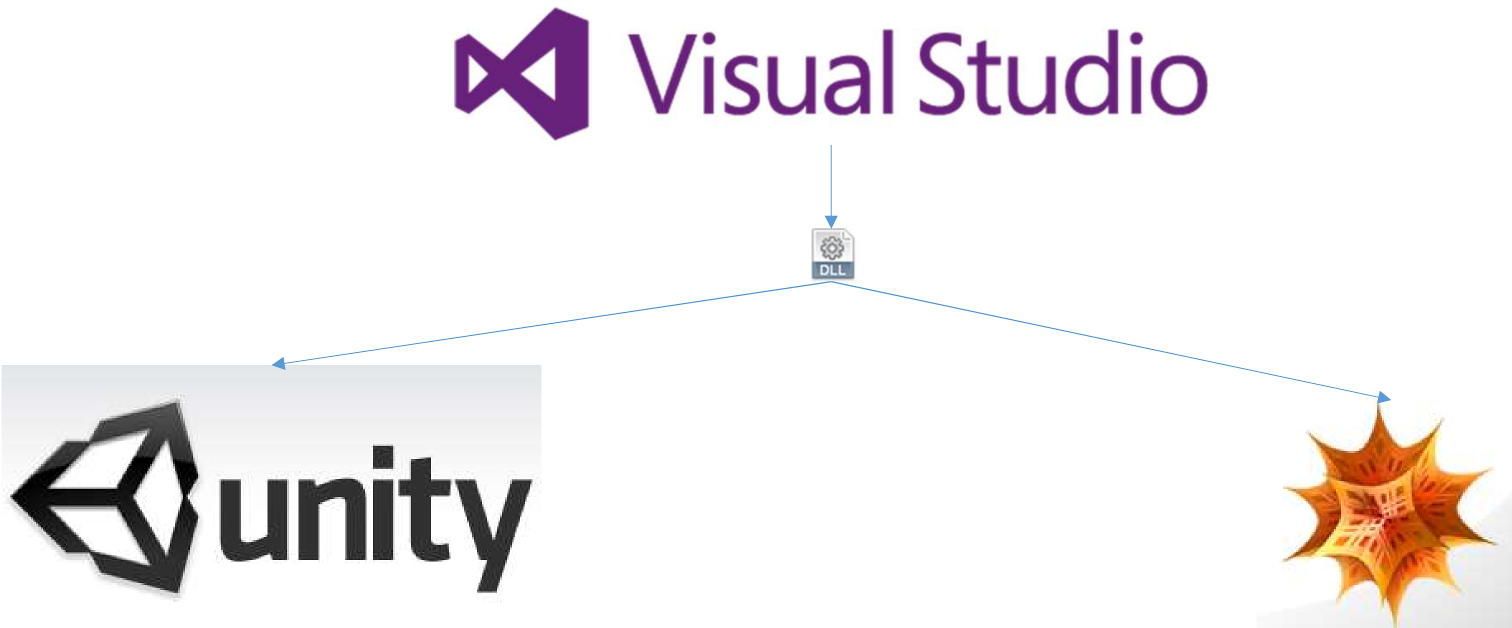
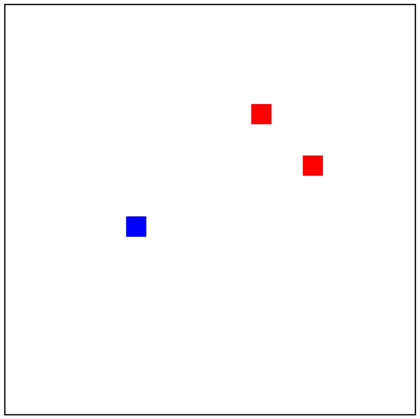
Le projet !

1. Mettre en place le pipeline de développement (exemple)

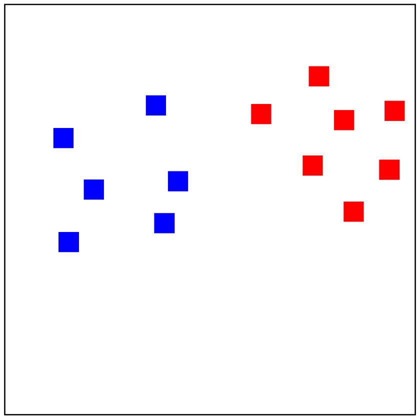


Le projet !

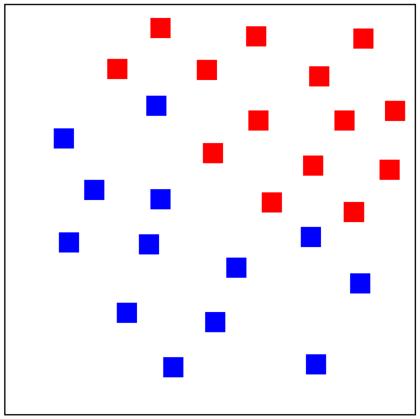
2. Création des cas de tests



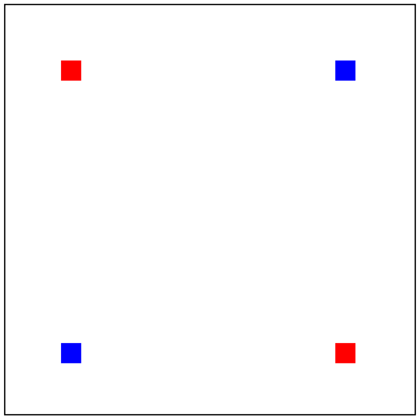
Simple, linéairement séparable



Réel, linéairement séparable



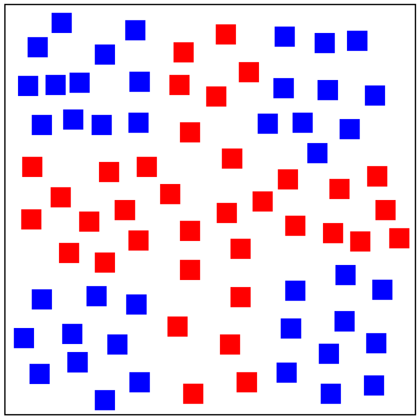
Soft, non linéairement séparable



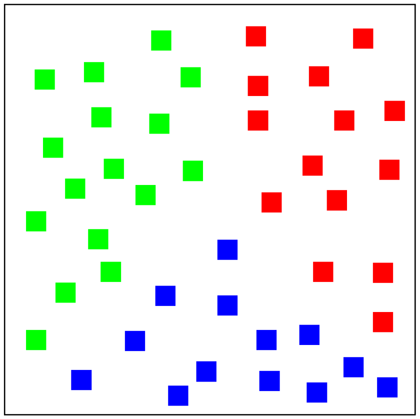
XOR

Le projet !

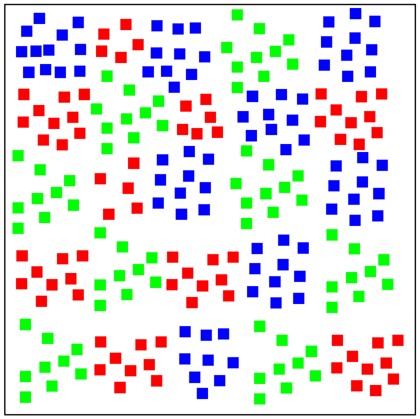
2. Création des cas de tests



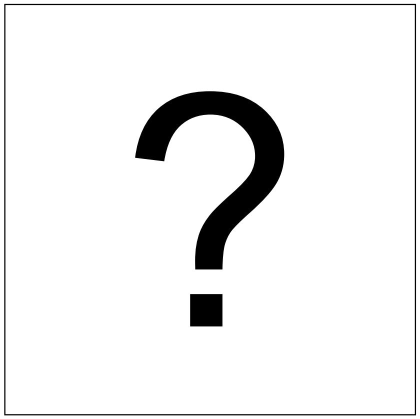
Cross



Multi Class Soft



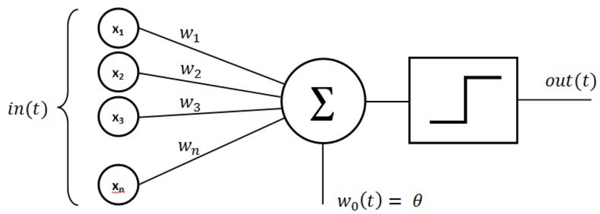
Multi Class Hard



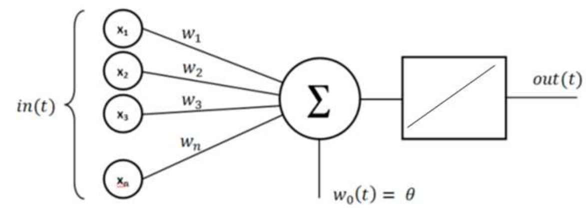
Real Dataset

Le projet !

3. Implémentation des algorithmes
    1. Modèles linéaires



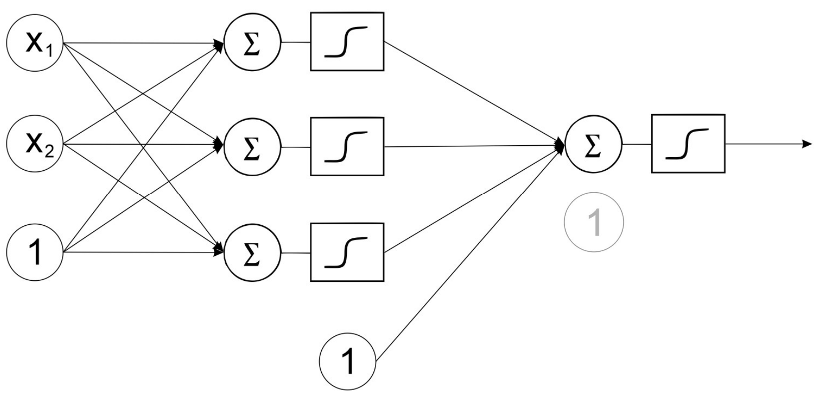
Perceptron pour la Classification



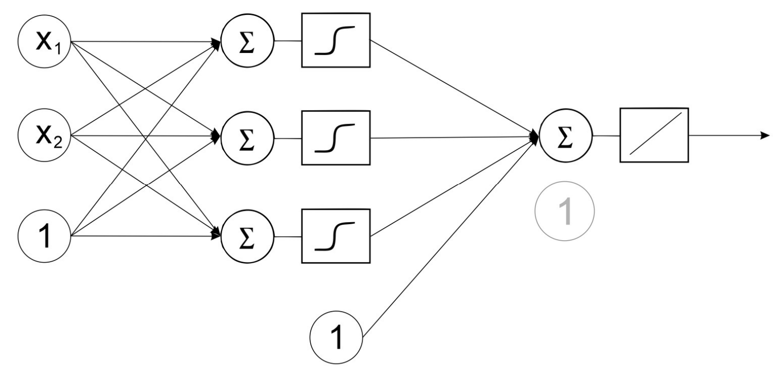
Perceptron pour la Régression

Le projet !

3. Implémentation des algorithmes
    2. Perceptron Multi Couches



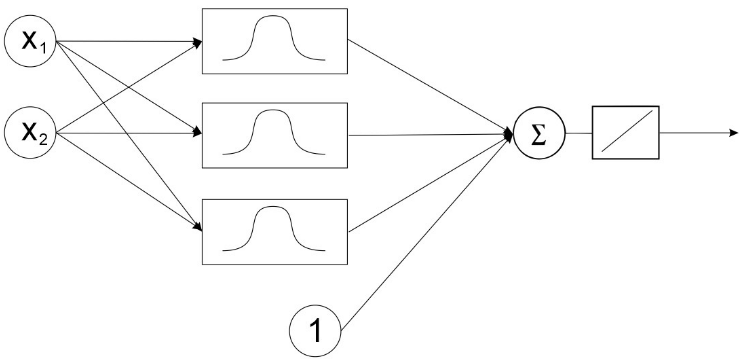
Perceptron multi couches pour la classification



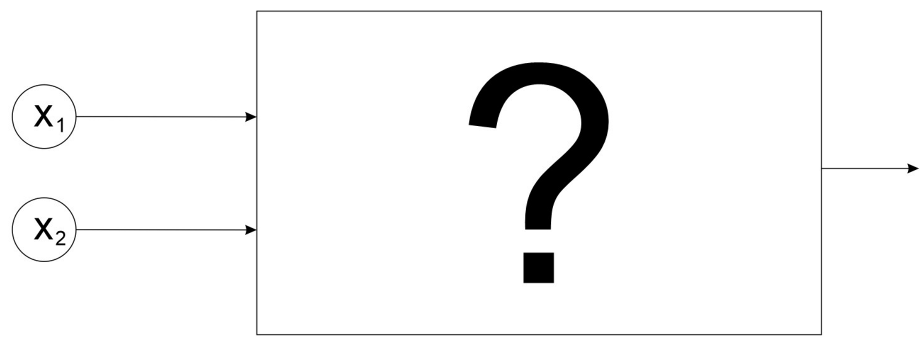
Perceptron multi couches pour la régression

Le projet !

3. Implémentation des algorithmes
   3. Modèles non linéaires



RBF (s)



Autre

Le projet !

4. Application a un dataset réel
   1. S'attaquer au dataset mystère ou trouver un dataset réel (ne soyez pas trop ambitieux ! ☺)
      - https://archive.ics.uci.edu/ml/datasets/
      - http://grouplens.org/datasets/movielens
      - http://yann.lecun.com/exdb/mnist/
      - https://www.kaggle.com/
      - https://toolbox.google.com/datasetsearch
      - … ?
   2. Etablir un protocole de test
   3. Entrainement du/des modèles
   4. Présentation et analyse des résultats

Le projet !

- Modalités pratiques
  - Groupes de 4 Max
    - Répartition des tâches est à éviter pour l'implémentation (surtout concernant le PMC)
  - Soutenance : 30 minutes (20 présentation + 10 questions)
  - Amusez-vous !

Qu’est-ce qu’apprendre ?

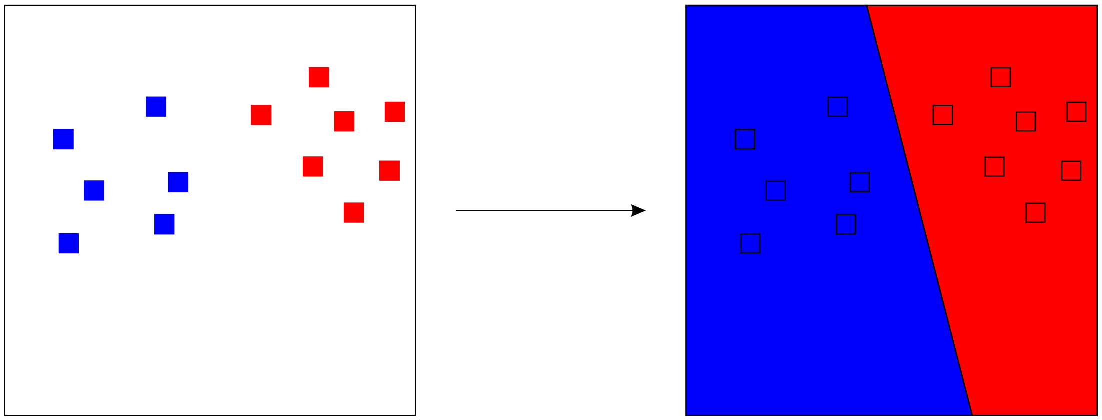
Qu’est-ce qu’apprendre ?

Intuition :

Découvrir (ou estimer) une fonction (ou une distribution) inconnue à partir d’un ensemble d’exemples

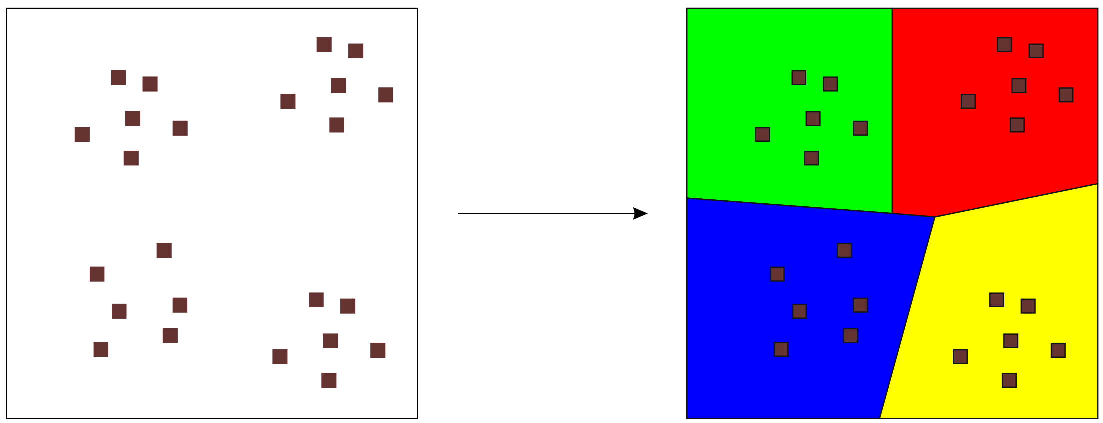
Qu’est-ce qu’apprendre ?

Apprentissage supervisé :



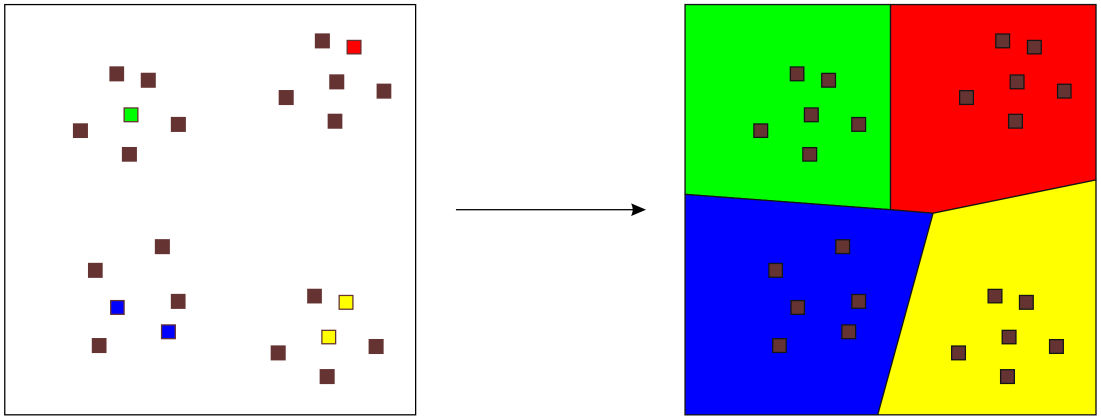
Qu'est-ce qu'apprendre ?

Apprentissage non supervisé :



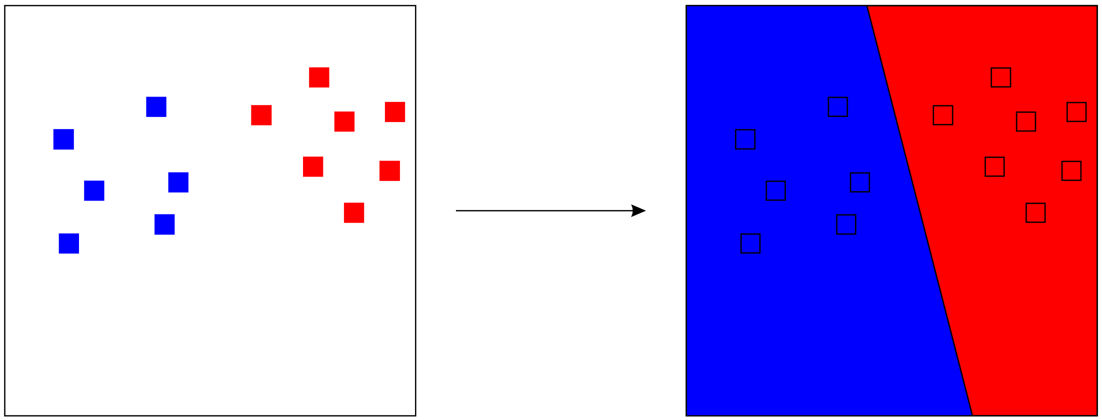
Qu’est-ce qu’apprendre ?

Apprentissage semi supervisé :



Qu'est-ce qu'apprendre ?

Apprentissage supervisé :



Apprentissage supervisé

Apprentissage supervisé

Apprendre …

Apprentissage supervisé

Apprendre …

- Apprendre par cœur ?

Apprentissage supervisé

Apprendre …

- Exemples (Input => Output)
  - {1 , 2} => {3}
  - {4 , 2} => {6}
  - {2 , 2} => {4}
  - {8 ,13} => {21}

Apprentissage supervisé

Apprendre …

- Exemples (Input => Output)
  - {1 , 2} => {3}
  - {4 , 2} => {6}
  - {2 , 2} => {4}
  - {8 ,13} => {21}
- Apprendre par cœur ?
  - Dictionnaire ?

Apprentissage supervisé

Apprendre …

- Exemples (Input => Output)
  - {1 , 2} => {3}
  - {4 , 2} => {6}
  - {2 , 2} => {4}
  - {8 ,13} => {21}
- Apprendre par cœur ?
  - Dictionnaire ?
  - Aucune information sur le reste de l'espace d'entrée !

Apprentissage supervisé

Apprendre …

… n’a d’intérêt que si on généralise !

Apprentissage supervisé

Qu’est-ce que généraliser ?

Apprentissage supervisé

⇒Généraliser :

⇒Supposer qu'il existe une **<u>fonction cible</u>** qui a généré les exemples que nous avons à disposition.

⇒Essayer d'**<u>approximer</u>** les résultats de cette fonction cible à l'aide d'un modèle.

⇒*Espérer* ☺ que si on approxime « *bien* » les résultats donnés sur les exemples étiquetés, on approximera « *bien* » sur l'ensemble de l'espace d'entrée

Apprentissage supervisé

⇒Généraliser :

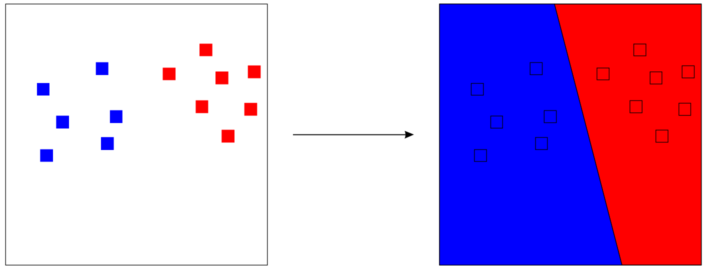
- ⇒Supposer qu'il existe une **<u>fonction cible</u>** qui a généré les exemples que nous avons à disposition.

- ⇒Essayer d'**<u>approximer</u>** les résultats de cette fonction cible à l'aide d'un modèle.

- ⇒*Espérer* ☺ que si on approxime « *bien* » les résultats donnés sur les exemples étiquetés, on approximera « *bien* » sur l'ensemble de l'espace d'entrée

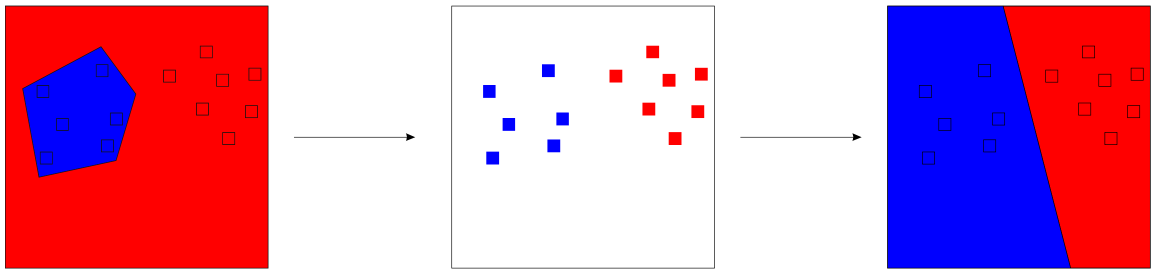
Apprentissage supervisé

$\Rightarrow$Contre exemple abstrait:



Apprentissage supervisé

⇒Contre exemple abstrait:



Apprentissage supervisé

⇒Contre exemple de l'arnaque à la prédiction

Apprentissage supervisé

Arnaque à la prédiction :

Apprentissage supervisé

Arnaque à la prédiction :

Apprentissage supervisé

Arnaque à la prédiction :

Apprentissage supervisé

Arnaque à la prédiction :

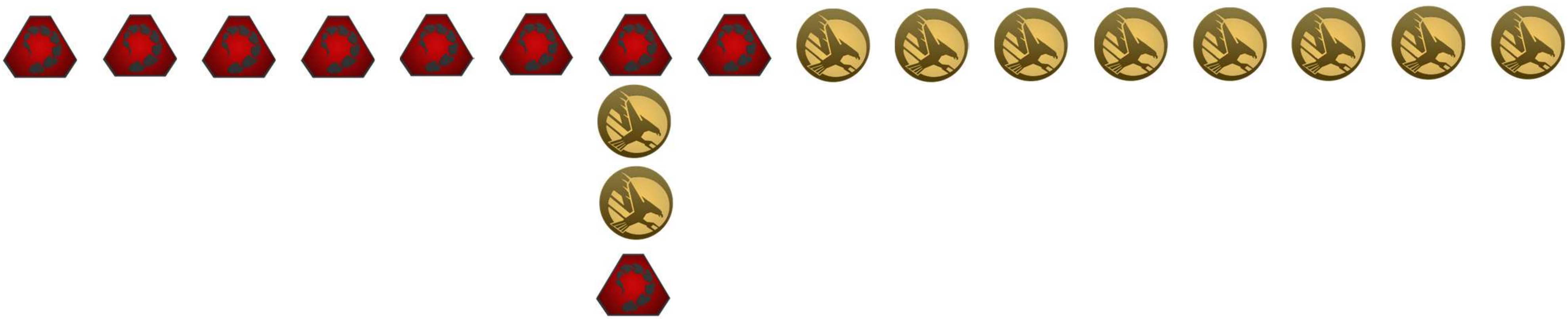
Apprentissage supervisé

Arnaque à la prédiction :

?

Apprentissage supervisé

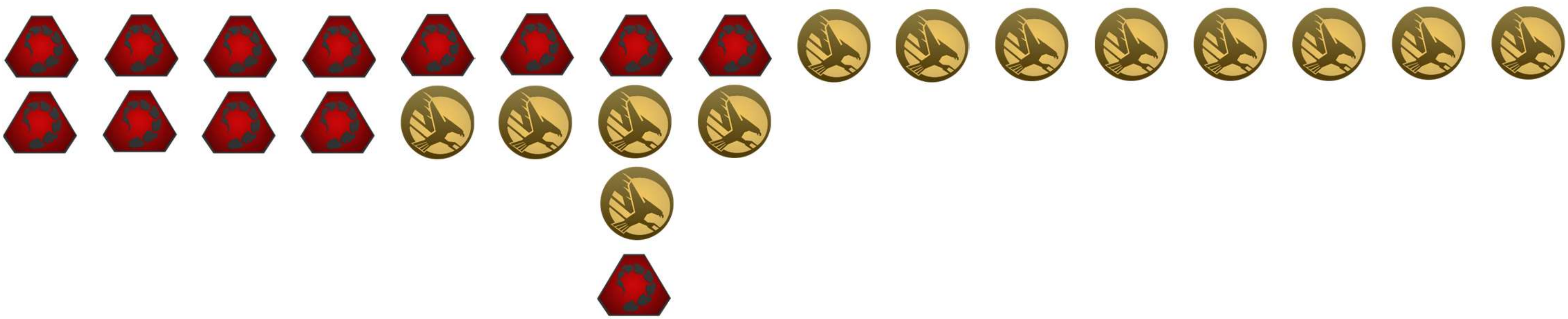
Arnaque à la prédiction :



?

Apprentissage supervisé

Arnaque à la prédiction :



?

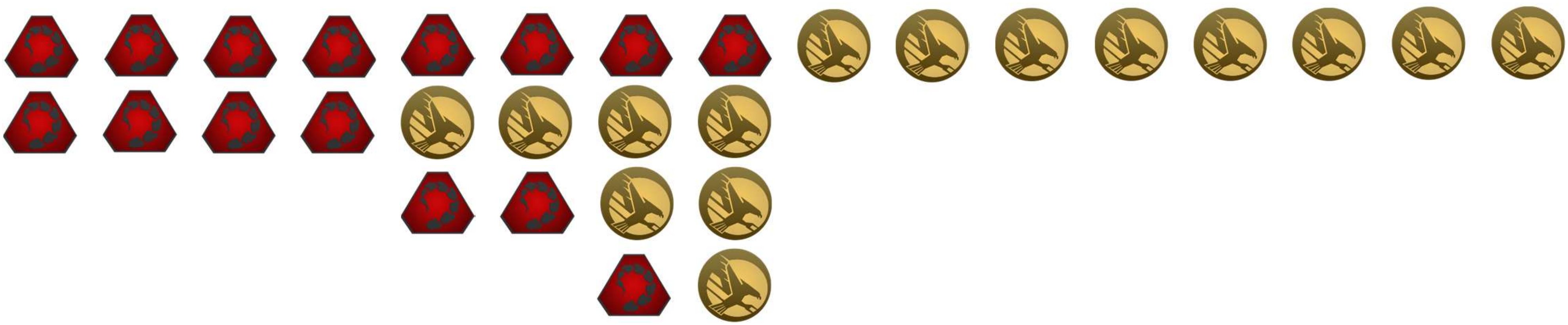
Apprentissage supervisé

Arnaque à la prédiction :

?

Apprentissage supervisé

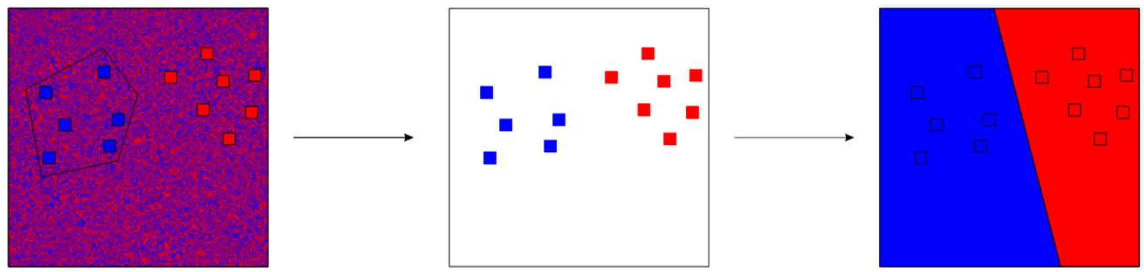
Arnaque à la prédiction :



?

Apprentissage supervisé

$\Rightarrow$Arnaque à la prédiction:



Apprentissage supervisé

Quelles validations théoriques ?

https://work.caltech.edu/telecourse.html

Apprentissage supervisé

Inégalité de Hoeffding :

soit μ: probabilité d'obtenir un échantillon bleu dans un ensemble

soit ν: proportion d'échantillons bleus dans un échantillonnage

Si N est mon nombre d'échantillons et ϵ un réel :

$$P[|\nu - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

Apprentissage supervisé

Ce qui nous amène à :

soit E_{in}: l'erreur de classement d'une hypothèse sur les échantillons par rapport à la fonction cible.

soit E_{out}: l'erreur de classement d'une hypothèse l'ensemble des entrées possibles par rapport à la fonction cible.

soit g : mon hypothèse

soit M : L'ensemble des hypothèses possibles pour mon modèle

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

Apprentissage supervisé

Inégalité de Vapnik-Chervonenkis :

soit m_H: le nombre maximum de dichotomies réalisables sur un ensemble d'exemples par une classe d'hypothèse H.

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 4m_H(2N)e^{-\frac{1}{8}\epsilon^2 N}$$

Apprentissage supervisé

Conclusion théorique :

- Généraliser est parfois possible
- Cela dépend :
  - Du nombre d'exemples étiquetés à disposition
  - De la qualité de la généralisation que l'on cherche
  - De la complexité du modèle utilisé pour générer nos hypothèses

- Règle générale, approximative mais pratique :
  - Ne pas espérer obtenir une bonne généralisation si le nombre d'exemple à disposition n'est pas supérieur à <u>**10 fois**</u> le nombre de paramètres du modèle utilisé.

Classification VS Régression

Classification :

- Appartenance d'un exemple à un ensemble fini :

Classification VS Régression

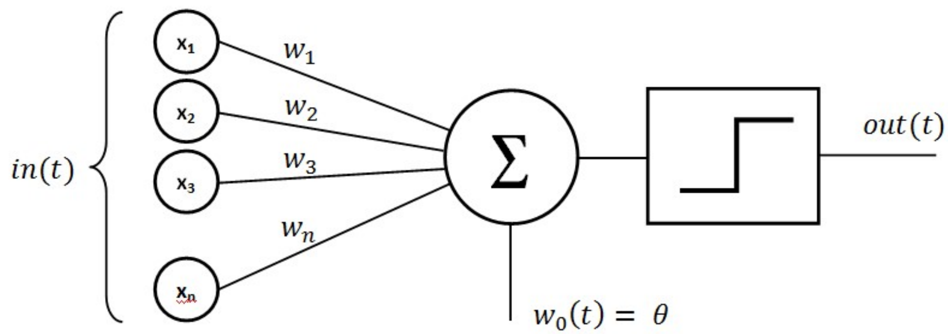
Régression :

- Prédire une (ou plusieurs) valeurs réelles :

Classification et séparations linéaires

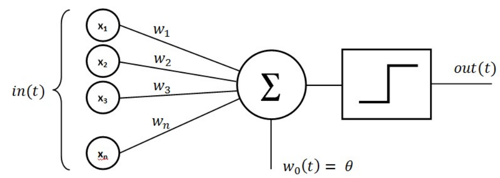
Classification et séparations linéaires

- Retours sur le Perceptron



Classification et séparations linéaires

- Retours sur le Perceptron



- Que l'on peut réécrire :
  - $out = Sign(\sum_{i=0}^{n} w_i x_i)$

- Ou sous forme matricielle :
  - $out = Sign(W^T X)$ en prenant soin d'ajouter le biais ($x_0 = 1$)

Classification et séparations linéaires

- Algorithmes d'apprentissages du perceptron pour la classification
- But du jeu : déterminer W
- Non supervisée
  - Règle de Hebb
- Supervisée
  - PLA ou Règle de Rosenblatt

Classification et séparations linéaires

- Perceptron Learning Algorithm (pour des sorties à -1 ou 1)
  - Initialiser W (random(-1,1) ou 0)
  - Répéter :
    - Prendre un exemple MAL classé (où $g(X^k) \neq Y^k$) au hasard et, mettre à jour W selon la règle :

$$W \leftarrow W + \alpha Y^k X^k$$

- Règle de Rosenblatt (pour des sorties à 0 ou 1) (marche aussi pour des sorties à -1 ou 1)
  - Initialiser W (random(-1,1) ou 0)
  - Répéter :

$$W \leftarrow W + \alpha\left(Y^k - g(X^k)\right)X^k$$

Avec :

- α le pas d'apprentissage
- X^k les paramètres de l'exemples k et le biais $x_0^k = 1$.
- Y^k la sortie attendue pour l'exemple k.
- $g(X^k)$ la sortie obtenue par le perceptron pour l'exemple k.

Régression linéaire

- Minimiser le carré de l’erreur
- Notons $X = \begin{bmatrix} x_0^0 & \cdots & x_n^0 \\ \vdots & \ddots & \vdots \\ x_0^N & \cdots & x_n^N \end{bmatrix}$ et $Y = \begin{bmatrix} y_0^0 & \cdots & y_n^0 \\ \vdots & \ddots & \vdots \\ y_0^N & \cdots & y_n^N \end{bmatrix}$
- Supposons $n \leq N$
- Utilisation de la pseudo inverse pour calculer W en un coup :

$$W \; = ((X^T X)^{-1} X^T) Y$$

Exemples

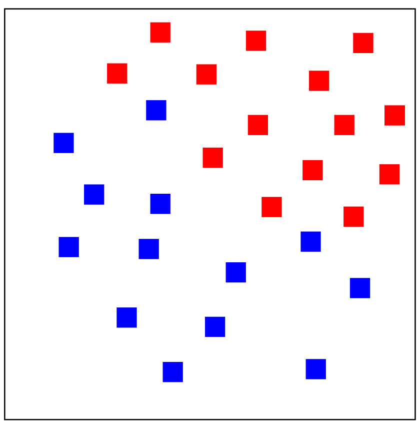
Implémentation

- A vos claviers !

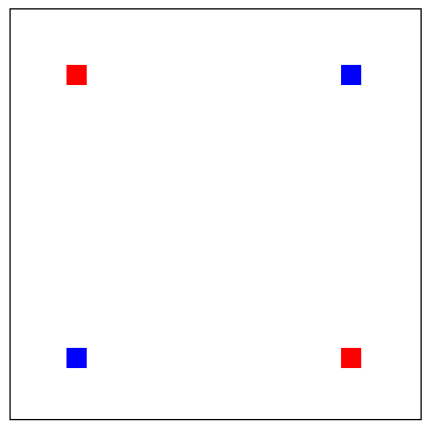
Données non linéairement séparables

Données non linéairement séparables

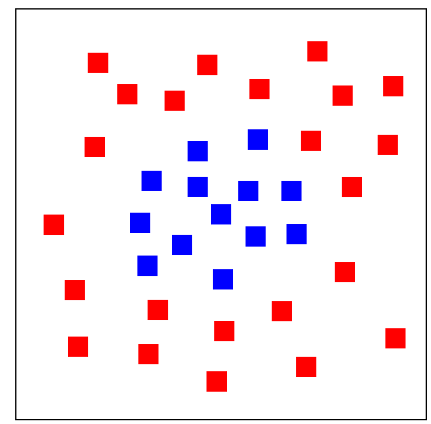
- Que faire dans ces situations ?



Soft (bruit)



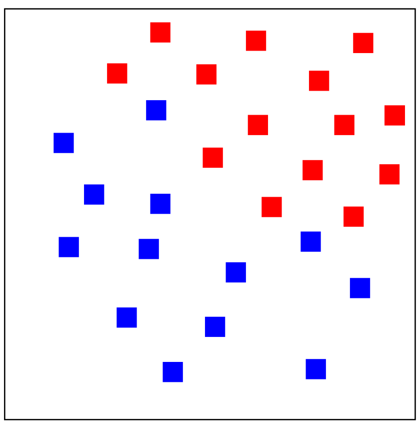
Hard (Intrinsèquement non linéaire)



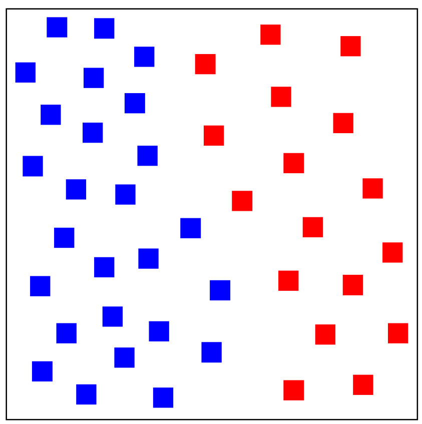
Intrinsèquement non linéaire réel

Données non linéairement séparables

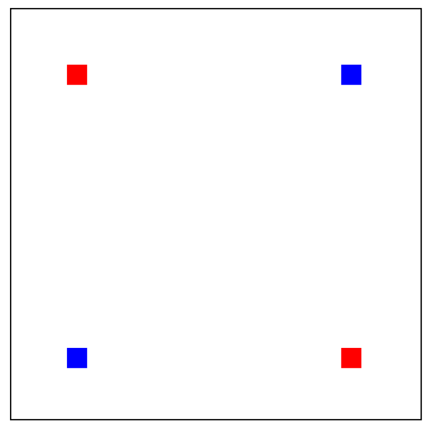
Que faire dans ces situations ?



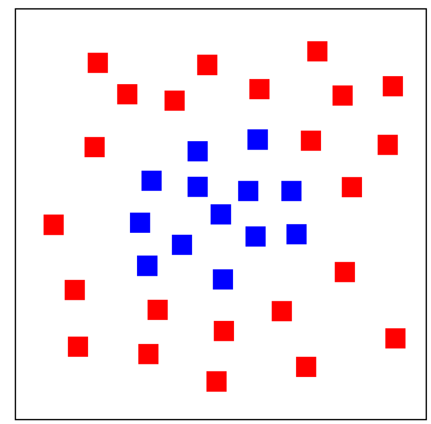
Soft (bruit)



Presque Linéaire (bruit ?)



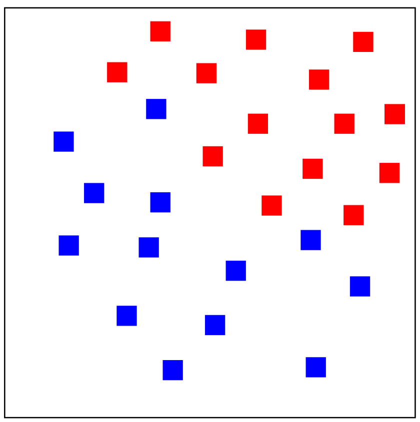
Hard (Intrinsèquement non linéaire)



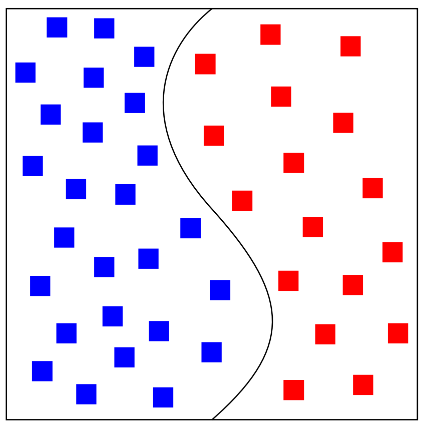
Intrinsèquement non linéaire réel

Données non linéairement séparables

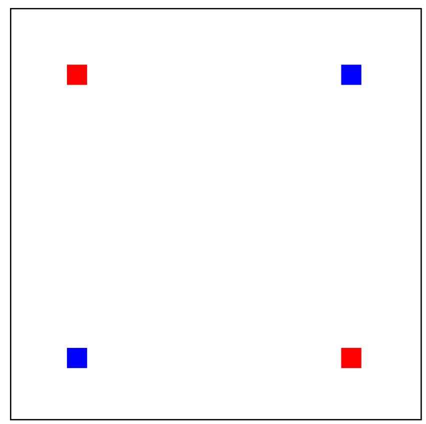
Que faire dans ces situations ?



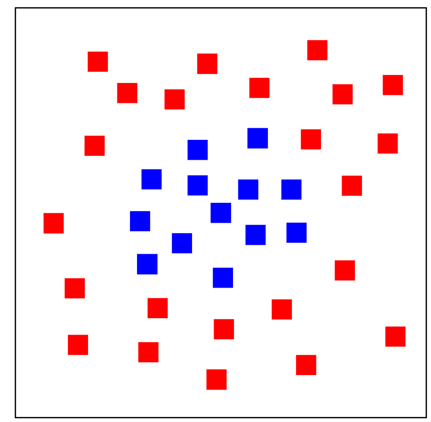
Soft (bruit)



Presque Linéaire (bruit ?)



Hard (Intrinsèquement non linéaire)



Intrinsèquement non linéaire réel

Données non linéairement séparables

De quelle linéarité parle-t-on dans le cas du perceptron ?

Données non linéairement séparables

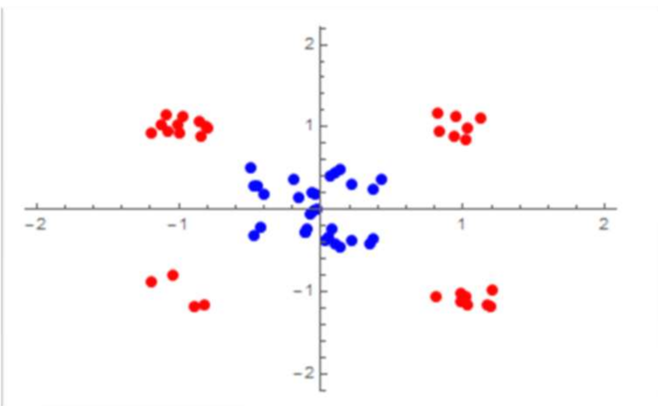
De quelle linéarité parle-t-on dans le cas du perceptron ?

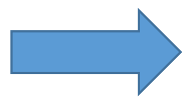
Linéaire en fonction de W, pas de X !

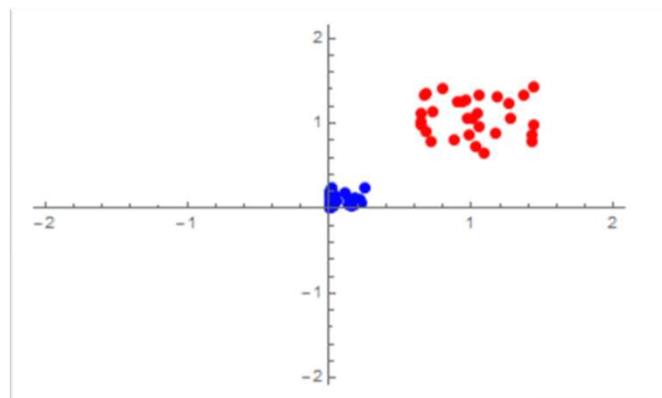
Transformation non linéaire des données d'entrée

Données non linéairement séparables

Transformation non linéaire sur les entrées…







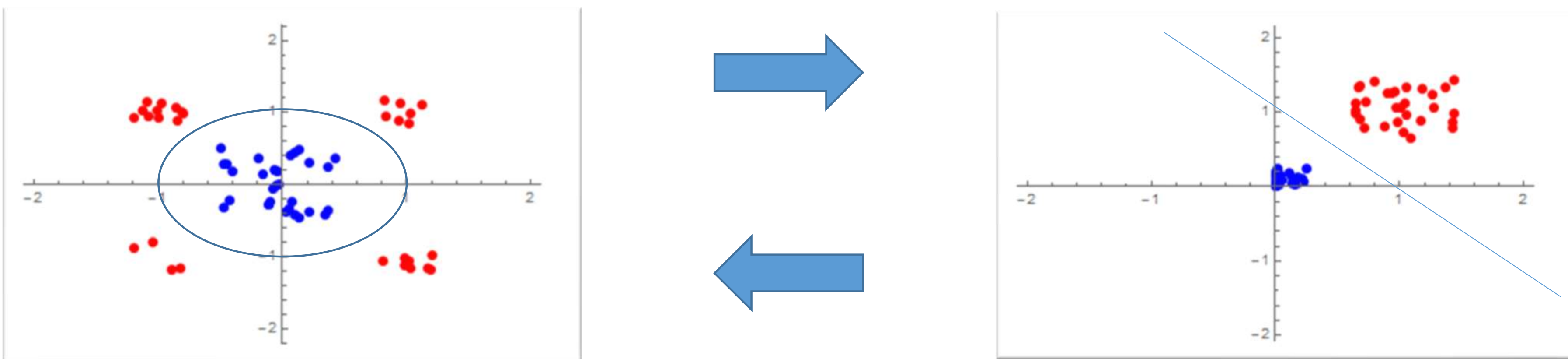
X = (x,y,1)

X = $(x^2, y^2, 1)$

Données non linéairement séparables

Transformation non linéaire sur les entrées…

… et classement dans ce nouvel espace par un perceptron !

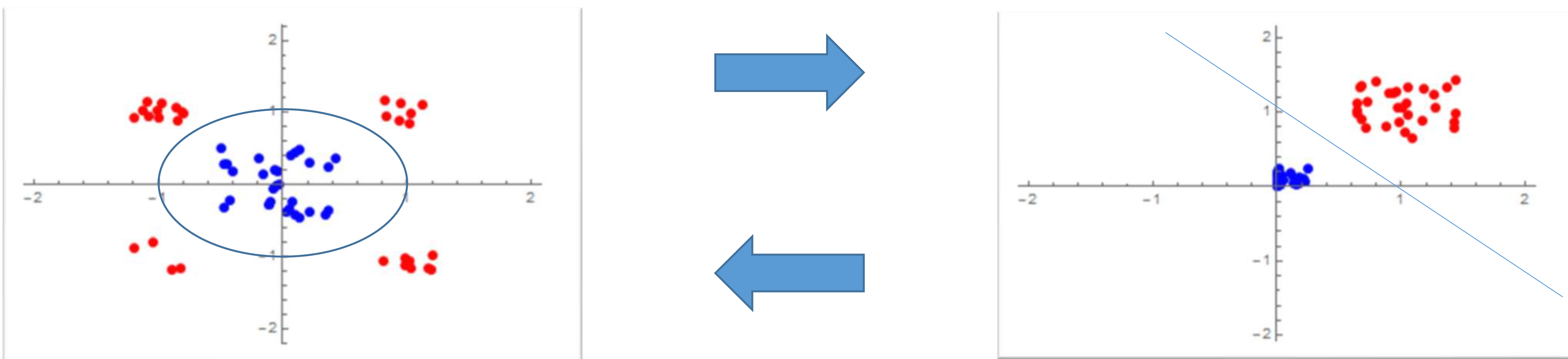


X = (x,y,1) $\qquad\qquad$ X = $(x^2, y^2, 1)$

Données non linéairement séparables

Le perceptron semble avoir toujours le même nombre d'entrées

=> Capacité de généralisation inchangée ? ☺



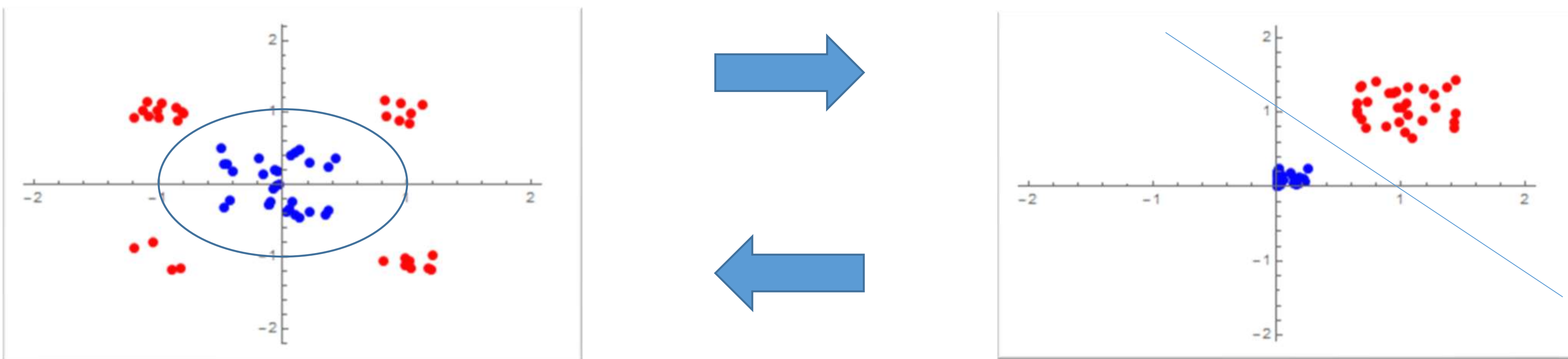
X = (x,y,1)

X = $(x^2, y^2$,1)

Données non linéairement séparables

Le perceptron semble avoir toujours le même nombre d'entrées

=> Capacité de généralisation inchangée ? ☺

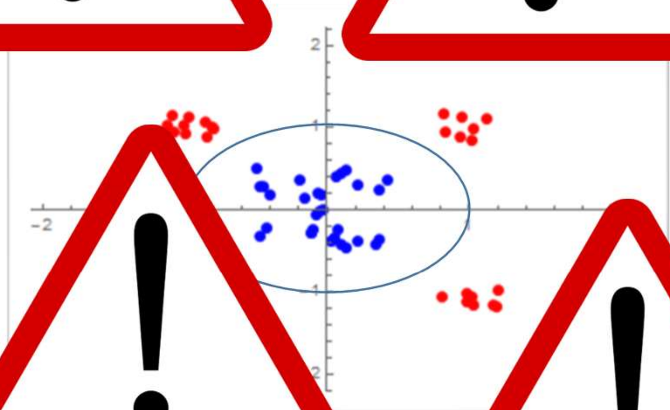


X = (x,y,1)

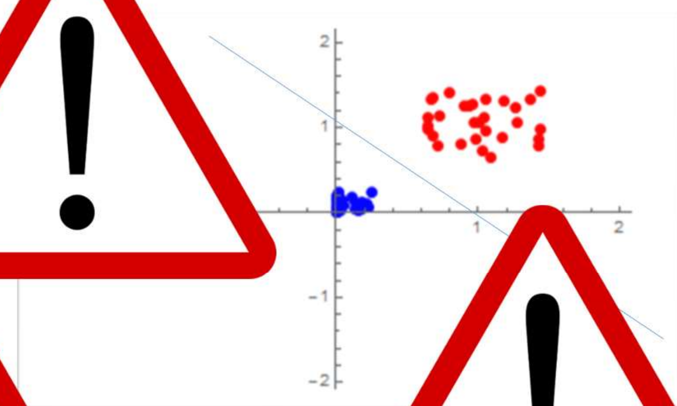
X = $(x^2, y^2, 1)$

Données non linéaire séparables

- ceptron se oir toujours le même nombre d'
- cité de ion inchangée ? ☺



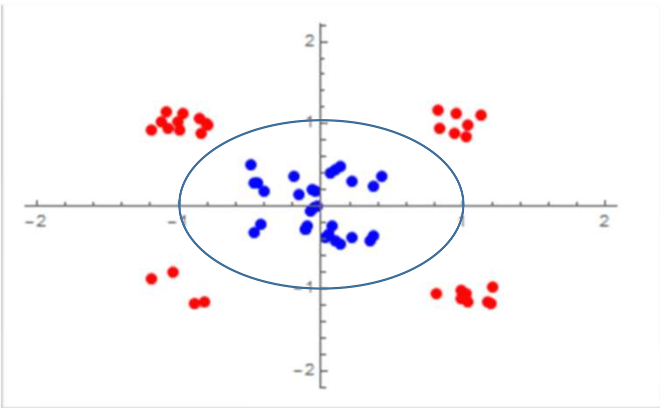
X = (x,y,1)

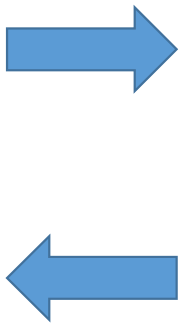


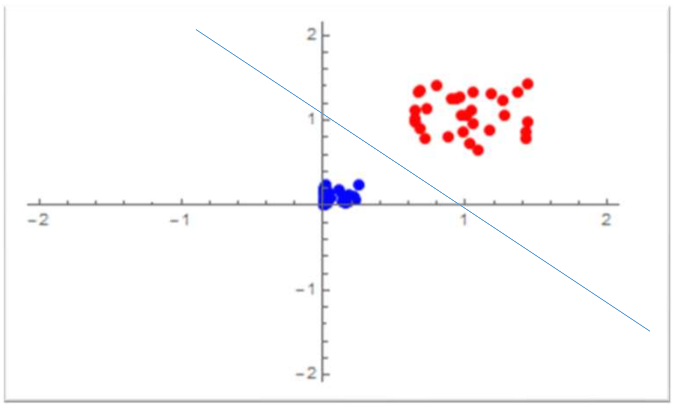
X = (x^2

Données non linéairement séparables

Nous avons choisi cette transformation en particulier …







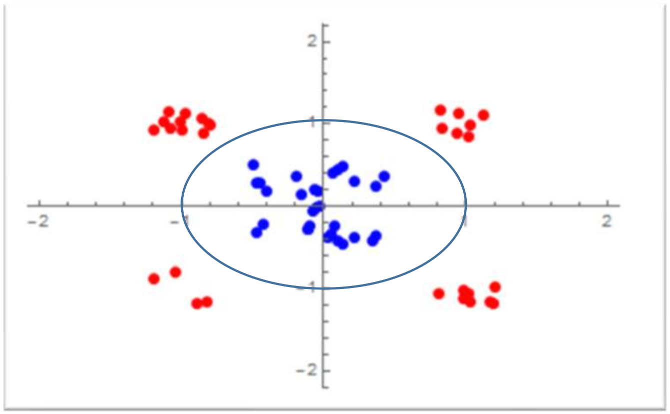
X = (x,y,1)

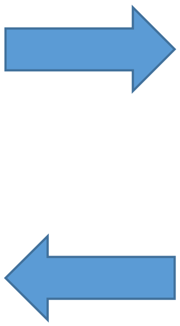
X = (x^2, y^2,1)

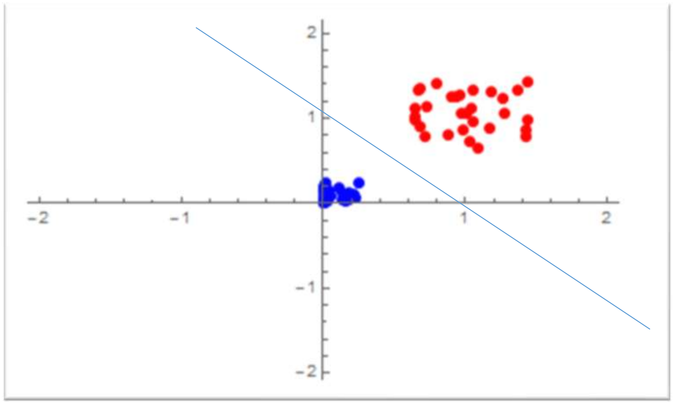
Données non linéairement séparables

Nous avons choisi cette transformation en particulier …

… car nous avons observé les données !!!







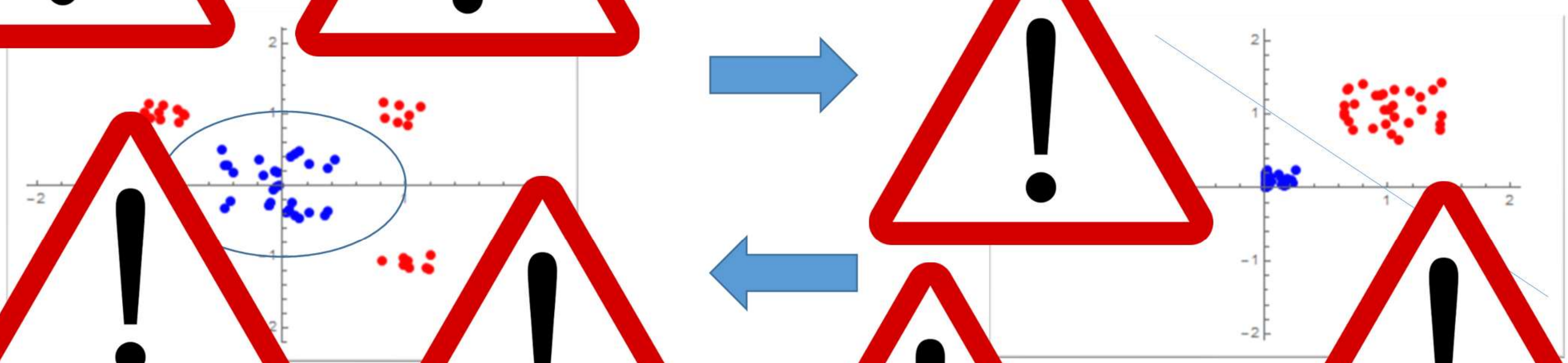
X = (x,y,1)

X = $(x^2, y^2, 1)$

Données non linéaire séparables

avons choi transformation en particulier …

car no servé les données !

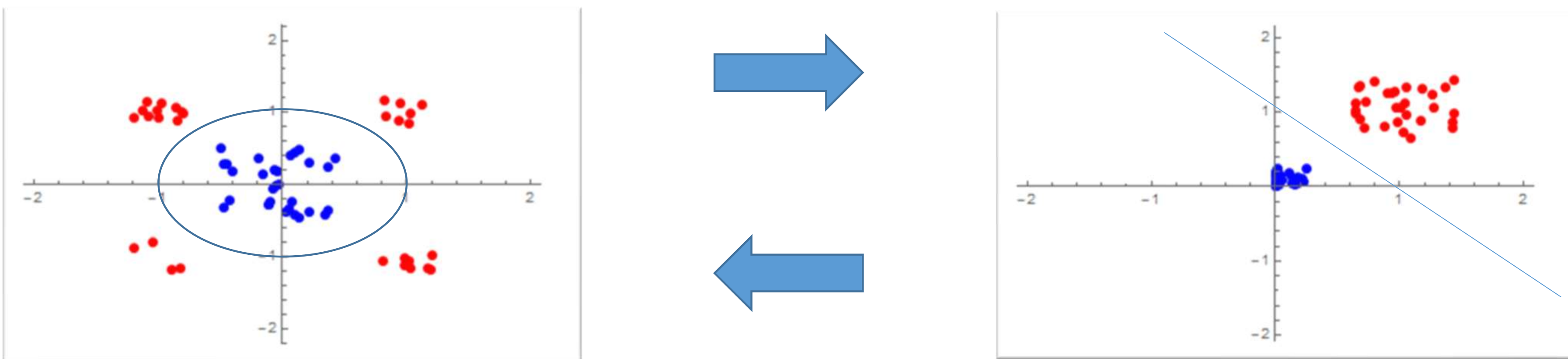


X = (x,y,1)

X = (x^2

Données non linéairement séparables

Entrées réelles :



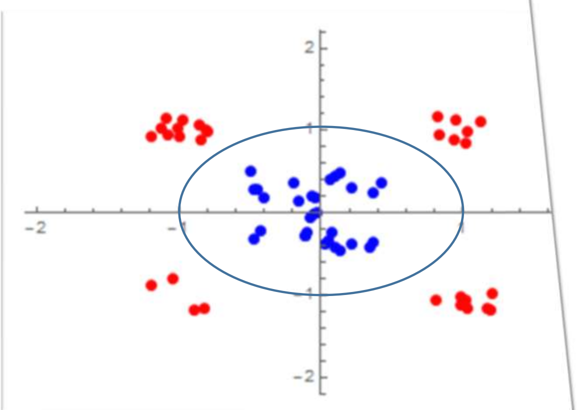
X = (1,x,y)

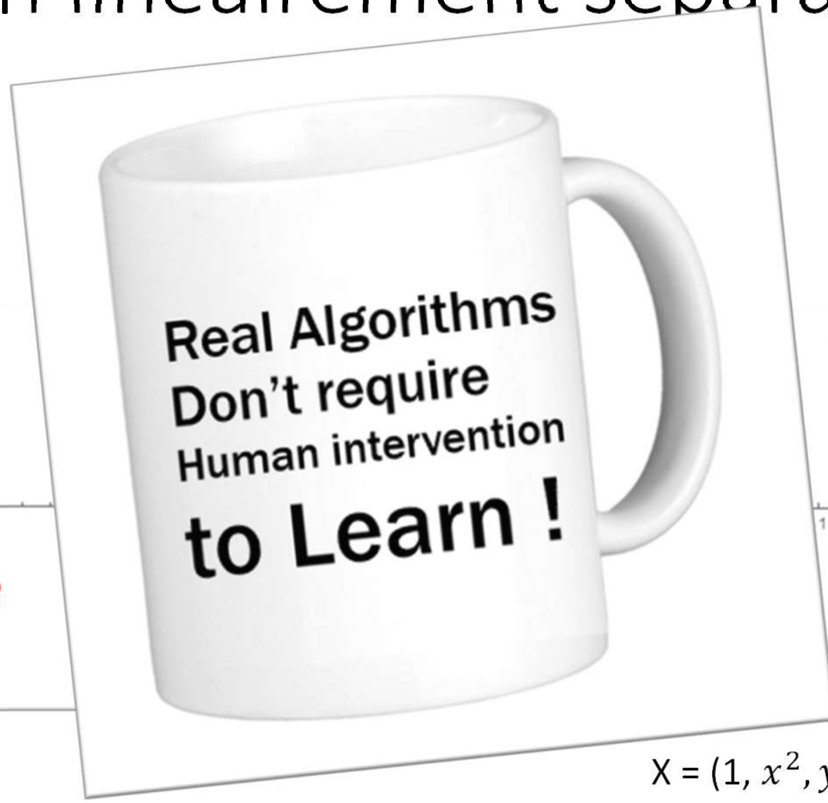
X = (1, x^2, y^2) $\Longleftrightarrow$ X = (1, x, y, xy, x^2, y^2)

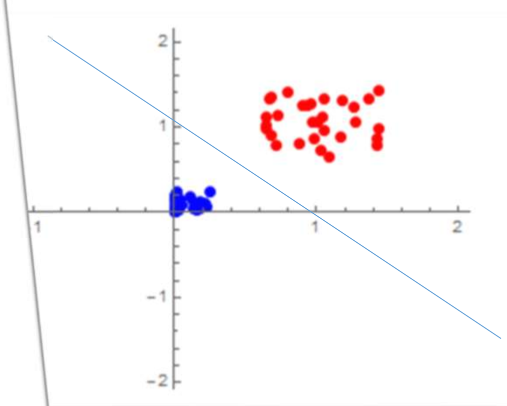
Si on fixe $w_1 = w_2 = w_3 = 0$

Données non linéairement séparables

Entrées réelles :







X = (1,x,y)

X = (1, x^2, y^2) ⟺ X = (1, x, y, xy, x^2, y^2)

Si on fixe $w_1 = w_2 = w_3 = 0$

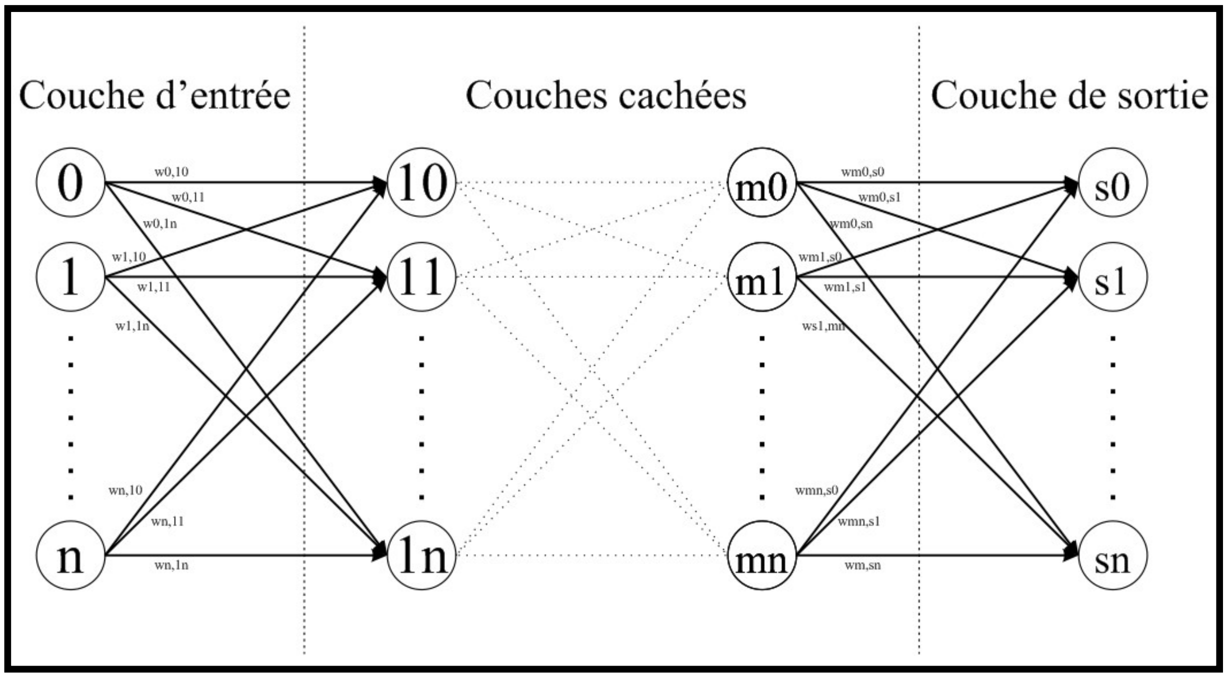
Données non linéairement séparables

Comment correctement estimer le prix de transformations non linéaires des entrées vis-à-vis de la généralisation?

Perceptron Multi Couches

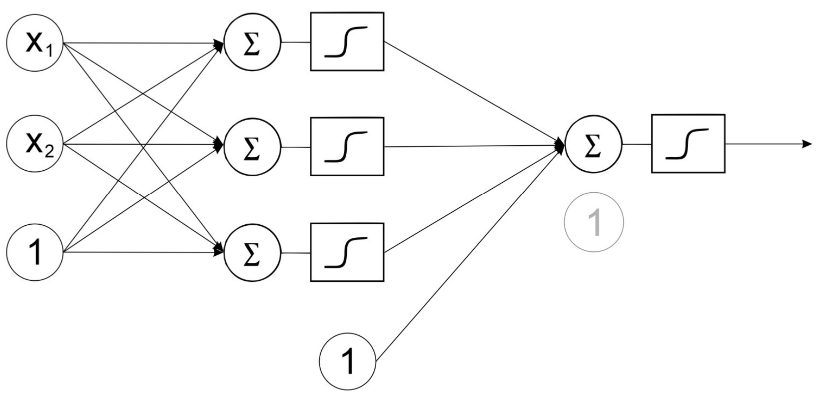
Principes

Intuition : mettre des perceptrons en série et en parallèle …

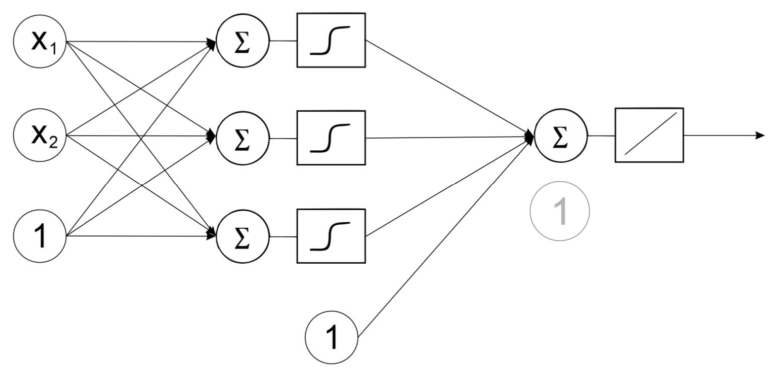


Principes

Intuition : mettre des perceptrons en série et en parallèle …



Perceptron multi couches pour la classification

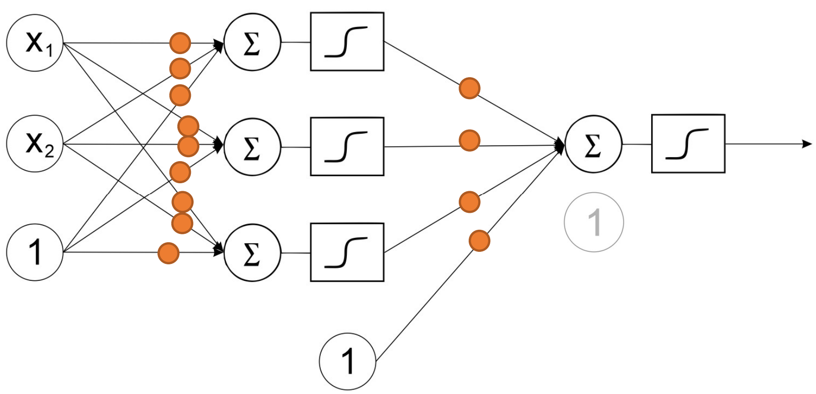


Perceptron multi couches pour la régression

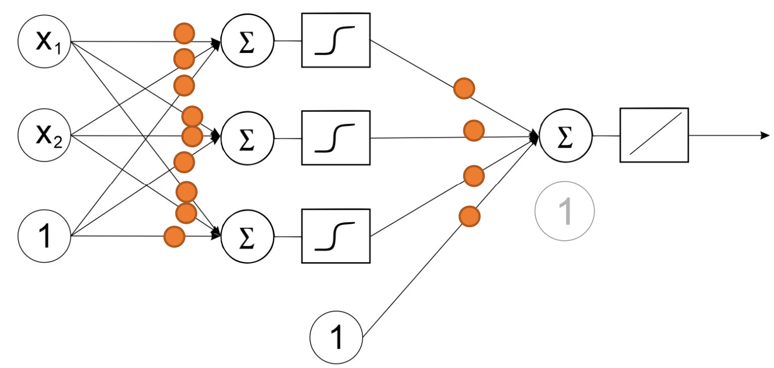
Principes

Intuition : mettre des perceptrons en série et en parallèle …

Paramètres : ensemble des poids



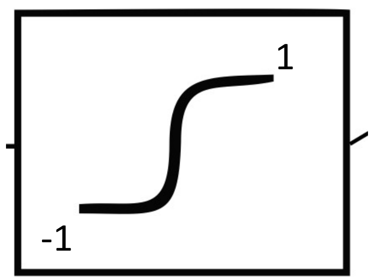
Perceptron multi couches pour la classification



Perceptron multi couches pour la régression

Principes

Nous n'utilisons pas le signe de la somme des entrées pour chaque perceptron, mais une fonction dite « d'activation » sigmoïde.



$$Tanh(x)$$

Principes

Soit w_{ij}^{l} le poids de la couche l liant le neurone i de la couche $l-1$ au neurone j de la couche l.

Soit s_j^l le signal (somme pondérée des entrées) du neurone j de la couche l.

Soit θ la fonction sigmoïde appliquée au signal de chaque neurone intermédiaire (on utilisera $Tanh$).

Soit x_j^l la valeur de sortie effective d'un neurone.

Soit d^l le nombre de neurones appartenant à la couche l (sans compter le neurone de biais)

Principes

Soit x_j^l la valeur de sortie effective du neurone j de la couche l.

Règle récursive de calcul des X:

$$x_j^l = \theta\left(s_j^l\right) = Tanh(\sum_{i=0}^{d^{l-1}} w_{ij}^l x_i^{l-1})$$

Principes

Comment trouver les w_{ij}^{l} minimisant l'erreur de classification sur la base d'exemples?

Rétropropagation du gradient stochastique

Pour la classification, répéter :

- Prendre un exemple étiqueté au hasard : $\begin{bmatrix} x_1^0 \\ \vdots \\ x_{d^0}^0 \end{bmatrix} \rightarrow \begin{bmatrix} y_0 \\ \vdots \\ y_{d^L} \end{bmatrix}$
- Pour tous les neurones j de la dernière couche L calculer :
$$\delta_j^L = (1 - \left(x_j^L\right)^2) \times (x_j^L - y_j)$$
- En déduire pour tous les autres neurones de l'avant dernière couche à la première :
$$\delta_i^{l-1} = (1 - \left(x_i^{l-1}\right)^2) \times \sum_{j=1}^{d^l} (w_{ij}^l \times \delta_j^l)$$
- Puis mettre à jour tous les w_{ij}^l :
$$w_{ij}^l \leftarrow w_{ij}^l - \alpha x_i^{l-1} \delta_j^l$$

Rétropropagation du gradient stochastique

Pour la régression, répéter :

- Prendre un exemple étiqueté au hasard : $\begin{bmatrix} x_1^0 \\ \vdots \\ x_{d^0}^0 \end{bmatrix} \rightarrow \begin{bmatrix} y_0 \\ \vdots \\ y_{d^L} \end{bmatrix}$
- Pour tous les neurones j de la dernière couche L calculer :
$$\delta_j^L = (x_j^L - y_j)$$
- En déduire pour tous les autres neurones de l'avant dernière couche à la première :
$$\delta_i^{l-1} = (1 - \left(x_i^{l-1}\right)^2) \times \sum_{j=1}^{d^l} (w_{ij}^l \times \delta_j^l)$$
- Puis mettre à jour tous les w_{ij}^l :
$$w_{ij}^l \leftarrow w_{ij}^l - \alpha x_i^{l-1} \delta_j^l$$