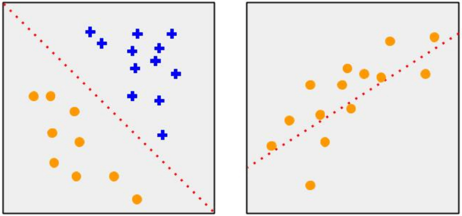
Chaîne d'analyse des données

Modèles supervisés :

On dispose d'une **base d'apprentissage**, sous ensemble de données "étiquetées" par des experts du type

X	X^1 ... X^i ... X^m	Classe
1 ⋮ i ⋮ n	Caractéristiques variables	S variables nominales

2 principaux types d'apprentissage supervisé :



Classification   Regression

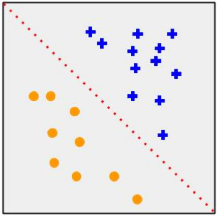
Classification : Assigner une catégorie à chaque observation :

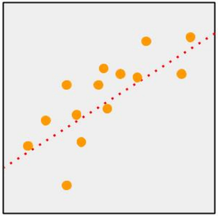
- Les catégories sont discrètes
- La cible est un indice de classe : $y \in \{0, .., K-1\}$
- *Exemple :* reconnaissance de chiffres manuscrits :
  - x : vecteur ou matrice des intensités des pixels de l'image
  - t : identité du chiffre

Régression : Prédire une valeur réelle à chaque observation :

- les catégories sont continues
- la cible est un nombre réel $y \in \mathbb{R}$
- *Exemple :* prédire le cours d'une action
  - x : vecteur contenant l'information sur l'activité économique
  - y : valeur de l'action le lendemain

2 principaux types d'apprentissage supervisé :


Classification


Regression

Approches par modèles supervisés :

- Arbres de Décision
- Apprentissage d'ensemble : Forêts aléatoires
- Classifieur linéaire
- Machine à vecteurs de support ou SVM
- Approche générative : classifieur bayésien (naïf)
- Réseaux de neurones

Méthodes d'évaluation :

- Validation croisée
- Matrice de confusion
- Precision, Rappel, F-mesure
- Courbe ROC

Classifieur bayésien naïf

Exemple : classifieur BN spam/non-spam

Problème : étant donné la séquence de mots d'un email, $W = w_1\ w_2\ \dots\ w_n$, prédire s'il s'agit d'un spam ou non.

Il faut estimer la probabilité suivante :

$$P(\text{spam}|W) = P(\text{spam}|w_1, w_2, \dots, w_n)$$

La règle de la chaîne donne :

$$\begin{aligned} P(\text{spam}|w_1, w_2, \dots, w_n) &= P(\text{spam})P(w_1|\text{spam})P(w_2|\text{spam}, w_1)\dots \\ &\quad \dots P(w_n|\text{spam}, w_1, w_2, \dots, w_n) \end{aligned}$$

Très compliqué à estimer... Nous faisons une simplification en considérant l'hypothèse Bayesienne Naïve !

$$
\begin{aligned}
P(\text{spam}|w_1, w_2, \ldots, w_n) \quad &= \quad P(\text{spam})P(w_1|\text{spam})P(w_2|\text{spam}, w_1)\ldots \\
&\quad\quad \ldots P(w_n|\text{spam}, w_1, w_2, \ldots, w_n)
\end{aligned}
$$

Simplification : on suppose que pour tout j :

$$P(w_j|\text{spam}, w_1, \ldots, w_{j-1}) = P(w_j|\text{spam})$$

Autrement dit, l'apparition d'un mot dans un email ne dépend pas des autres mots présents dans ce mail mais uniquement de sa classe spam ou non-spam.

On obtient :

$$P(\text{spam}|W) \quad = \quad P(\text{spam})P(w_1|\text{spam})P(w_2|\text{spam})\ldots P(w_n|\text{spam})$$

On obtient :

$$P(\mathrm{spam}|W) \quad = \quad P(\mathrm{spam})P(w_1|\mathrm{spam})P(w_2|\mathrm{spam})\dots P(w_n|\mathrm{spam})$$

En réalité on obtient un score avec cette formule qu'il faut normaliser pour obtenir une probabilité :

$$P(\mathrm{spam}|W) \quad = \quad \frac{1}{Z}P(\mathrm{spam})P(w_1|\mathrm{spam})P(w_2|\mathrm{spam})\dots P(w_n|\mathrm{spam})$$

Avec Z facteur de normalisation :

$$Z = P(W) = P(W|\mathrm{spam})P(\mathrm{spam}) + P(W|\mathrm{non-spam})P(\mathrm{non-spam})$$

	Spam	Non-spam
"acheter"	60%	10%
"promotion"	50%	30%
Proba a priori	10%	90%

Que vaut la probabilité qu'un email contenant les mots "acheter" et "promotion" soit un spam ?

	Spam	Non-spam
"acheter"	60%	10%
"promotion"	50%	30%
Proba a priori	10%	90%

Que vaut la probabilité qu'un email contenant les mots "acheter" et "promotion" soit un spam ?

score(spam)=0.1*0.6*0.5=0.03
score(non-spam)=0.9*0.1*0.3=0.027
Z = 0.03+0.027=0.057
P(spam|"acheter", "promotion") = 1/0.057 * 0.03 = 52.6%

Que se passe-t-il quand un mot est nouveau dans un email de test, par exemple "biscuit" ?

	Spam	Non-spam
"acheter"	60%	10%
"promotion"	50%	30%
"biscuit"	0%	0%
Proba a priori	10%	90%

Que se passe-t-il quand un mot est nouveau dans un email de test, par exemple "biscuit" ?

	Spam	Non-spam
"acheter"	60%	10%
"promotion"	50%	30%
"biscuit"	0%	0%
Proba a priori	10%	90%

Il faut utiliser des "pseudo-comptes" de mots en lissant les comptes : on ajoute +1 à tous les mots par exemple

Un classifieur bayésien **naïf** est :

- un classifieur linéaire,
- qui repose sur le théorème de Bayes,
- et sur l'hypothèse d'indépendance statistique des attributs :

$$\begin{aligned} p(x_1, x_2, \ldots, x_d|y) &= p(x_1|y)p(x_2|y)\ldots p(x_d|y) \\ &= \prod_{i=1}^{d} p(x_i|y) \end{aligned}$$

Un classifieur BN peut identifier une pomme en fonction de sa couleur et sa forme, en considérant la couleur et la forme d'un fruit de manière indépendante (pas d'interactions entre les deux).
$\rightarrow$ Chaque distribution peut être estimée indépendamment comme une distribution unidimensionnelle

Résumé classification bayésienne naïve

Pour faire des prédictions la règle de classification du BN est :

$$\hat{y} = \arg\max_y p(y) \prod_{i=1}^{d} p(x_i|y)$$

Avantages :

- En général performant même avec peu d'exemples d'apprentissage
- Rapide à entraîner

Implémentations disponibles sur sklearn :

- Gaussien
- Bernoulli
- Multinomial,
- etc.

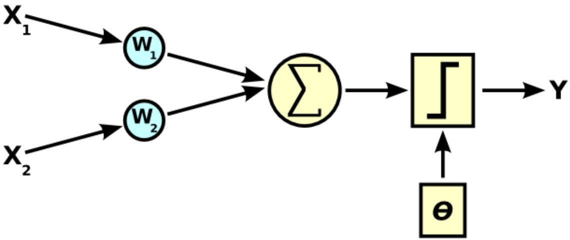
`https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes`

Perceptron, intro réseaux de neurones

- Proposé par Frank Rosenblatt vers 1957
- Algorithme de classification linéaire
- On suppose que l'on a un problème à deux classes (c_-, c_+) avec m exemples d'apprentissage :

$$\mathcal{D} = \{(x^j, y^j), j = 1 \ldots m\}, \text{avec } x^j \in \mathbb{R}^d, y^j \in \{-1, +1\}$$

- Hypothèse : les deux classes sont linéairement séparables par un hyperplan

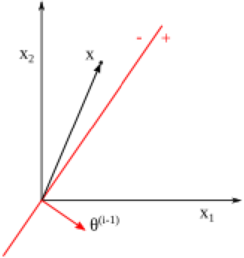


Algorithme perceptron online $(\mathcal{D}, y \in \{-1, +1\})$

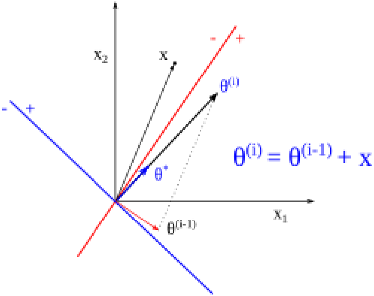
1: Initialiser $\boldsymbol{\theta} \leftarrow 0$
2: TANT QUE pas convergence FAIRE
3: POUR j de 1 à m FAIRE
4: SI $y^j \, \boldsymbol{\theta}^t \mathrm{x}^j \leq 0$ ALORS
5: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^j \, \mathrm{x}^j$

Illustration 2D d'une itération de la règle du perceptron

- x $\in c_+$ mal classé à l'itération $i-1$

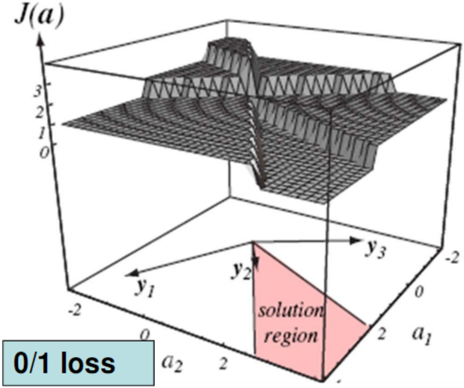


- x $\in c_+$ bien classé à l'itération i



- Choix naturel : nombre d'exemples mal classés : la perte 0/1 ou $\mathcal{L}_{0/1}$
- Problème : $\mathcal{L}_{0/1}$, fonction de θ, est constante par morceaux avec des discontinuités lorsque la frontière de décision passe par dessus des exemples

  $\rightarrow$ descente de gradient pas applicable avec cette fonction



- Alternative : on veut que tous les exemples x^j satisfassent $y^j \theta^t x^j > 0$, en utilisant le fait que $y^j \in \{-1, +1\}$
- Pour un exemple mal classé, on aura $y^j \theta^t x^j < 0$ et on cherchera donc à augmenter cette valeur pour la rendre positive, ce qui est équivalent à vouloir minimiser l'opposé : $-y^j \theta^t x^j$

$$J(\theta) = -\sum_{j \in \mathcal{M}} y^j \boldsymbol{\theta}^t x^j$$

où $\mathcal{M}$ dénote l'ensemble des points mal-classés. J est linéaire par morceaux en θ : linéaire dans les régions de l'espace de θ où un exemple est mal-classé, et vaut zéro dans les régions où un exemple est bien classé.

$$J(\theta) = -\sum_{j \in \mathcal{M}} y^j \, \boldsymbol{\theta}^t \mathrm{x}^j$$

Autre façon d'écrire cette perte (version modifiée de la perte Hinge) :

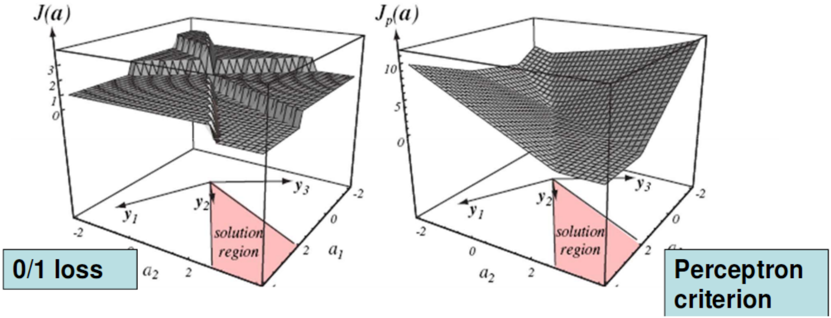
$$J(\theta) = \sum_{j=1}^{m} \max(0, -y^j \, \boldsymbol{\theta}^t \mathrm{x}^j)$$

- Le terme $\max(0, -y^j \, \boldsymbol{\theta}^t \mathrm{x}^j)$ vaut 0 si l'exemple est bien classé et sinon il est égal à la "confiance" (score) lorsqu'il est mal classé

Perceptron : fonction de perte

$$J(\theta) = \sum_{j=1}^{m} \max(0, -y^j\, \boldsymbol{\theta}^t x^j)$$

- Cette fonction de perte est linéaire en θ dans les régions où un exemple est mal-classé et vaut 0 ailleurs $\rightarrow$ OK !



- Pour l'exemple d'apprentissage d'indice j :

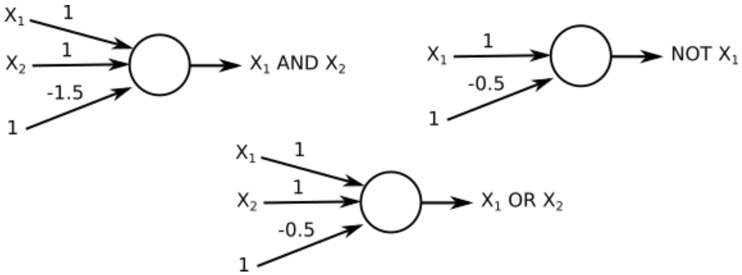
$$J^j(\theta) = \max(0, -y^j\, \boldsymbol{\theta}^t \mathrm{x}^j)$$

- Gradient:

$$\nabla J^j(\theta) = \begin{cases} 0 & \text{si } y^j\, \boldsymbol{\theta}^t \mathrm{x}^j > 0 \text{ (prédiction correcte)} \\ -y^j\, \mathrm{x}^j & \text{ sinon (prédiction incorrecte)} \end{cases}$$

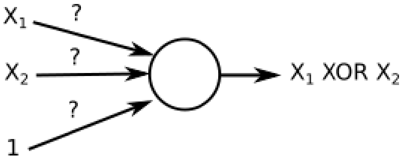
- Si les données ne sont pas linéairement séparables, la frontière de décision va osciller indéfiniment et l'algorithme ne convergera jamais
- L'algorithme ne dit pas si les données sont séparables ou non
- L'hyperplan trouvé n'est pas unique : il y a un cône de solutions
- L'hyperplan trouvé a tendance à trop "coller" aux données $\rightarrow$ ajouter une marge...

Exemples de fonctions binaires que l'on peut modéliser par un perceptron



(erreur sur le "NOT" : multiplier par -1 les poids)

Et la fonction XOR (ou exclusif) ?

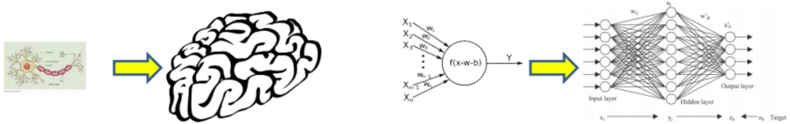


$\Rightarrow$ Pas de solution avec le perceptron !

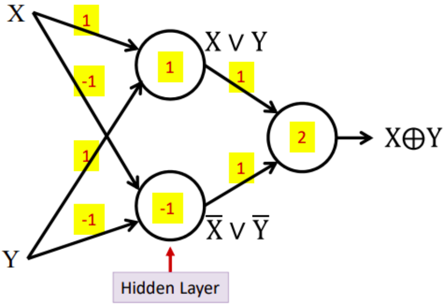
‘

- Minsky and Papert, 1968

Un seul neurone n'est pas suffisant

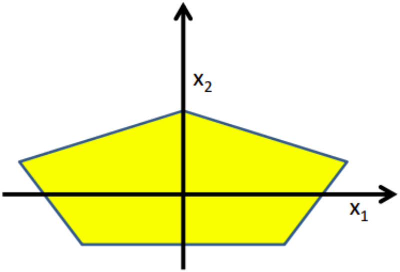


- Minsky and Papert, 1969, *Perceptrons: An Introduction to Computational Geometry*
  - Un neurone seul est un élément faible
  - Il faut des neurones interconnectés ⇒ "Réseau de neurones"

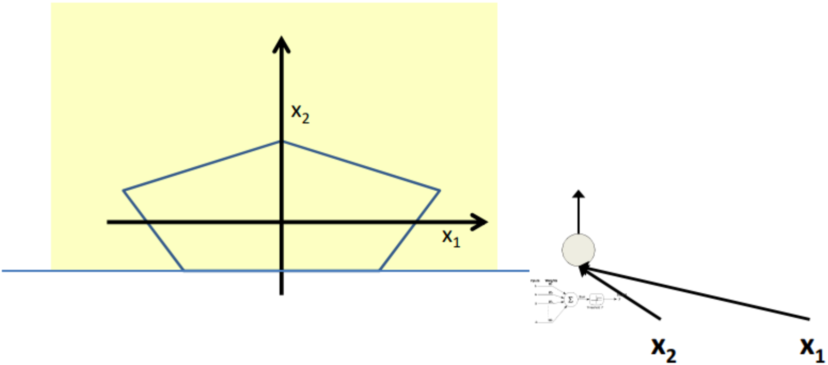


- XOR
  - La première couche est une couche "cachée"
  - Architecture suggérée dans Minsky and Papert, 1968

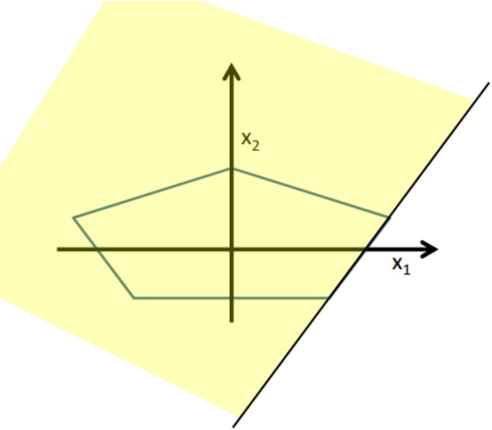
Composer des frontières de décision complexes

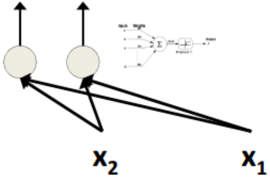


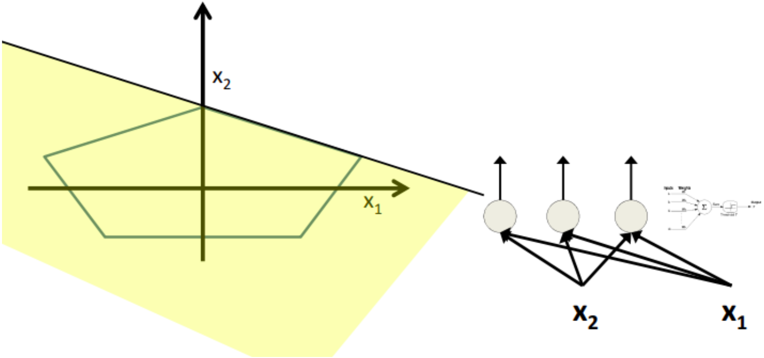
Objectif : construire un réseau de neurones avec un unique neurone de sortie qui "s'active" uniquement pour les points de la région jaune

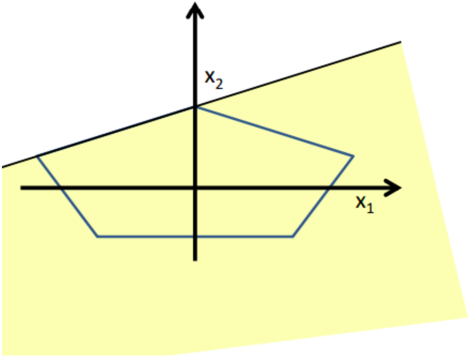


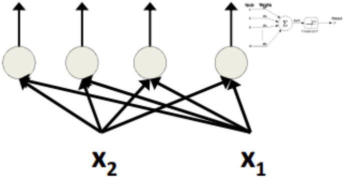
x_2
x_1
x_2
x_1

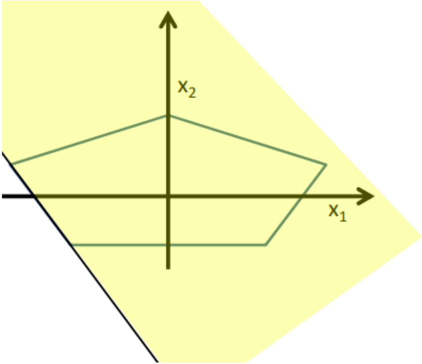

x2
x1

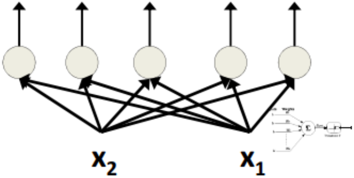

x2
x1

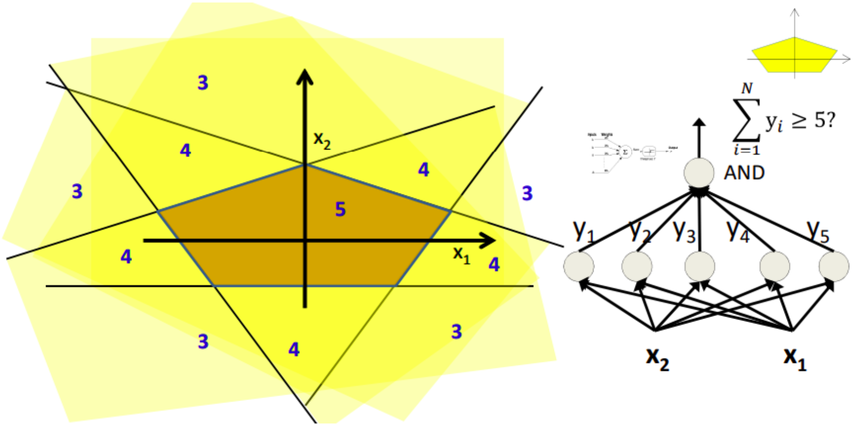

x_2
x_1
x_2
x_1


x2
x1

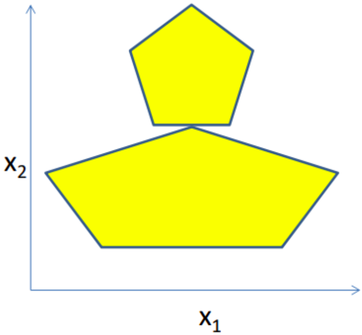

x2
x1

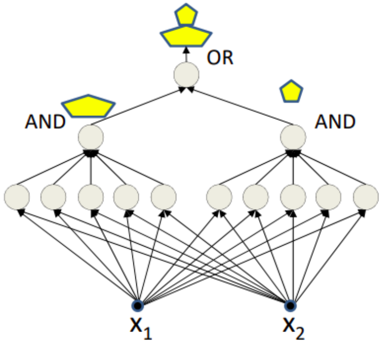

x_2
x_1


x_2
x_1


3
4
3
4
5
4
3
4
4
3
4
3
x2
x1
$\sum_{i=1}^{N} y_i \geq 5?$
AND
y1
y2
y3
y4
y5
x2
x1

Deux polygones ?

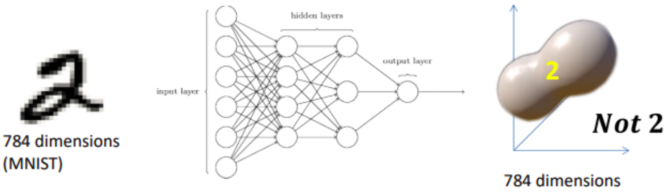




- Le réseau ne doit s'activer que pour la région jaune
- Deux polygones

  ⇒ Un neurone "OR"

  ⇒ Trois couches : 2 couches cachées, 1 couche de sortie



- Problèmes de classification dans la vie réelle : trouver des frontières de décision dans des espaces à grande dimension
  - Faisable par un MLP
  - Un MLP peut prendre en entrée un vecteur de valeurs réelles et donner un probabilité d'appartenance à une ou plusieurs classes

Une seule couche cachée et une couche de sortie à un unique neurone :