Introduction à l'Apprentissage Automatique

Cours 2 - le perceptron

Thomas Pellegrini, équipe SAMoVA, IRIT,
thomas.pellegrini@irit.fr

IRIT - UPS

Les fondamentaux : le perceptron

- Frank Rosenblatt, inventeur de la ”solution à tout”, le ”perceptron”, en 1958

Les fondamentaux : le perceptron

- Frank Rosenblatt, inventeur de la ”solution à tout”, le ”perceptron”, en 1958
- Algorithme de classification inspiré d’un modèle très simplifié du cerveau humain

Le perceptron

- On suppose que l'on a un problème à deux classes (c_-, c_+) avec m exemples d'apprentissage :

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

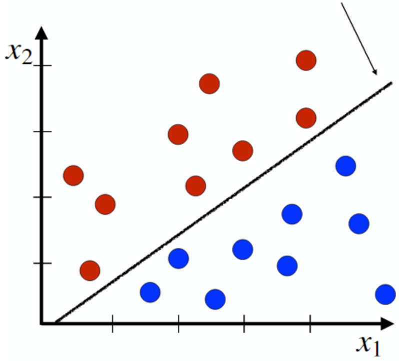
- Hypothèse : les deux classes sont linéairement séparables par un hyperplan, par ex. une droite si on est en 2-d

Le perceptron

- On suppose que l'on a un problème à deux classes (c_-, c_+) avec m exemples d'apprentissage :

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

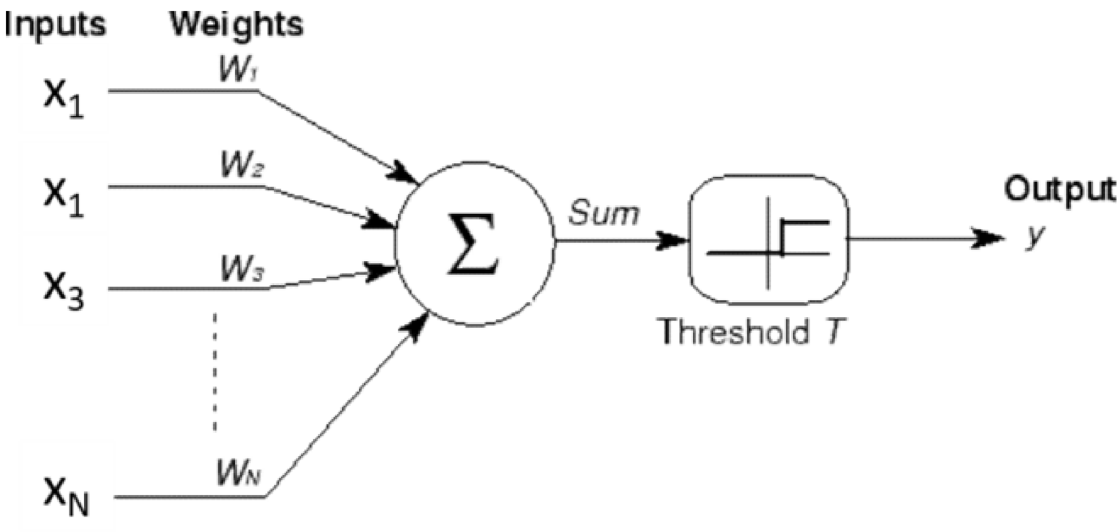
- Hypothèse : les deux classes sont linéairement séparables par un hyperplan, par ex. une droite si on est en 2-d



Prédiction :

$$y(x) = \hat{y} = \begin{cases} +1 & \text{si } w_1 x_1 + w_2 x_2 > T \\ -1 & \text{sinon} \end{cases}$$

$$\text{Prédiction} : y(x) = \hat{y} = \begin{cases} +1 & \text{si } w_1 x_1 + w_2 x_2 > T \\ -1 & \text{sinon} \end{cases}$$



- Les paramètres sont combinés linéairement
- $\Rightarrow$ perceptron = "classifieur linéaire"
- le "neurone" est activé si la somme dépasse un seuil T

On pose $w_0 = -T$ et $z = w_0 \times 1 + w_1 x_1 + w_2 x_2$

Les prédictions deviennent : $\hat{y} = \begin{cases} +1 & \text{si } z > 0 \\ -1 & \text{sinon} \end{cases}$

Écriture "vectorielle" :

$\mathbf{w} =$

$\mathbf{x} =$

$z =$

Comment trouver les valeurs des poids w et du seuil T ?

Algorithme du perceptron :

1. On initialise **w** à $\vec{0}$
2. On fait des prédictions sur les points
3. On modifie **w**, **uniquement pour les points mal classés**
4. On recommence à l'item 2 jusqu'à convergence

Comment trouver les valeurs des poids w et du seuil T ?

Algorithme du perceptron :

1. On initialise **w** à $\vec{0}$
2. On fait des prédictions sur les points
3. On modifie **w**, **uniquement pour les points mal classés**
4. On recommence à l'item 2 jusqu'à convergence

$\Rightarrow$ Quelle règle pour actualiser **w** ?

Comment trouver les valeurs des poids w et du seuil T ?

Algorithme du perceptron :

1. On initialise **w** à $\vec{0}$
2. On fait des prédictions sur les points
3. On modifie **w**, **uniquement pour les points mal classés**
4. On recommence à l'item 2 jusqu'à convergence

$\Rightarrow$ Quelle règle pour actualiser **w** ?

$$\mathbf{w} = \begin{cases} \mathbf{w} + \mathbf{x} & \text{si } y = 1 \text{ et } \hat{y} = -1 \\ \mathbf{w} - \mathbf{x} & \text{si } y = -1 \text{ et } \hat{y} = 1 \end{cases}$$

Soit

$$\boxed{\mathbf{w} = \mathbf{w} + y\mathbf{x} \text{ si } \mathbf{x} \text{ mal classé}}$$

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

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

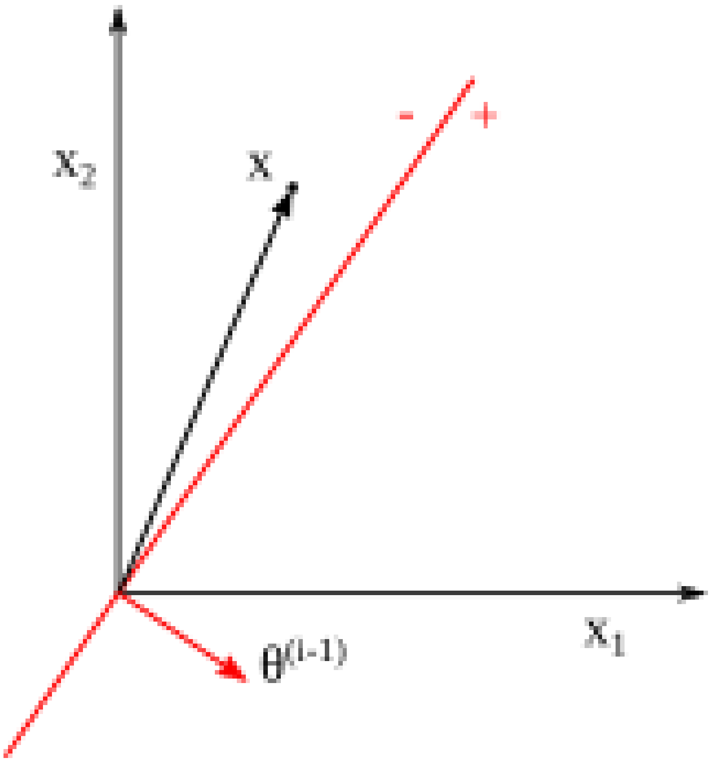
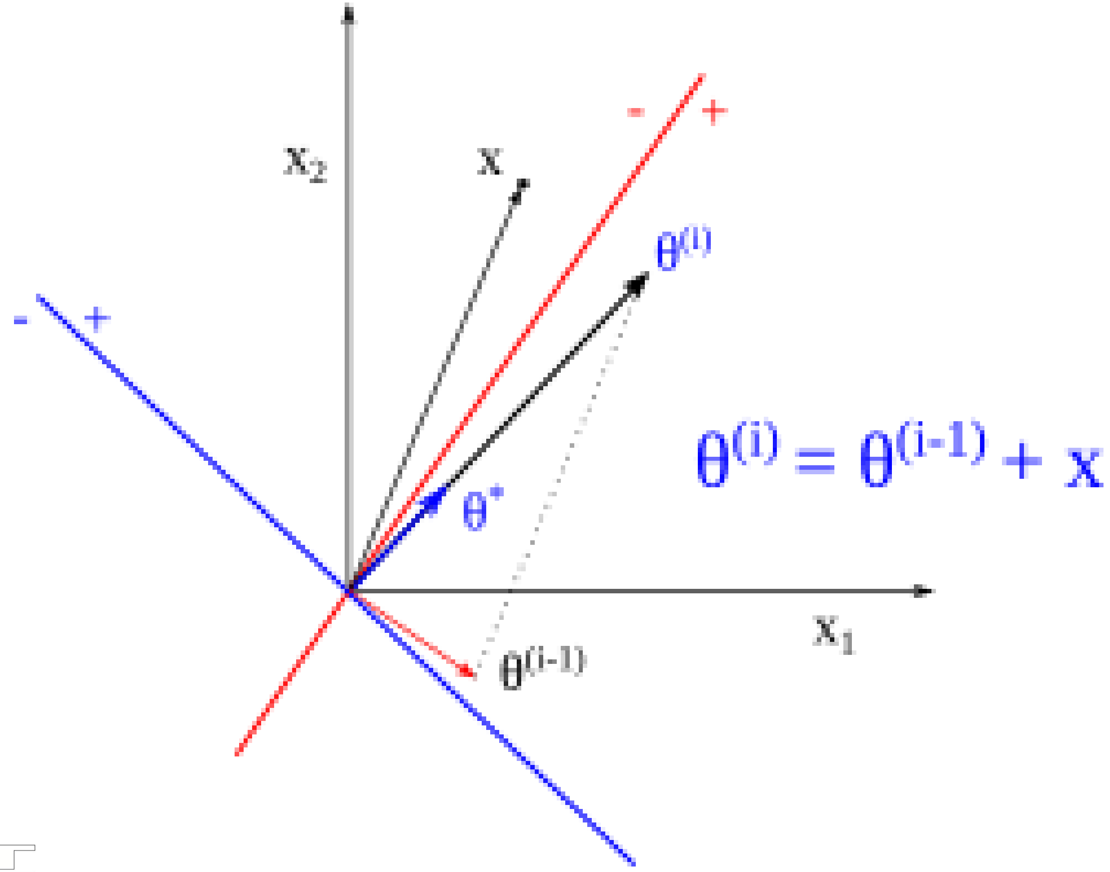


Illustration 2D d'une itération de la règle du perceptron

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



Variantes équivalentes de la règle

- $\mathbf{w} = \mathbf{w} + y\mathbf{x}$ si $\mathbf{x}$ mal classé

- Avec η appelé "pas d'apprentissage" ou *learning rate* :

$\mathbf{w} = \mathbf{w} + \eta y\mathbf{x}$ si $\mathbf{x}$ mal classé

- On peut enlever le test conditionnel :

$\mathbf{w} = \mathbf{w} + \eta(y - \hat{y})\mathbf{x}$

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

Initialiser $\mathbf{w} \leftarrow \mathbf{0}$
TANT QUE pas convergence FAIRE
  POUR j de 1 à m FAIRE
    SI $y^j \, \mathbf{w}^t \mathbf{x}^j \leq 0$ ALORS
      $\mathbf{w} \leftarrow \mathbf{w} + y^j \, \mathbf{x}^j$

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

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

- Apprentissage online : le modèle est actualisé à chaque exemple vu
- Apprentissage batch : le modèle est actualisé après avoir vu le corpus entier

Version apprentissage par "batch"

- "Batch" en anglais : "jeu de plusieurs exemples de données"

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

Initialiser $\mathbf{w} \leftarrow \mathbf{0}$

FAIRE

  Initialiser $\boldsymbol{\delta} \leftarrow \mathbf{0}$

  POUR j de 1 à m FAIRE

    $s^j \leftarrow \mathbf{w}^t \mathbf{x}^j$

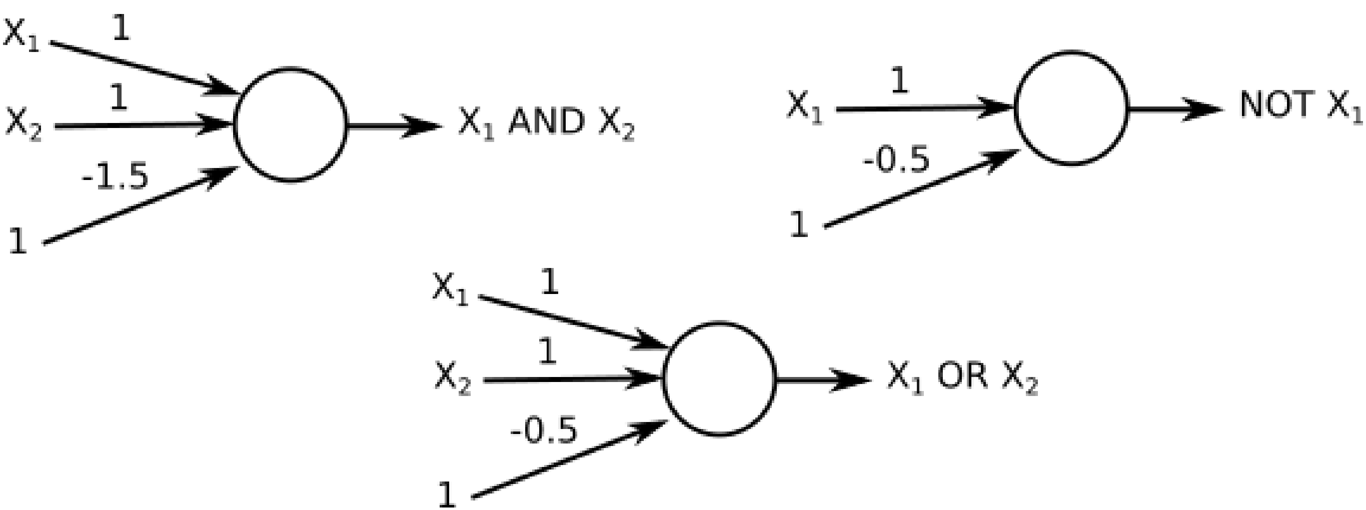
    SI $y^j s^j \leq 0$ ALORS

      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + y^j \mathbf{x}^j$

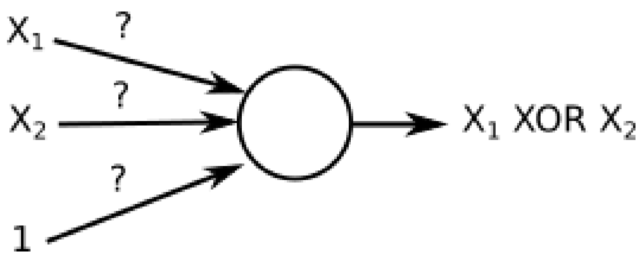
  $\mathbf{w} \leftarrow \mathbf{w} + \delta$

TANT QUE $|\delta| > \epsilon$

Facile de simuler des fonctions booléennes...



Sauf... La fonction XOR !

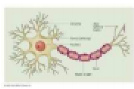


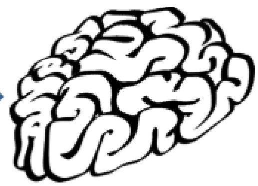
$\Rightarrow$ Pas de solution avec le perceptron !

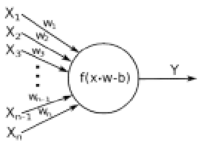
‘

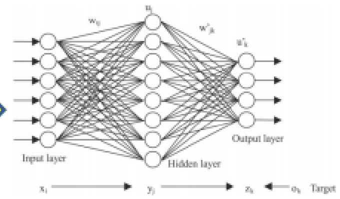
- Minsky and Papert, 1968

Un seul neurone n'est pas suffisant



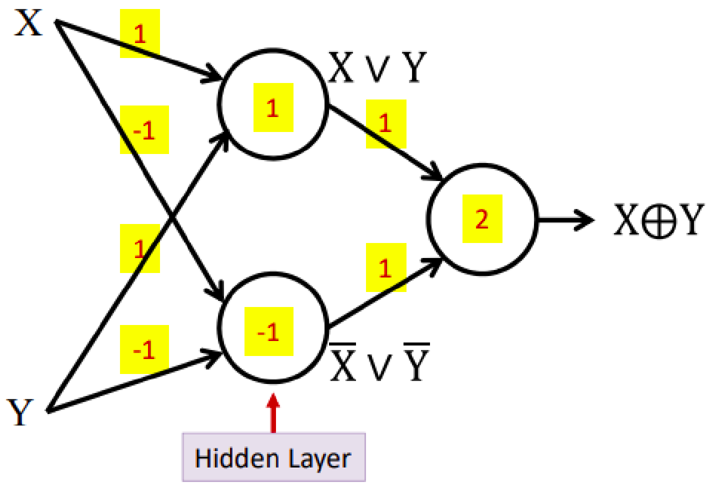






- Minsky and Papert, 1969, *Perceptrons : An Introduction to Computational Geometry*
  - Un neurone seul est un élément faible
  - Il faut des neurones interconnectés $\Rightarrow$ "Réseau de neurones"

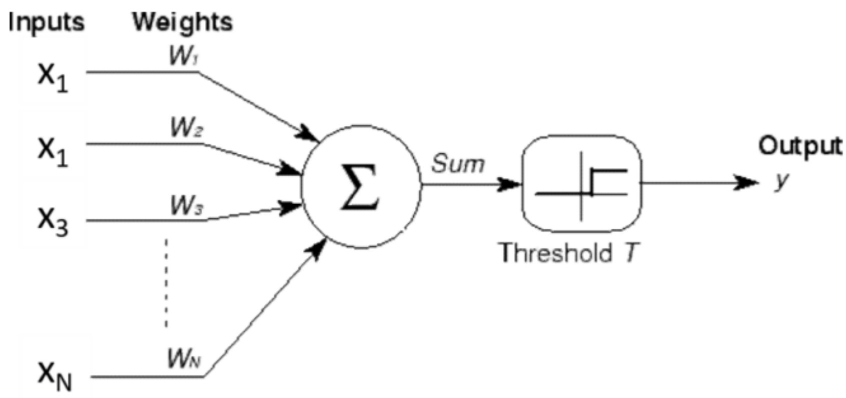
Perceptron Multi-Couche, Multi-Layer Perceptron (MLP)



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

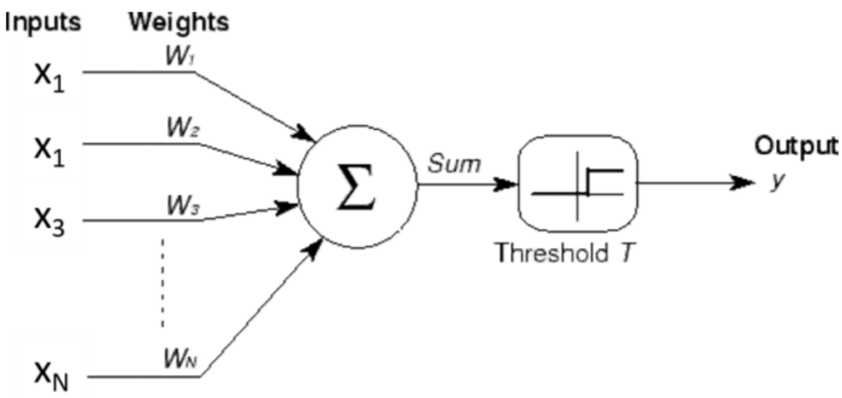
Perceptron à valeurs réelles

- Prédiction : $y(x) = \hat{y} \in \{-1, +1\}$
- Le neurone a une fonction d'activation **"tout-ou-rien"**



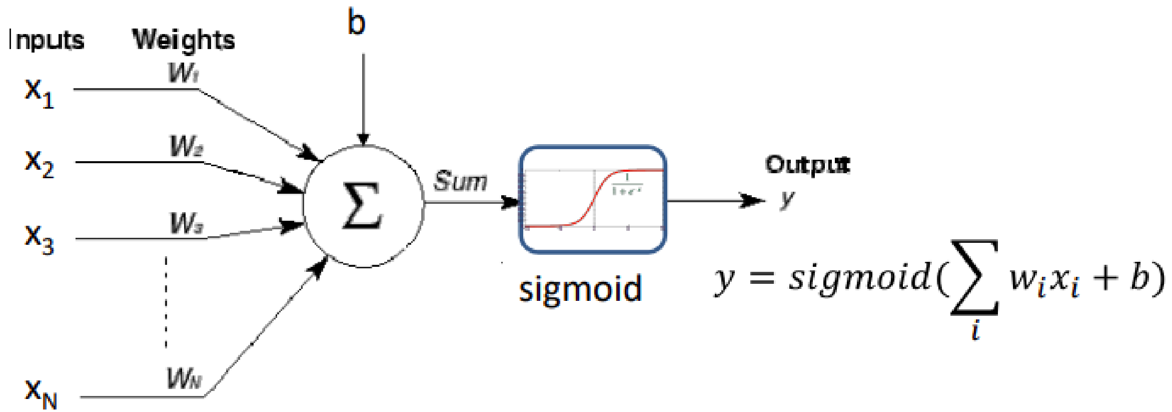
Perceptron à valeurs réelles

- Prédiction : $y(x) = \hat{y} \in \{-1, +1\}$
- Le neurone a une fonction d'activation **"tout-ou-rien"**



- Que faire si on veut $\hat{y}$ réel ? Par exemple on veut $\hat{y} \in [0, 1]$ pour avoir une probabilité ?

Fonction d'activation "**tout-ou-rien**" $\Rightarrow$ remplacée par la fonction "**sigmoide**" mais beaucoup d'autres fonctions d'activation existent...

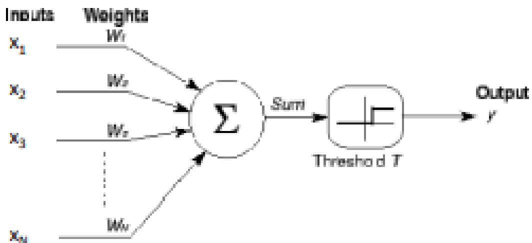


$$f(x) = \frac{1}{1 + e^{-x}}$$

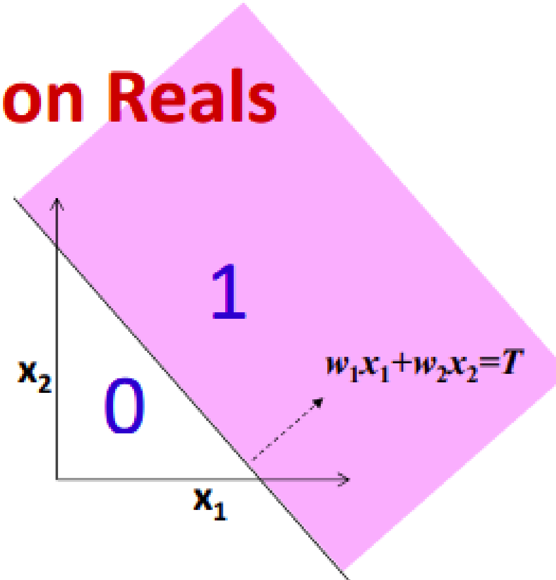
Exercice :

- montrer que $f'(x) = (1 - f(x))f(x)$
- monter qu'elle a un point d'inflexion en $x = 0$

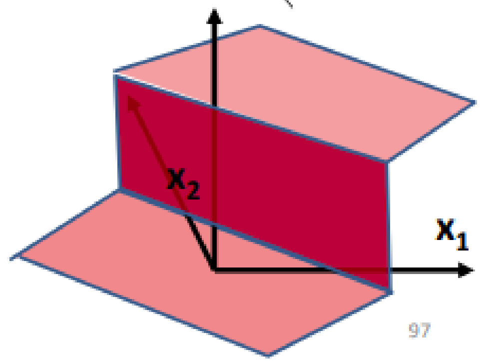
A Perceptron on Reals



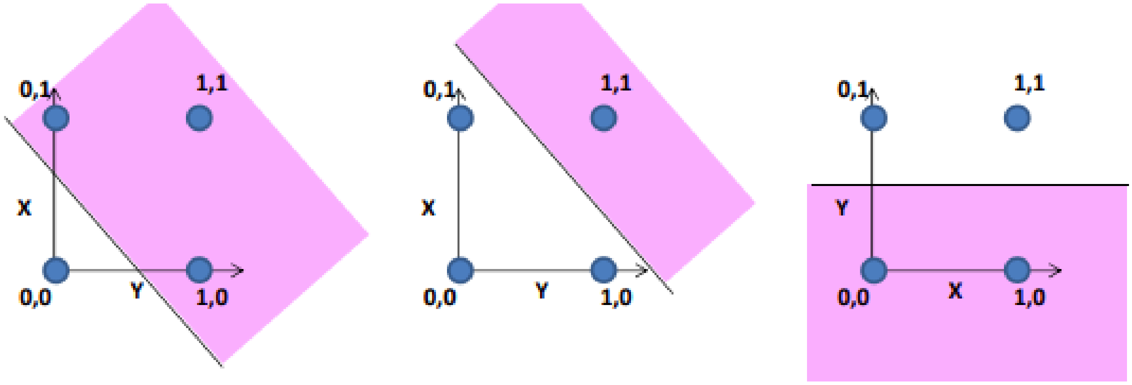
$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$



- A perceptron operates on *real*-valued vectors
  - **This is a *linear classifier***

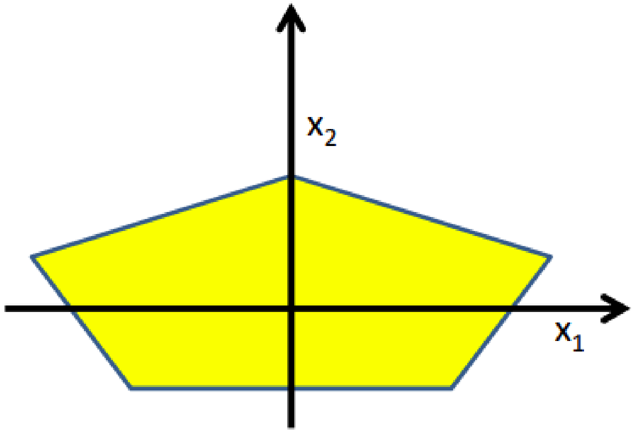


Slide par Bhiksha Raj

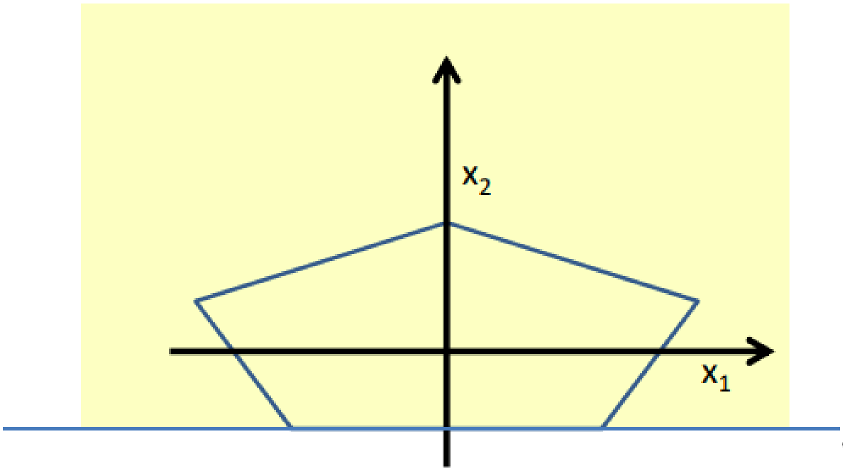


- Dans les régions colorées, la sortie du perceptron est $+1$
- En ajoutant un neurone, on ajoute une région colorée
- On peut obtenir des frontières de décision très complexes

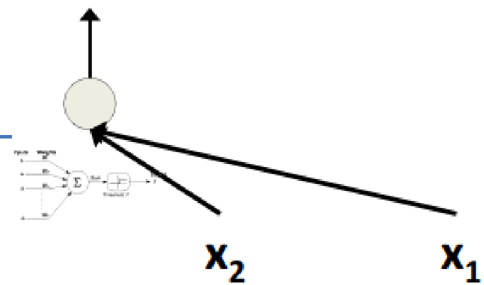
Composer des frontières de décision complexes



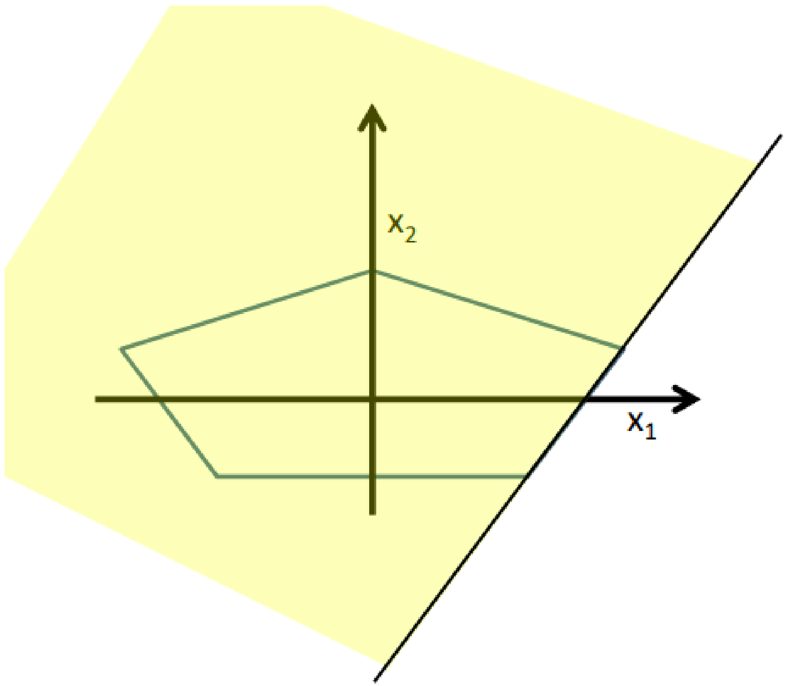
Objectif : construire un réseau de neurones avec un unique neurone de sortie qui "s'active" uniquement pour les points de la région jaune

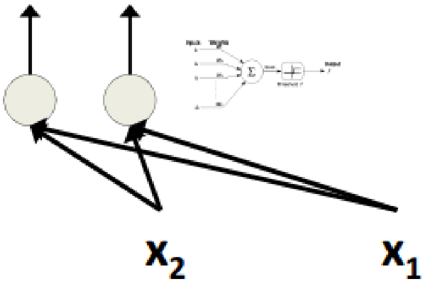


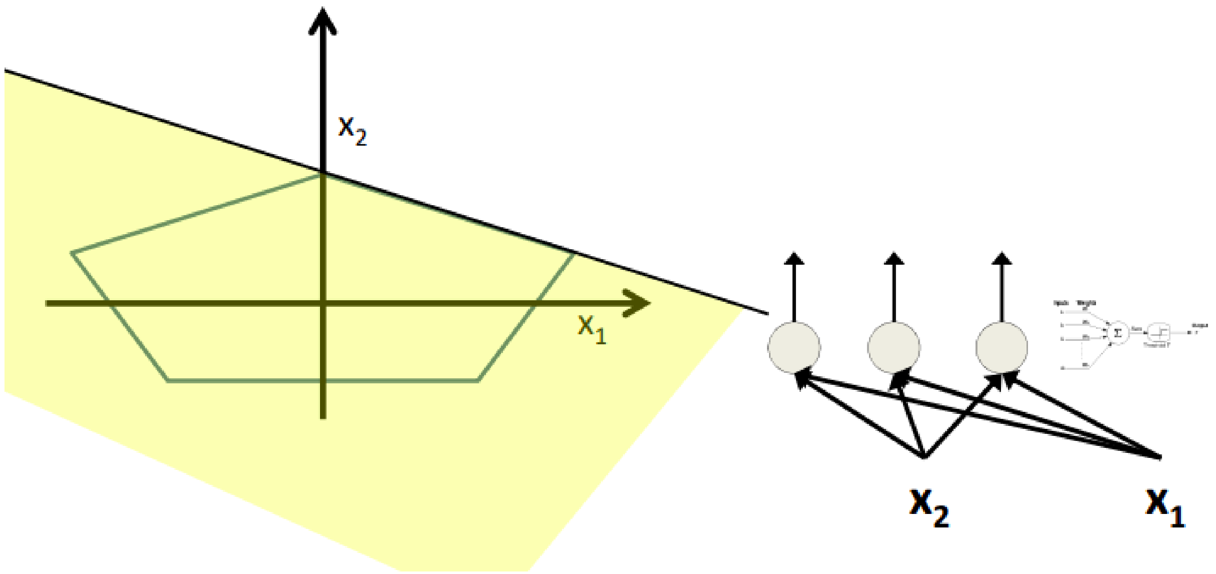
x_2
x_1

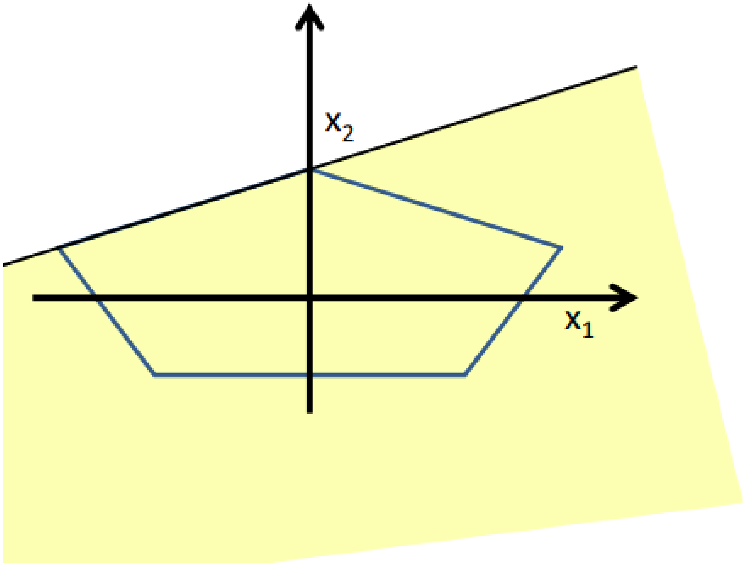


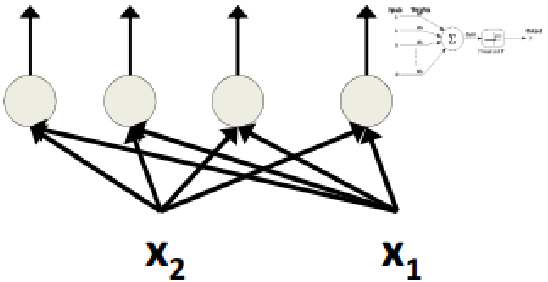
x_2
x_1

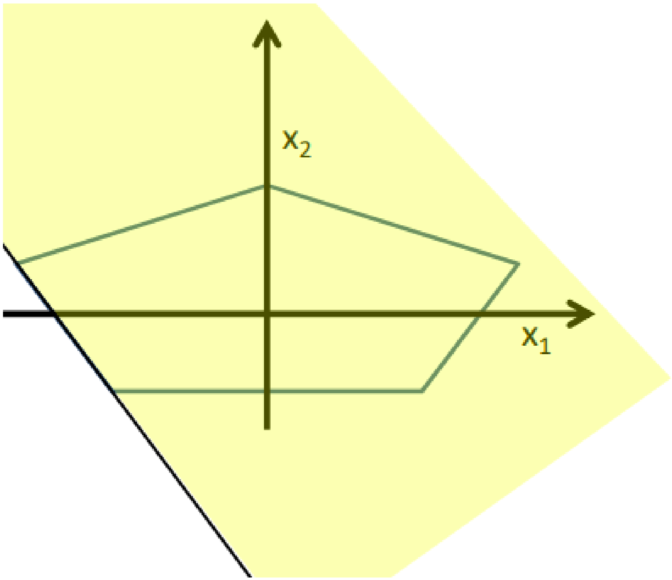

x_2
x_1


x_2
x_1

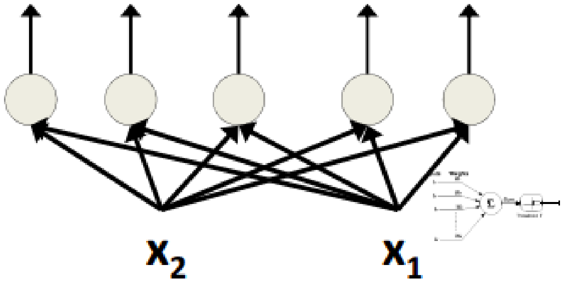

x_2
x_1
x_2
x_1


x_2
x_1

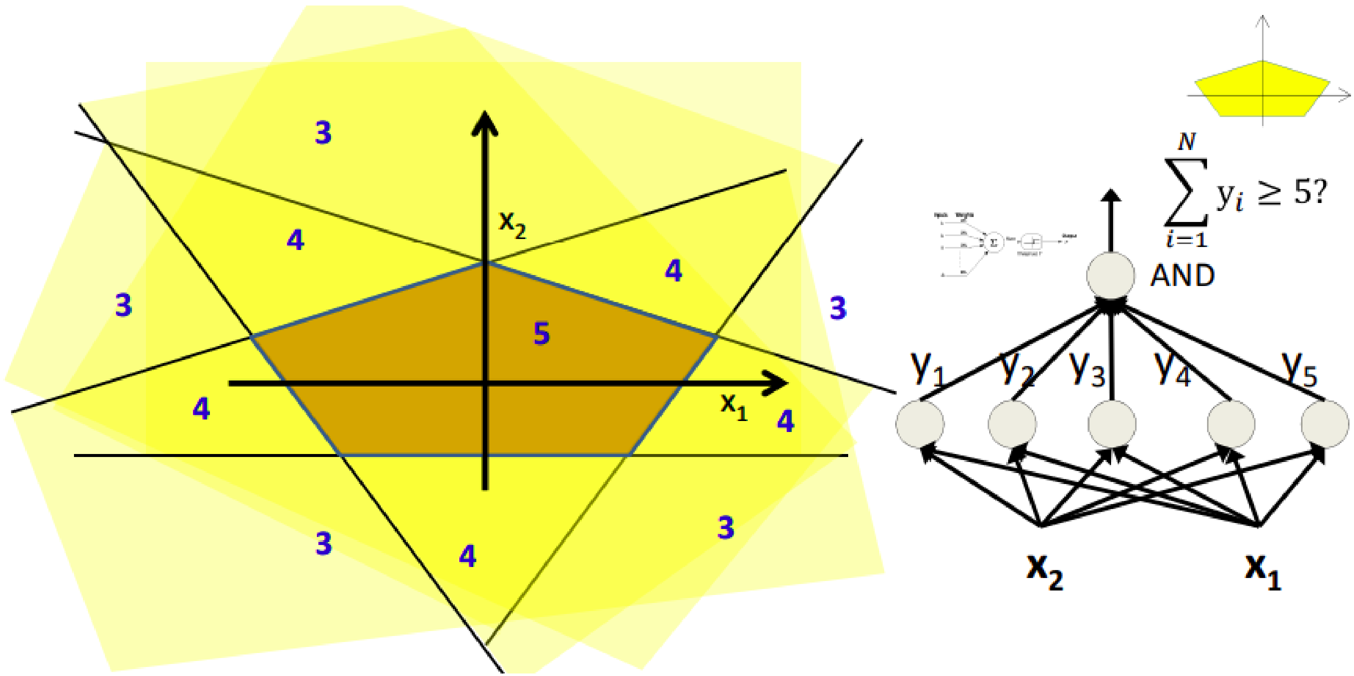

x_2
x_1



x_2
x_1

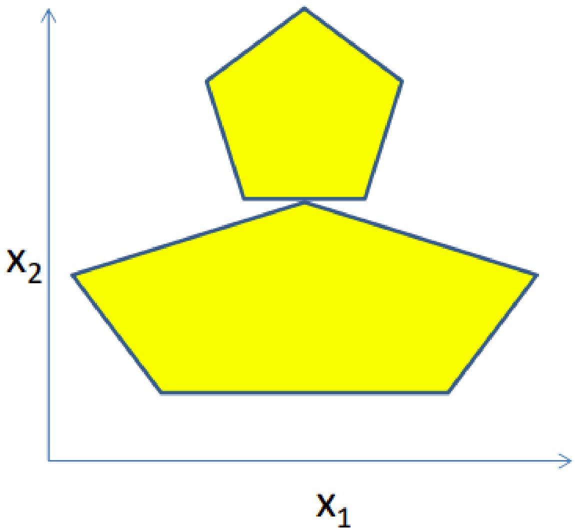


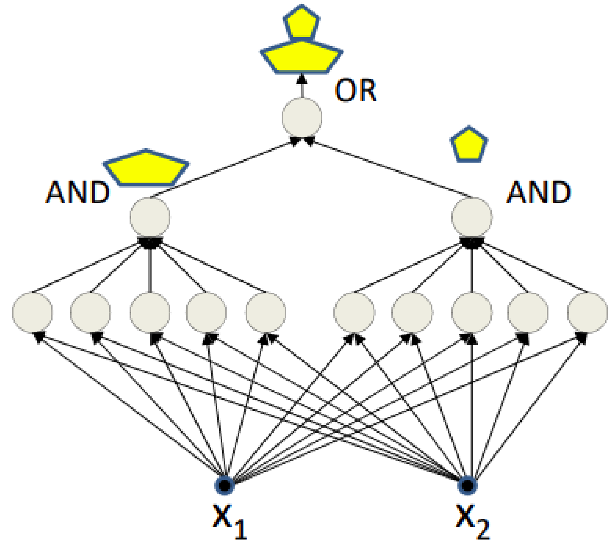
x_2
x_1



3
4
3
4
5
4
3
4
4
3
4
3
x_2
x_1
$\sum_{i=1}^{N} y_i \geq 5?$
AND
y_1
y_2
y_3
y_4
y_5
x_2
x_1

Deux polygones ?



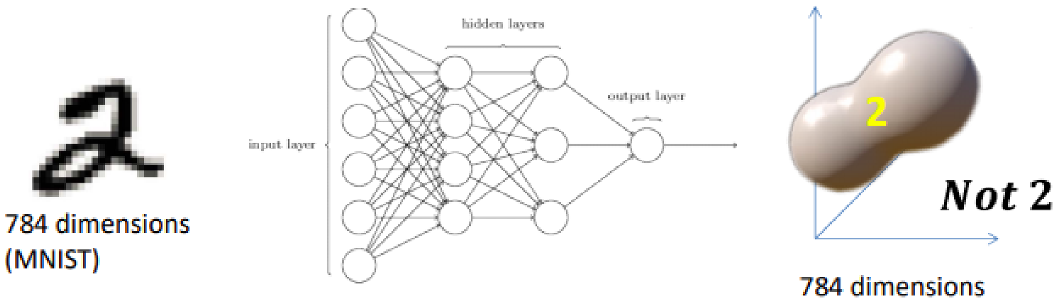


- Le réseau ne doit s’activer que pour la région jaune
- Deux polygones

  ⇒ Un neurone ”OR”

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

Frontières de décision complexes



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