

Machine Learning

4A

Bienvenue !

- Contact : pro@nicolasvidal.fr

Modalités

- \approx 10h de cours
- \approx 15h de suivi de projet (beaucoup plus de votre côté 😊 !)

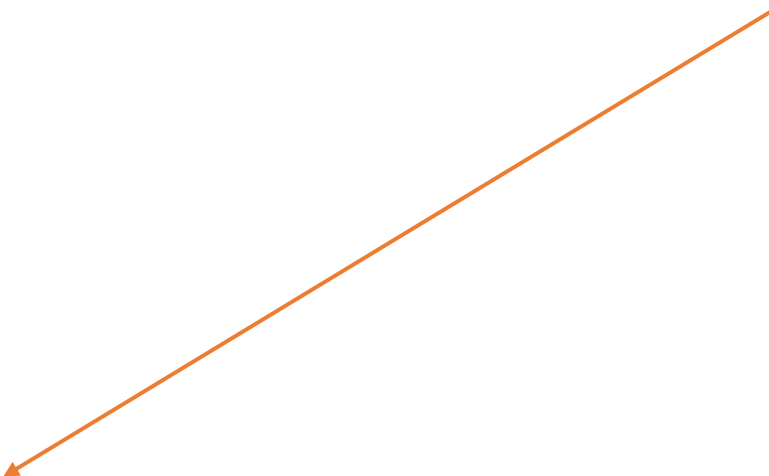
Modalités

- Parlons du projet!

Outils à installer

- Un environnement de développement de visualisation 3D:
 - Unity
 - Unreal Engine
 - ... ?
- Contraintes :
 - Visualisation 3D simple
 - Pouvoir importer une dll C/C++
 - .NET/Java dans le pire des cas (mais déconseillé)

Outils à installer

- Un environnement de développement de de visualisation 3D :
 - Unity
 - Unreal Engine
 - ... ?
 - Contraintes :
 - Visualisation 3D simple
 - Pouvoir importer une dll C/C++
 - .NET/Java dans le pire des cas (mais déconseillé)
- Pourquoi ???
- 

Outils à installer

- Un environnement
 - Unity
 - Unreal Engine
 - ... ?
- Contraintes
 - Visualisation
 - Pouvoir interagir
 - .NET/Java



Pourquoi ???

Outils à installer

- Un environnement
 - Unity
 - Unreal Engine
 - ... ?
- Contraintes
 - Visualisation
 - Pouvoir interagir
 - .NET/Java



Pourquoi ???

Outils à installer

- Un environnement
 - Unity
 - Unreal Engine
 - ... ?
- Contraintes
 - Visualisation
 - Pouvoir interagir
 - .NET/Java

Pourquoi ???

Outils à installer

- Un environnement de calcul scientifique et modélisation
 - Exemples
 - Mathematica (trial 15 days)
 - Anaconda (Python ! 😊) + Keras + Jupyter
 - Octave (Matlab-like)
 - R (Je ne pourrais pas vous aider !)
 - ... ?
 - Contraintes :
 - Pouvoir importer une dll C/C++
 - .NET/Java dans le pire des cas (mais déconseillé)
 - Plot / Génération de data aisée

Outils à installer

- Un environnement de développement en C/C++
 - Exemples
 - Visual Studio
 - CLion
 - Build Essentials
 - ... ?
 - Contraintes :
 - Pouvoir créer une dll C/C++
 - .NET/Java dans le pire des cas (mais déconseillé)

Outils à installer

- Un environnement de développement en ~~C/C++~~
 - Exemples
 - Visual Studio
 - CLion
 - Build Essentials
 - ... ?
 - Contraintes :
 - Pouvoir créer une dll ~~C/C++~~
 - .NET/Java dans le pire des cas (mais déconseillé)

Outils à installer

- Un environnement de développement en C++
 - Exemples
 - Visual Studio
 - CLion
 - Build Essentials
 - ... ?
 - Contraintes
 - Pouvoir créer des dll C/C++
 - .NET/Java dans certains cas (mais déconseillé)

Le projet !

- Vous faire implémenter votre propre toolbox de Machine Learning !

Le projet !

- Vous faire implémenter votre propre toolbox de Machine Learning !
- Vous faire implémenter votre propre toolbox de Machine Learning !

Le projet !

- Vous faire implémenter votre propre toolbox de Machine Learning !
- Vous faire implémenter votre propre toolbox de Machine Learning !
- Utiliser votre toolbox dans votre environnement de développement de visualisation 3D préféré
- Acquérir une sensibilité à la problématique de l'apprentissage artificiel supervisé

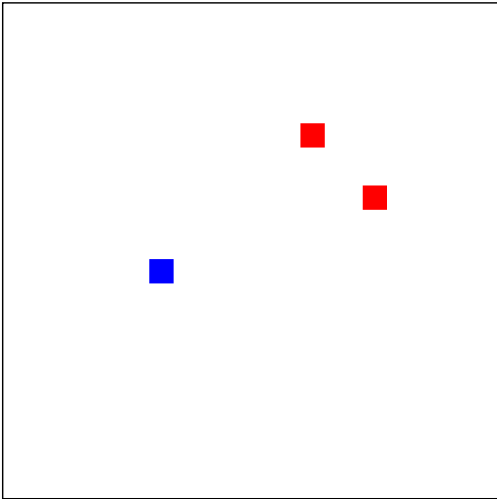
Le projet !

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

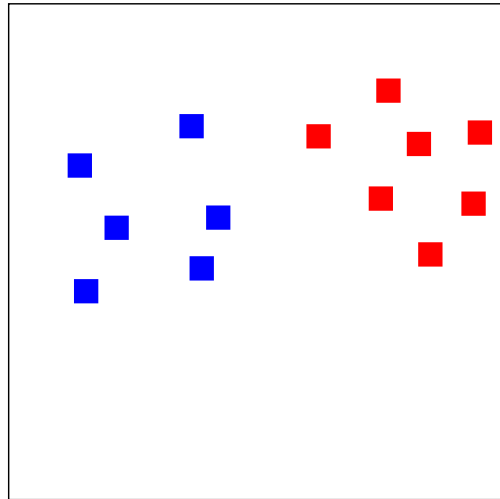


Le projet !

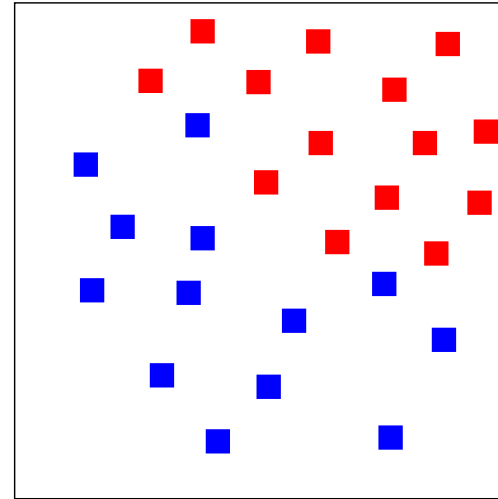
2. Création des cas de tests



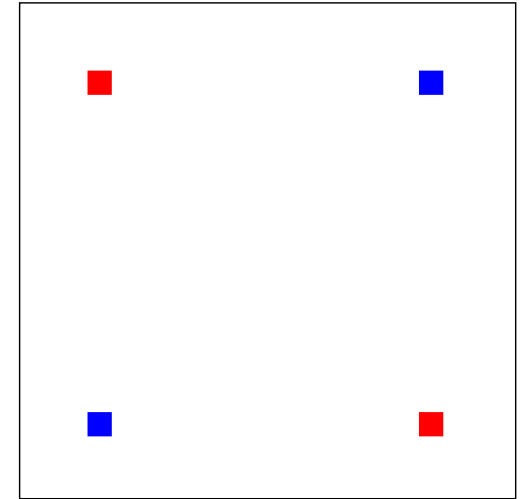
Simple, linéairement séparable



Réel, linéairement séparable



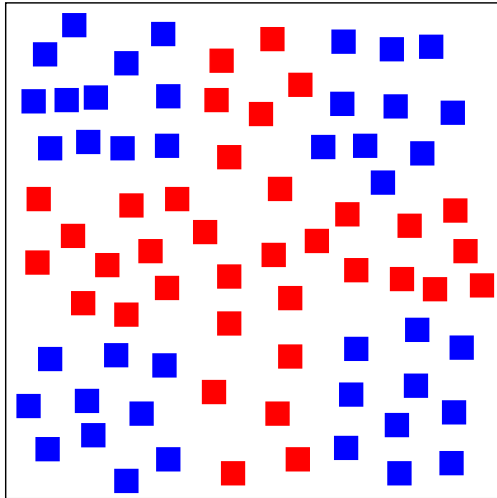
Soft, non linéairement séparable



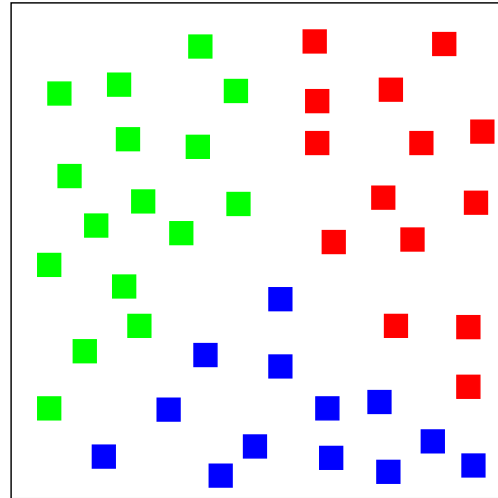
XOR

Le projet !

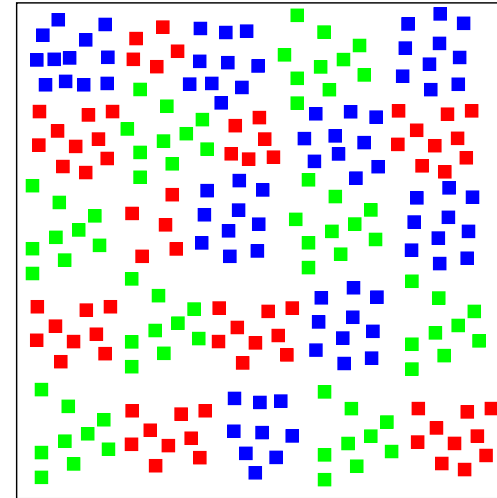
2. Création des cas de tests



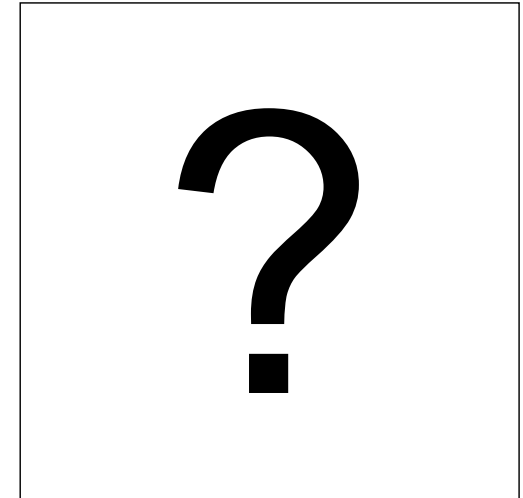
Cross



Multi Class Soft



Multi Class Hard

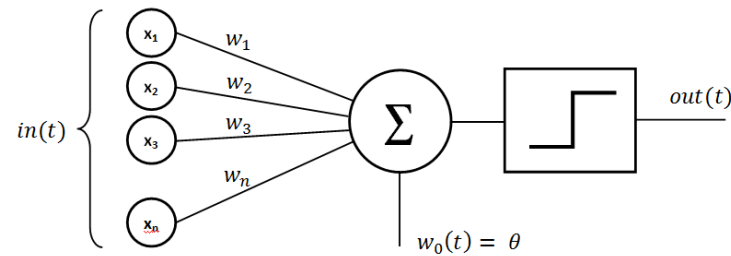


Real Dataset

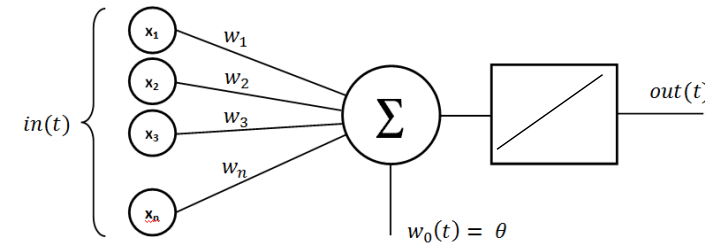
Le projet !

3. Implémentation des algorithmes

1. Modèles linéaires



Perceptron pour la Classification

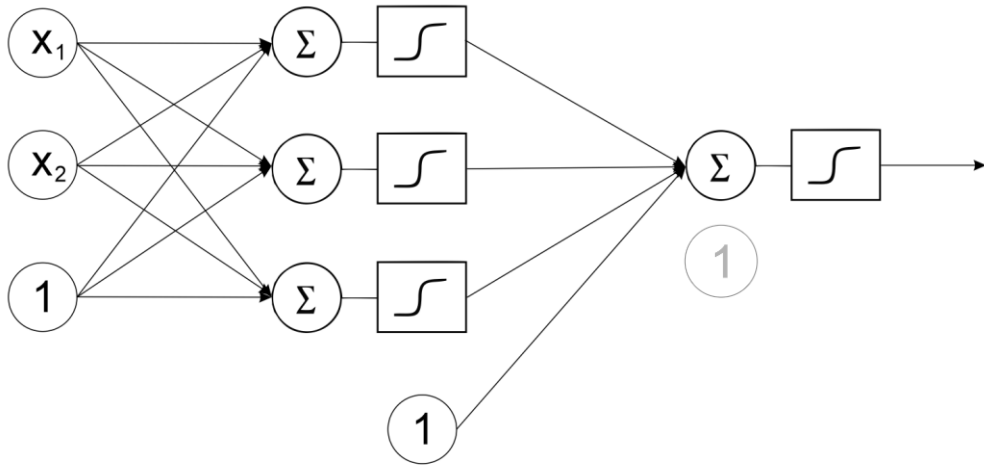


Perceptron pour la Régression

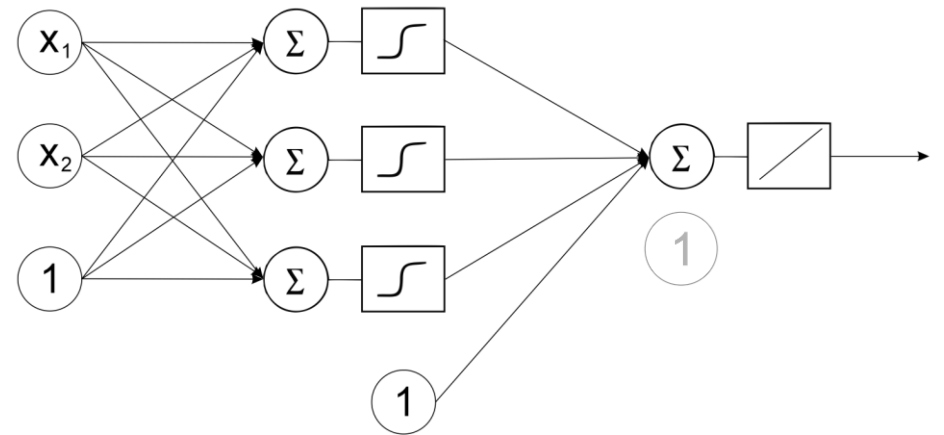
Le projet !

3. Implémentation des algorithmes

2. Perceptron Multi Couches



Perceptron multi couches pour la classification

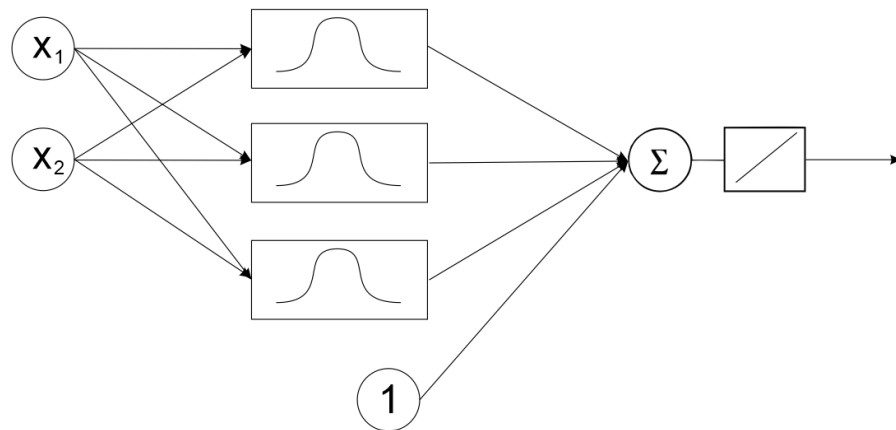


Perceptron multi couches pour la régression

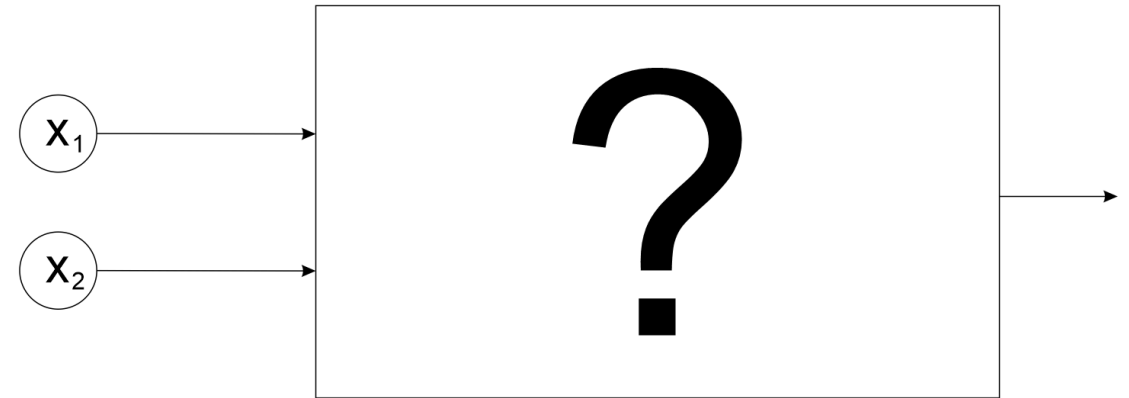
Le projet !

3. Implémentation des algorithmes

3. Modèles non linéaires



RBF (s)



Autre

Le projet !

4. Application a un dataset réel

1. Réaliser un jeu
2. Enregistrer l'historique « état du jeu » / « action choisie par le joueur » de plusieurs joueurs humains sur de nombreuses parties
3. Etablir un protocole de test
4. Entraînement du/des modèles
5. Présentation et analyse des résultats
6. Démonstration des modèles entraînés qui jouent « comme un humain »

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 ?

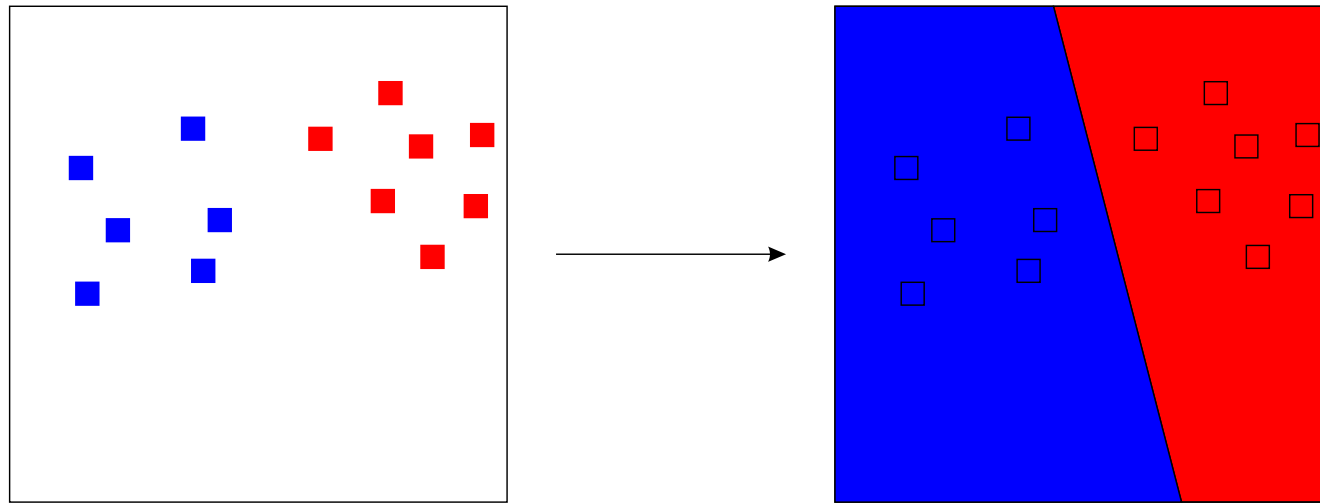
Qu'est-ce qu'apprendre ?

Intuition :

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

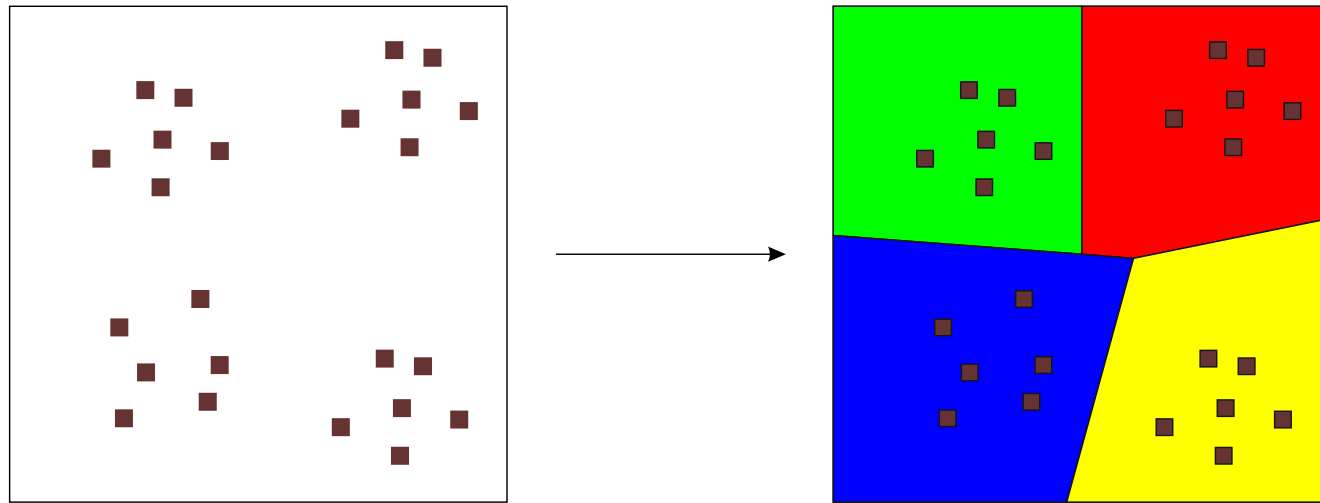
Qu'est-ce qu'apprendre ?

Apprentissage supervisé :



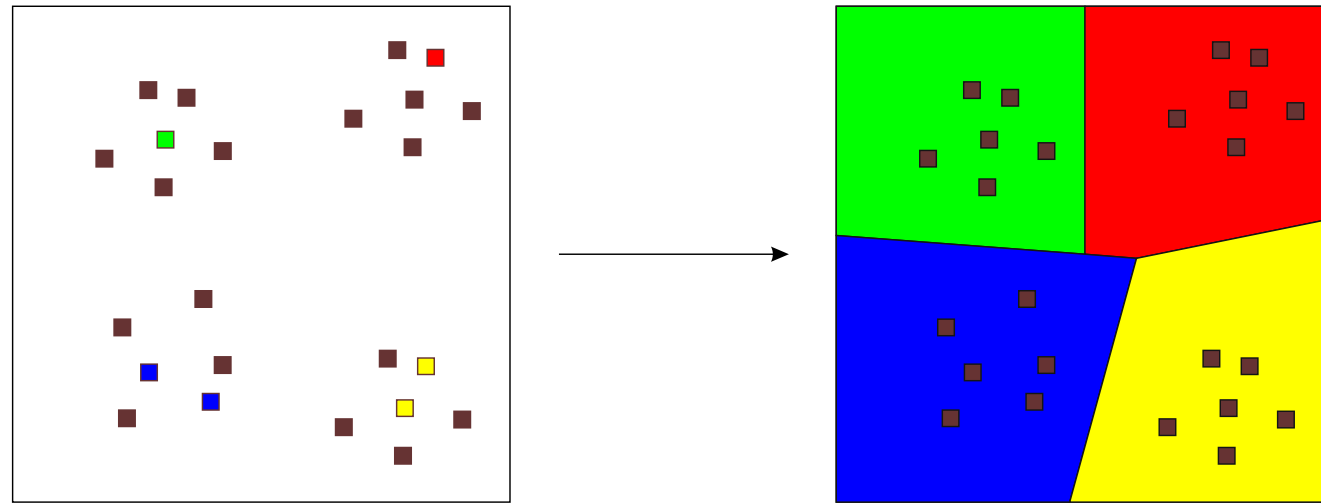
Qu'est-ce qu'apprendre ?

Apprentissage non supervisé :



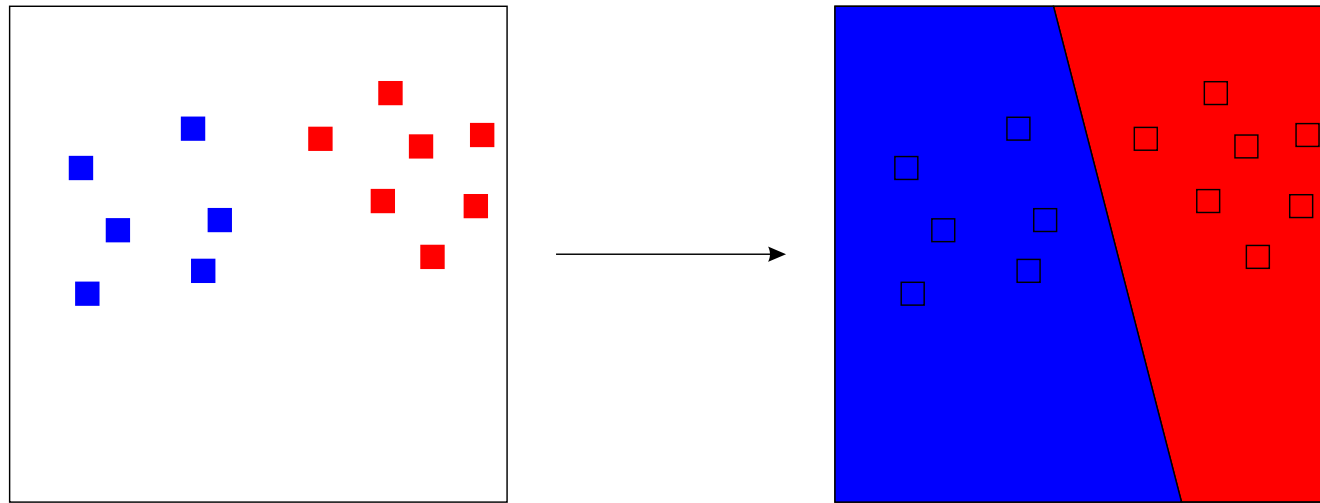
Qu'est-ce qu'apprendre ?

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)
 - $\{1, 2\} \Rightarrow \{3\}$
 - $\{4, 2\} \Rightarrow \{6\}$
 - $\{2, 2\} \Rightarrow \{4\}$
 - $\{8, 13\} \Rightarrow \{21\}$

Apprentissage supervisé



Apprendre ...

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

Apprentissage supervisé



Apprendre ...

- Exemples (Input => Output)
 - $\{1, 2\} \Rightarrow \{3\}$
 - $\{4, 2\} \Rightarrow \{6\}$
 - $\{2, 2\} \Rightarrow \{4\}$
 - $\{8, 13\} \Rightarrow \{21\}$
- 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 **fonction cible** qui a généré les exemples que nous avons à disposition.

⇒ Essayer d'**approximer** 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 :

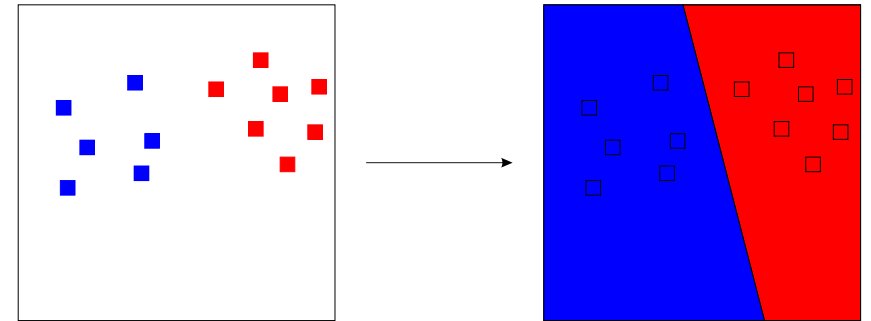
⇒ Supposer qu'il existe une **fonction cible** qui a généré les exemples que nous avons à disposition.

⇒ Essayer d'**approximer** 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

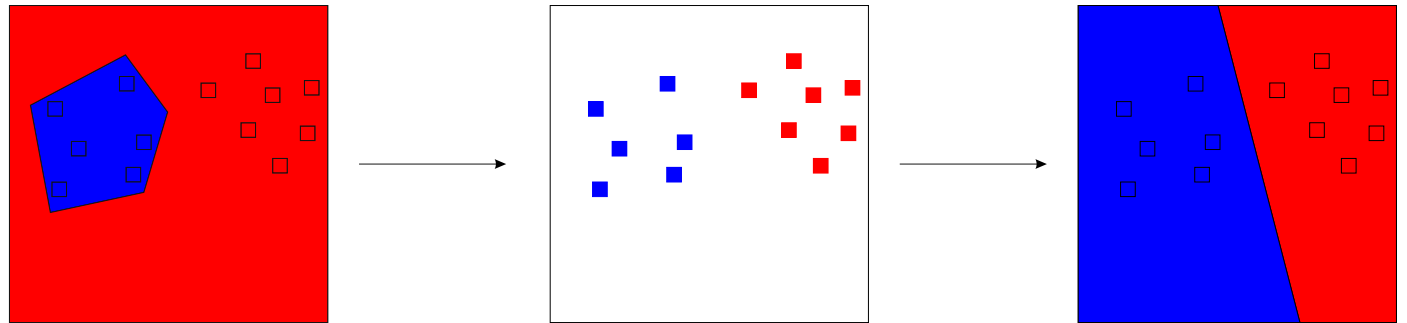
Apprentissage supervisé

⇒ 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 :

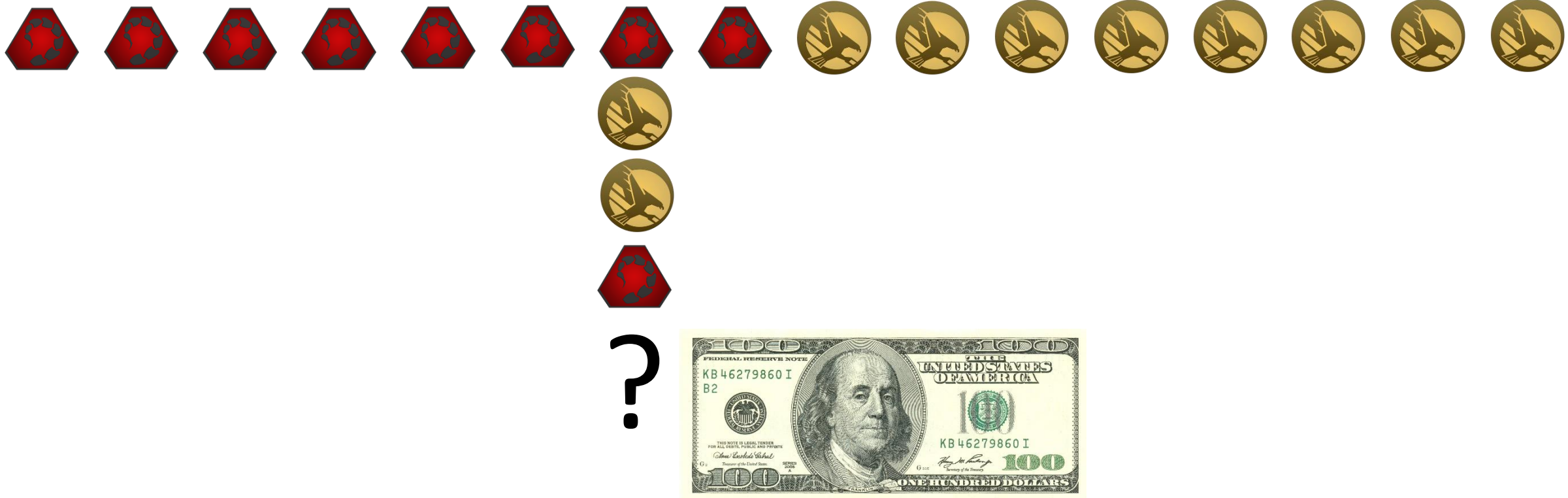


?



Apprentissage supervisé

Arnaque à la prédiction :



Apprentissage supervisé

Arnaque à la prédiction :



?



Apprentissage supervisé

Arnaque à la prédiction :



?



Apprentissage supervisé

Arnaque à la prédiction :

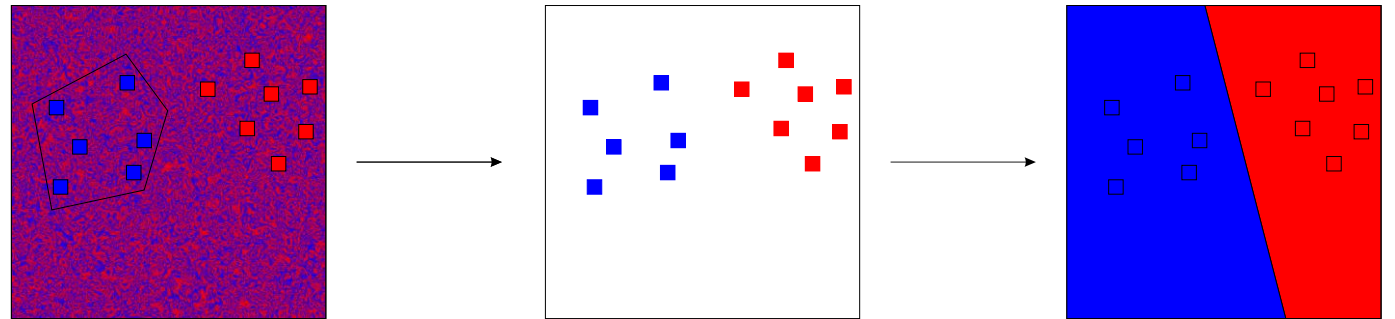


?



Apprentissage supervisé

⇒ 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

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

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 **10 fois** le nombre de paramètres du modèle utilisé.



Classification VS Régression

Classification :

- Appartenance d'un exemple à un ensemble fini :



Classification VS Régression

Régression :

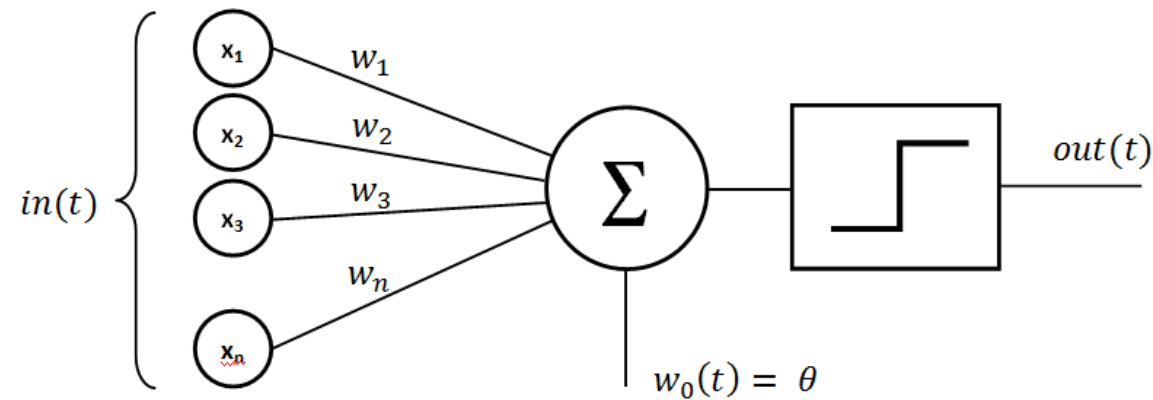
- Prédire une (ou plusieurs) valeurs réelles :



Classification et séparations linéaires

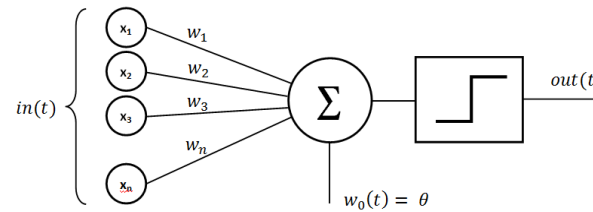
Classification et séparations linéaires

- 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 :

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

Avec :

- α le pas d'apprentissage
- X^k les paramètres de l'exemple 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

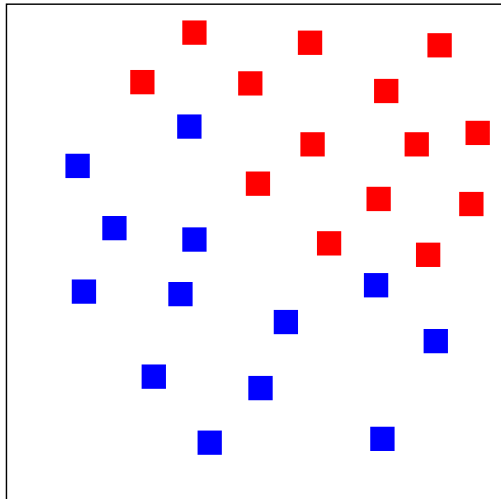
Implémentation

- A vos claviers !

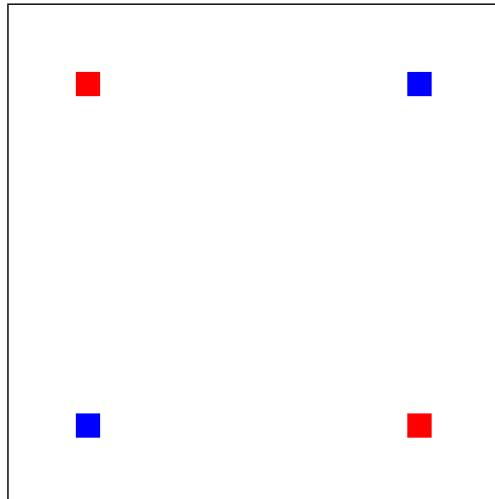
Données non linéairement séparables

Données non linéairement séparables

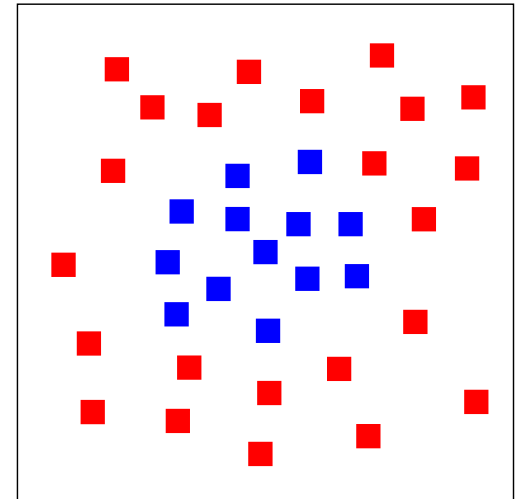
- Que faire dans ces situations ?



Soft (bruit)



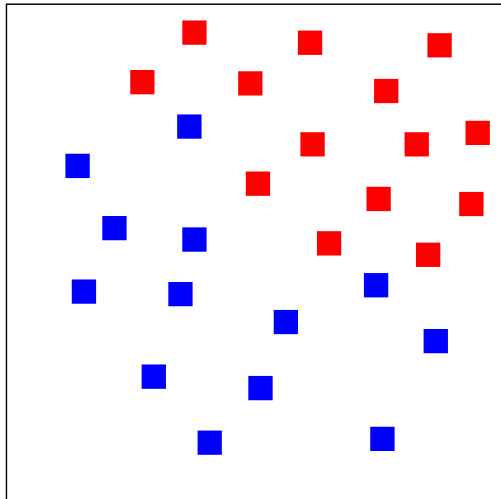
Hard (Intrinsèquement non linéaire)



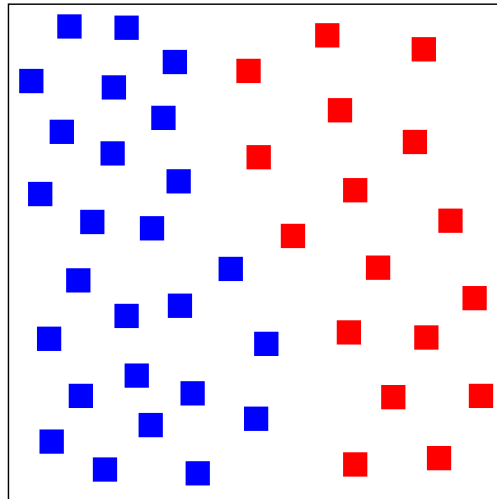
Intrinsèquement non linéaire réel

Données non linéairement séparables

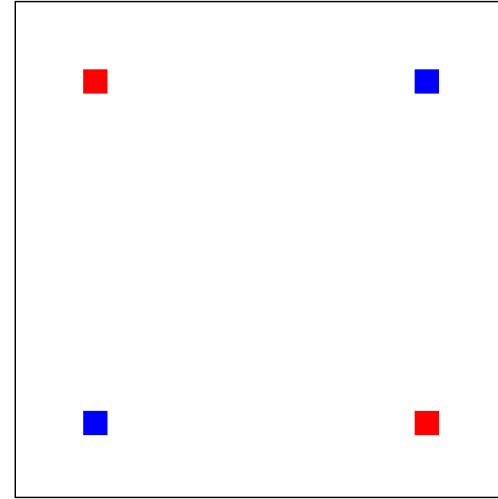
Que faire dans ces situations ?



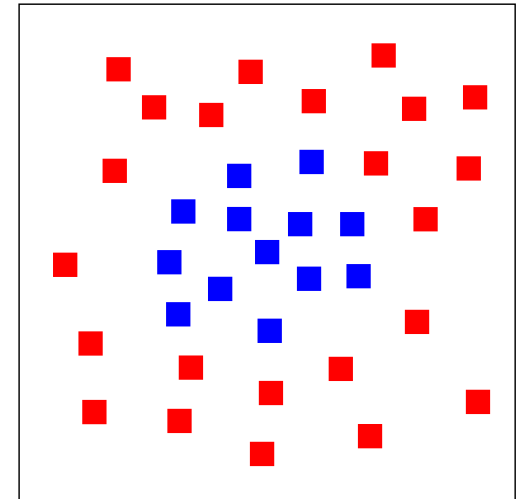
Soft (bruit)



Presque Linéaire (bruit ?)



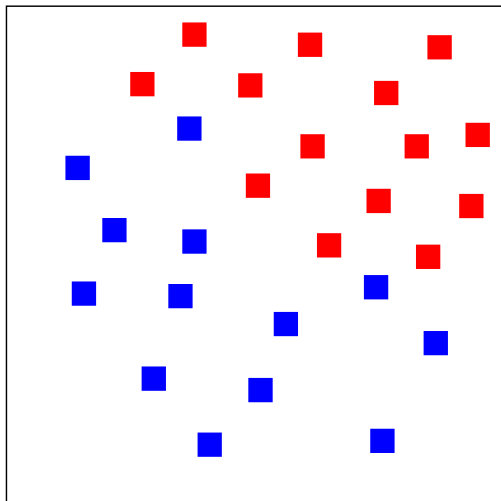
Hard (Intrinsèquement
non linéaire)



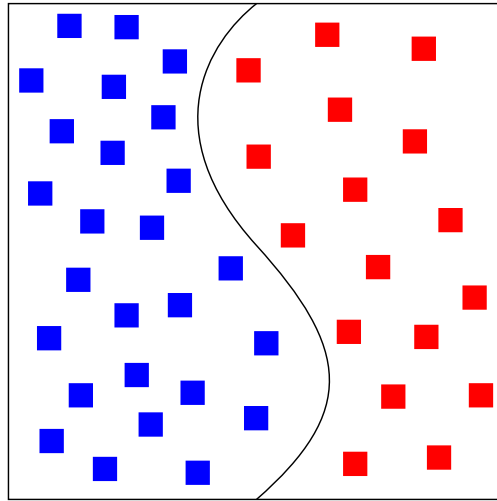
Intrinsèquement non linéaire réel

Données non linéairement séparables

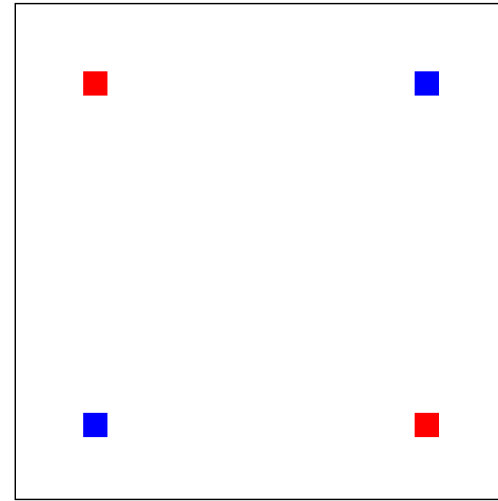
Que faire dans ces situations ?



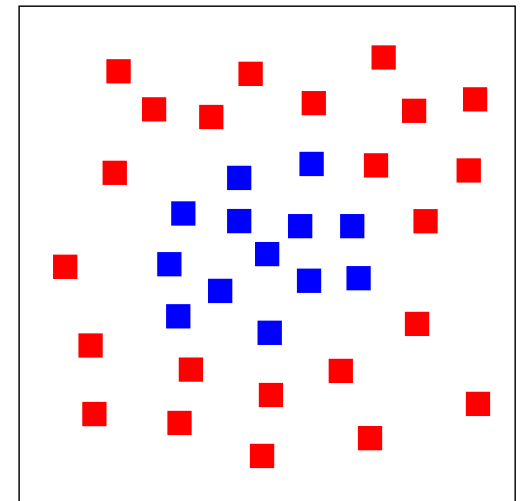
Soft (bruit)



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

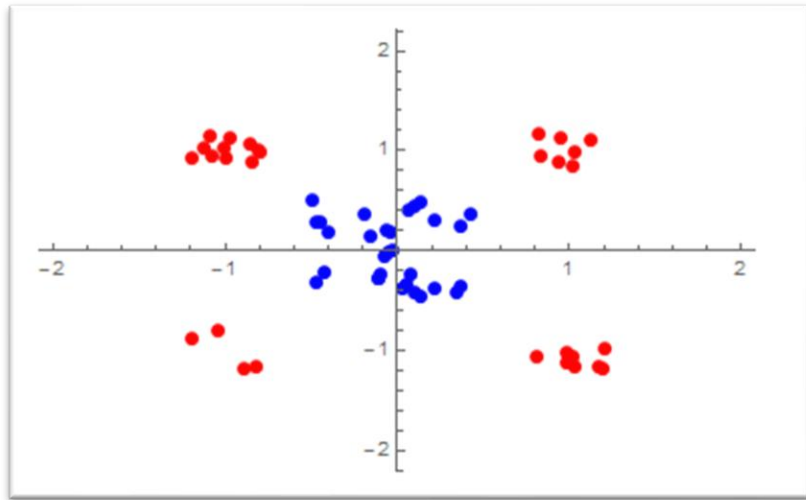
De quelle linéarité parle-t-on dans le cas du perceptron ?

Linéaire en fonction de W , pas de X !

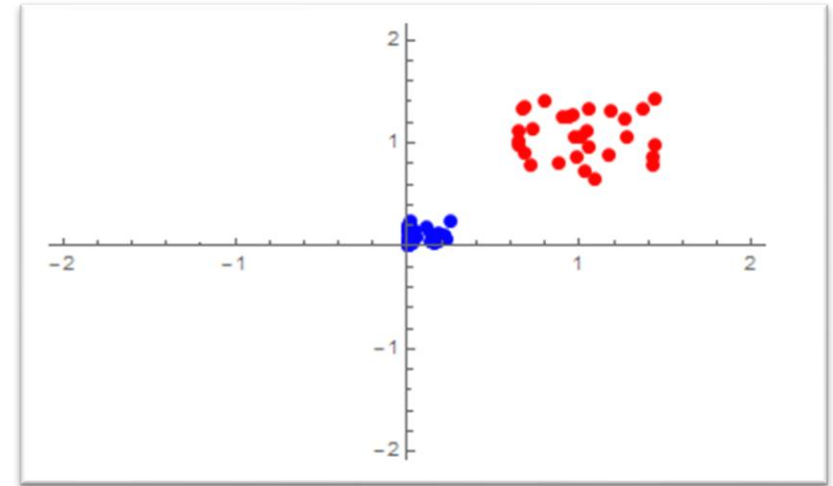
Transformation non linéaire des données d'entrée

Données non linéairement séparables

Transformation non linéaire sur les entrées...



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

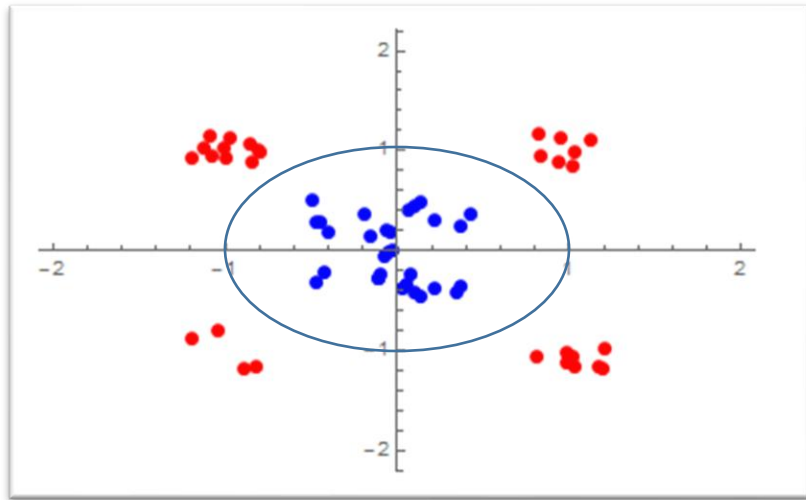


$$X = (x^2, y^2, 1)$$

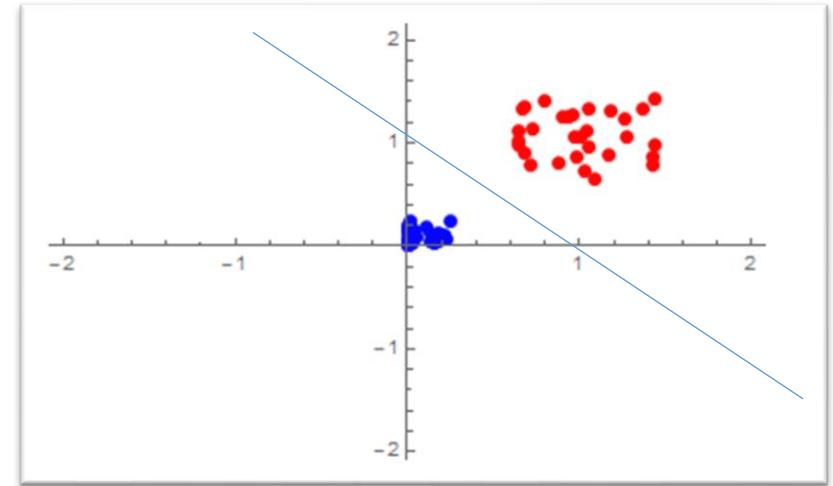
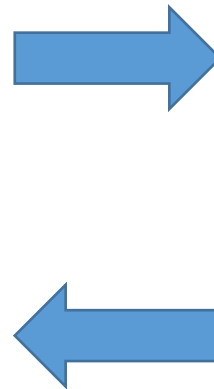
Données non linéairement séparables

Transformation non linéaire sur les entrées...

... et classement dans ce nouvel espace par un perceptron !



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

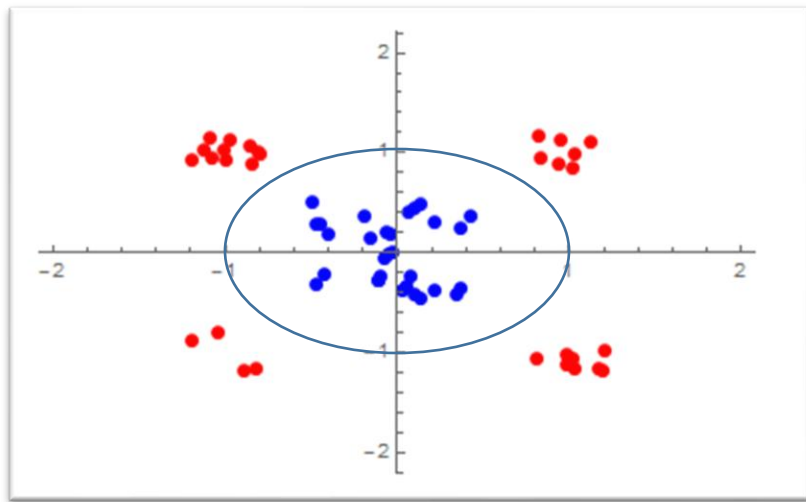


$$X = (x^2, y^2, 1)$$

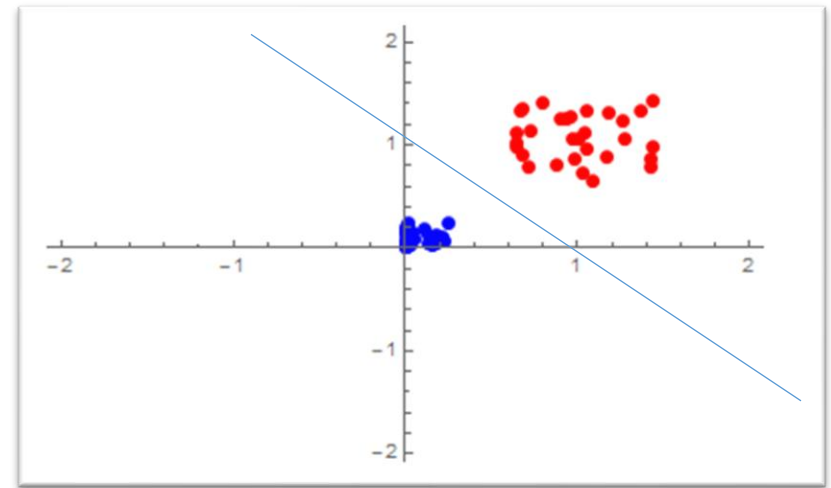
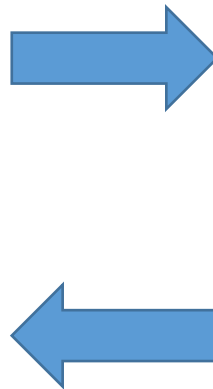
Données non linéairement séparables

Le perceptron semble avoir toujours le même nombre d'entrées

=> Capacité de généralisation inchangée ? 😊



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



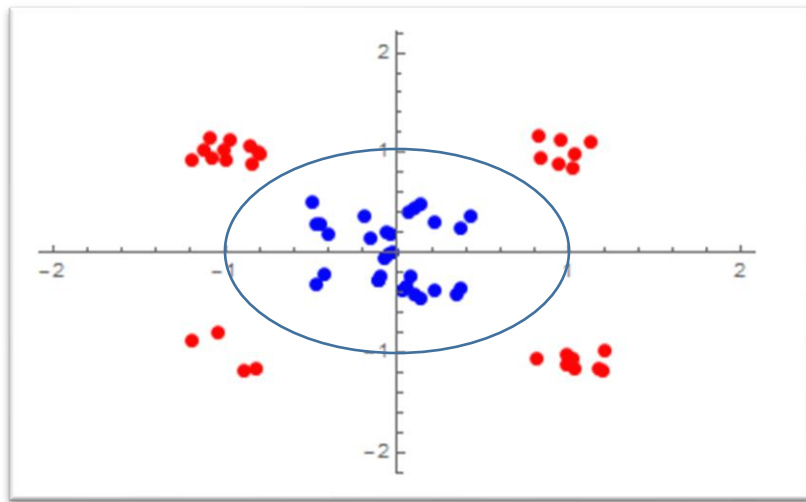
$$X = (x^2, y^2, 1)$$

Données non linéairement séparables

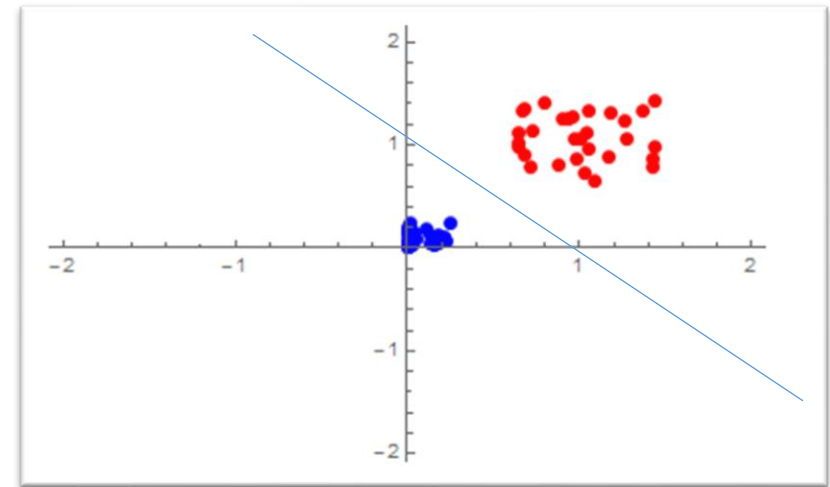
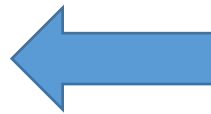


Le perceptron semble avoir toujours le même nombre d'entrées

=> Capacité de généralisation inchangée ? 😊



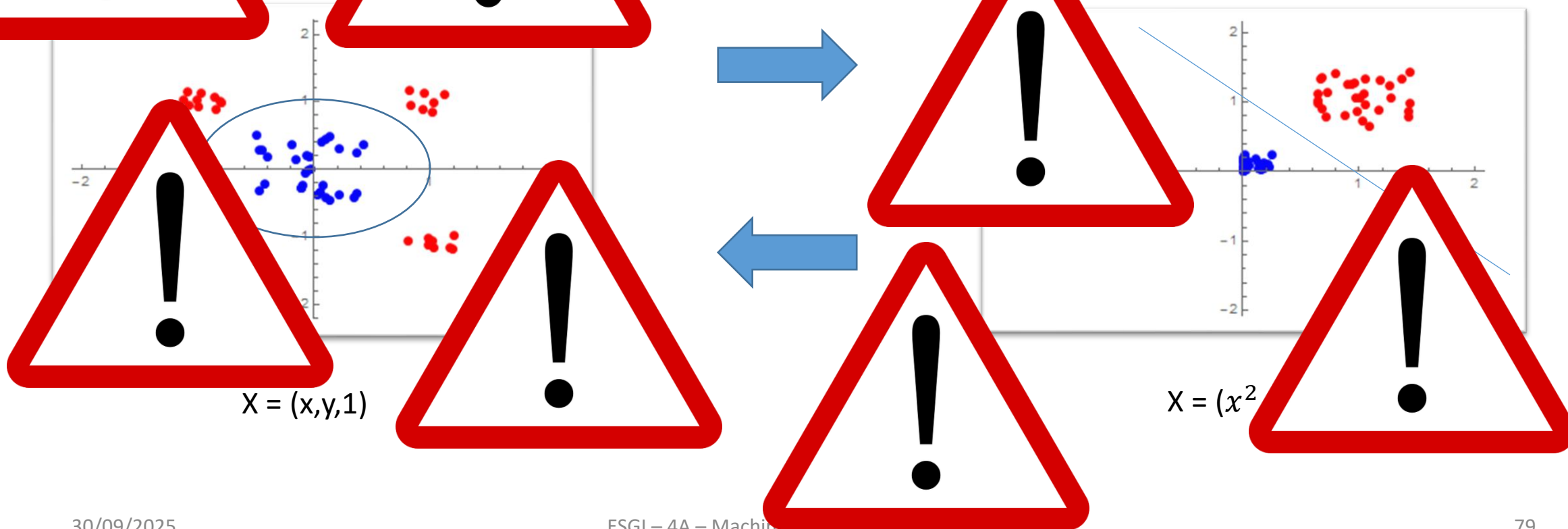
$$X = (x, y, 1)$$



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

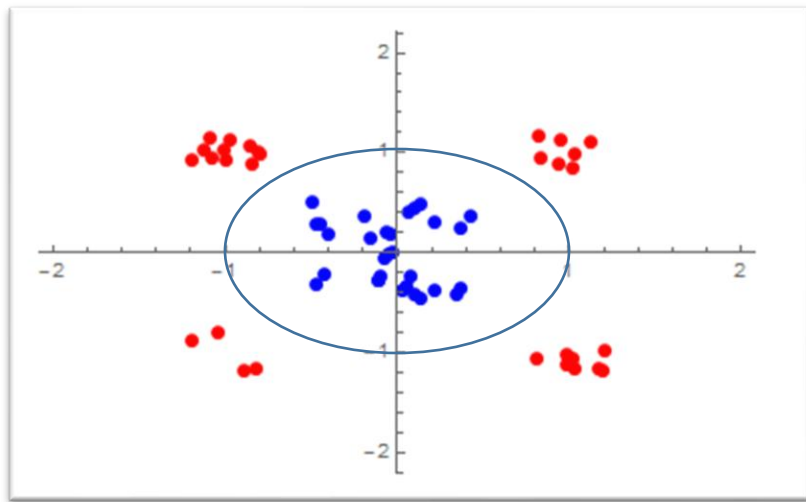
Données non linéairement séparables

Un perceptron se voit-il toujours le même nombre d'itérations de mise à jour ? 😊

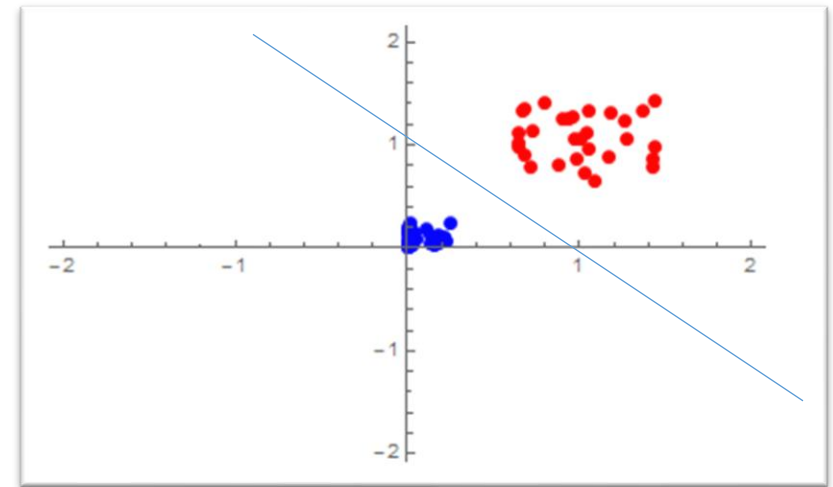
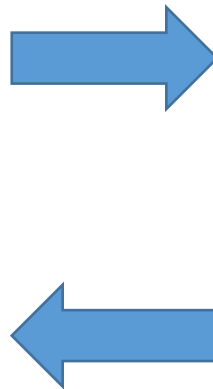


Données non linéairement séparables

Nous avons choisi cette transformation en particulier ...



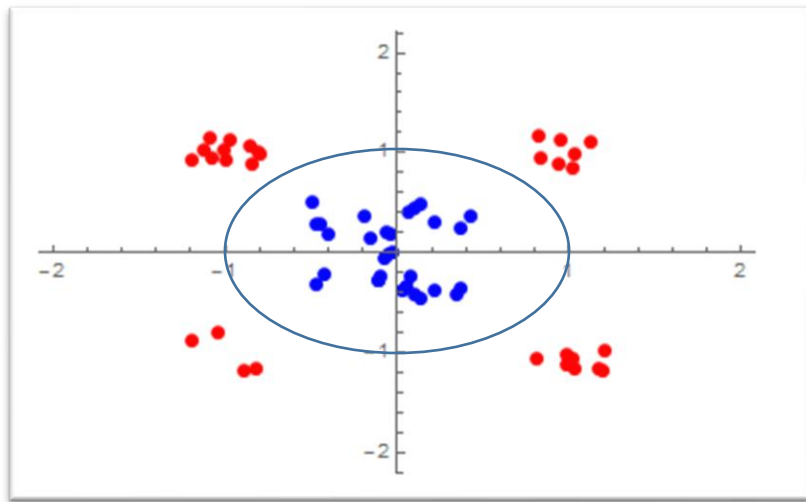
$$X = (x, y, 1)$$



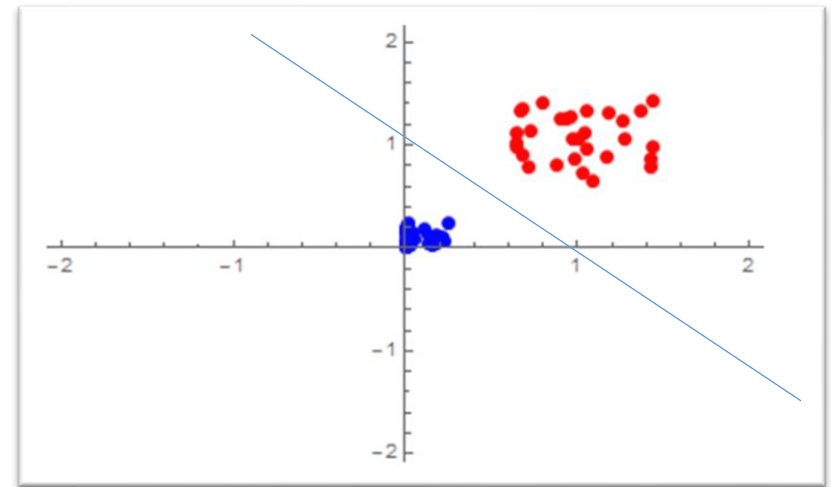
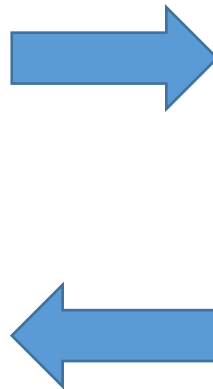
$$X = (x^2, y^2, 1)$$

Données non linéairement séparables

Nous avons choisi cette transformation en particulier ...
... car nous avons observé les données !!!



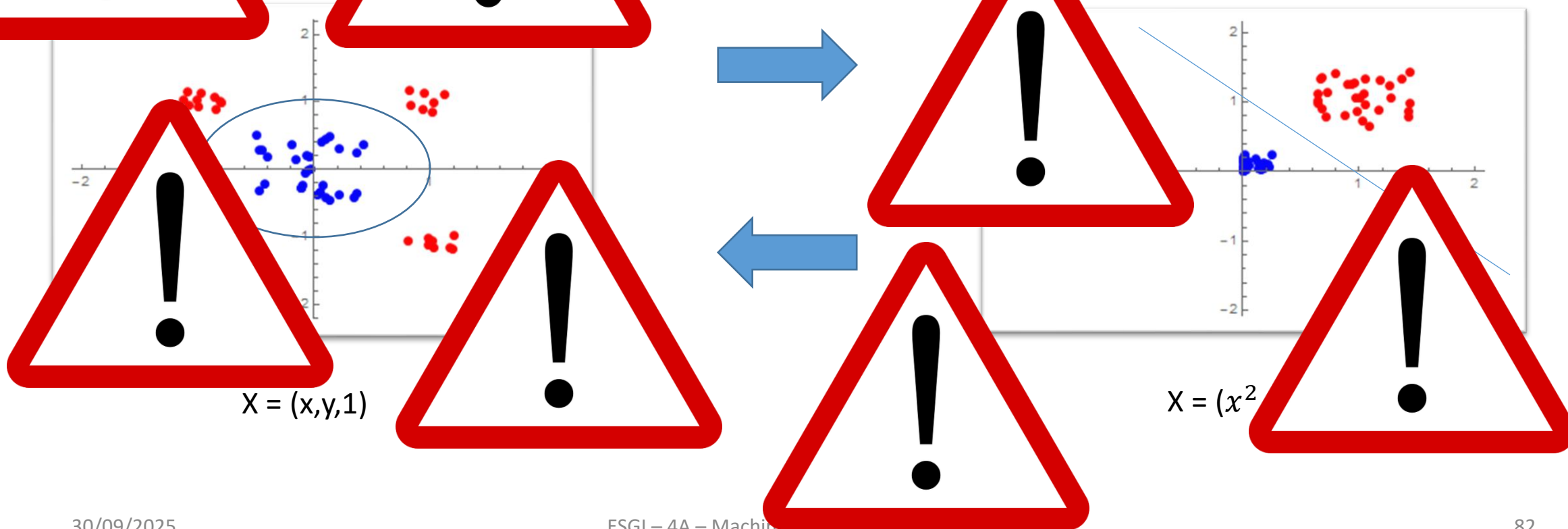
$$X = (x, y, 1)$$



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

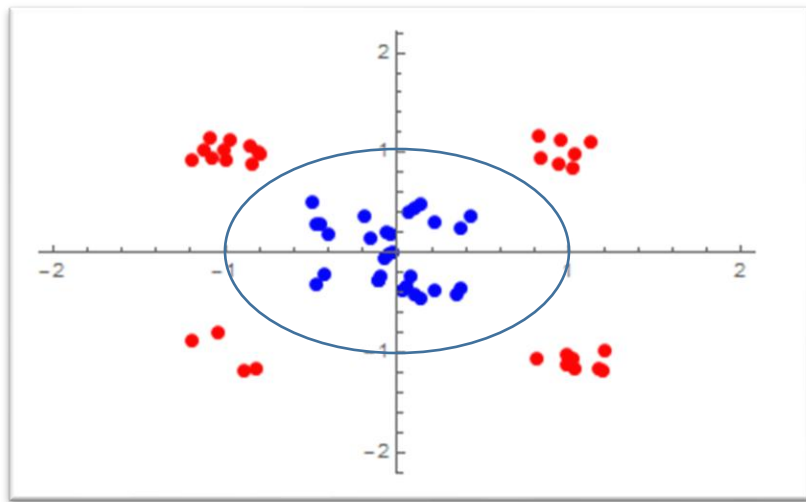
Données non linéairement séparables

avons choisi une transformation en particulier ...
car nous n'avons pas préservé les données !

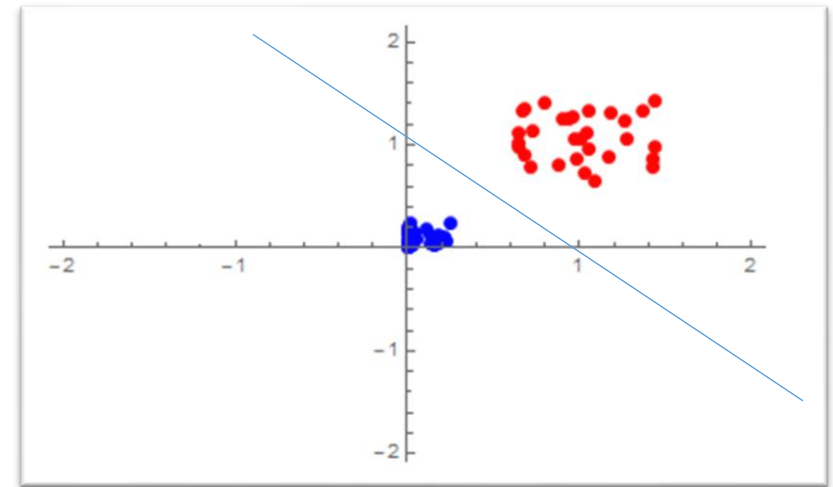
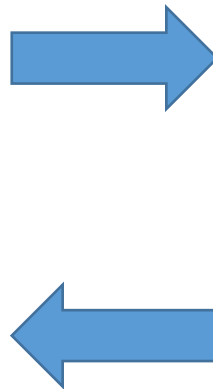


Données non linéairement séparables

Entrées réelles :



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

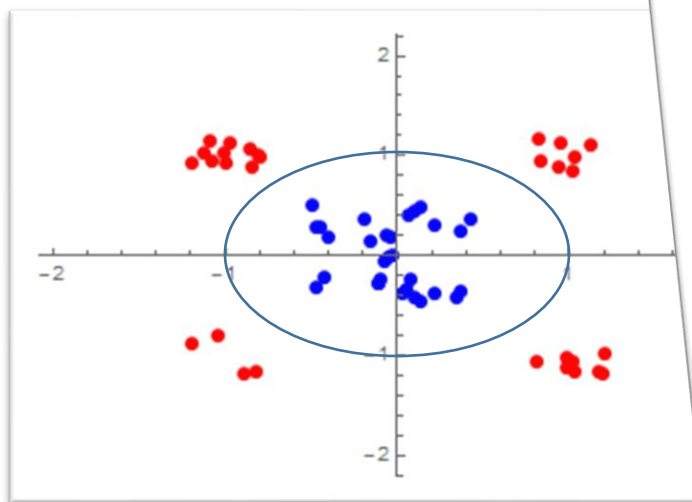


$$X = (1, x^2, y^2) \longleftrightarrow X = (1, x, y, xy, x^2, y^2)$$

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

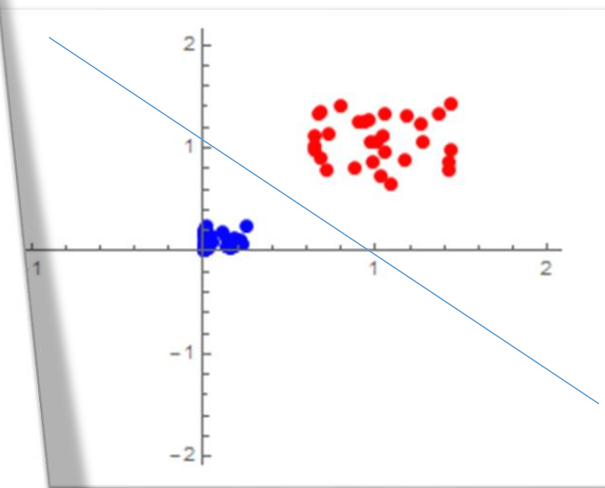
Données non linéairement séparables

Entrées réelles :



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

Real Algorithms
Don't require
Human intervention
to Learn !



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

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

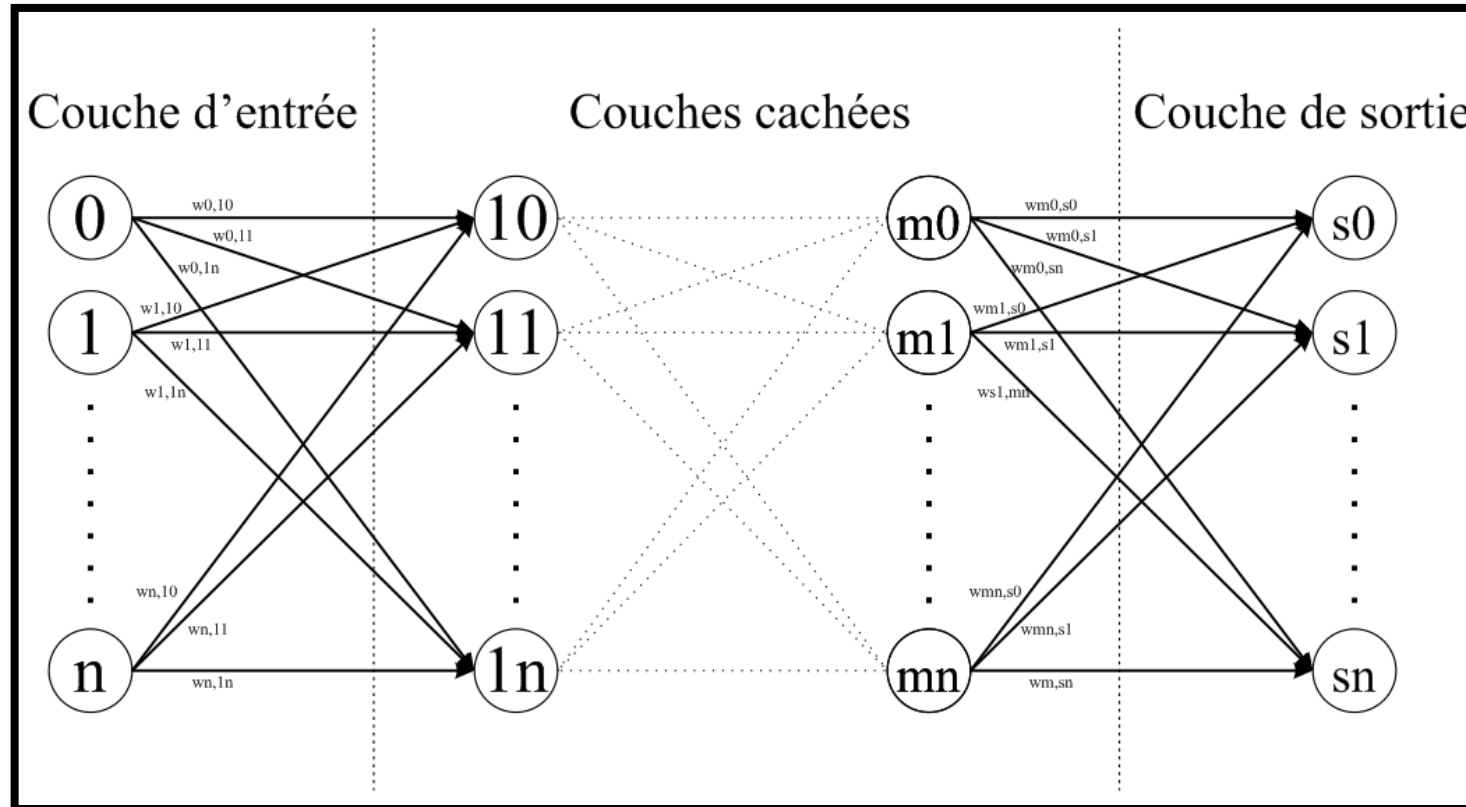
Données non linéairement séparables

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

Perceptron Multi Couches

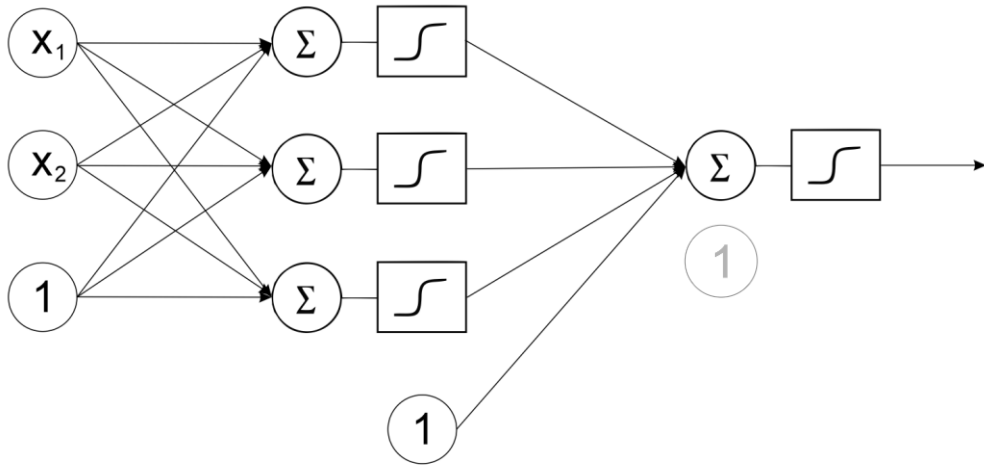
Principes

Intuition : mettre des perceptrons en série et en parallèle ...

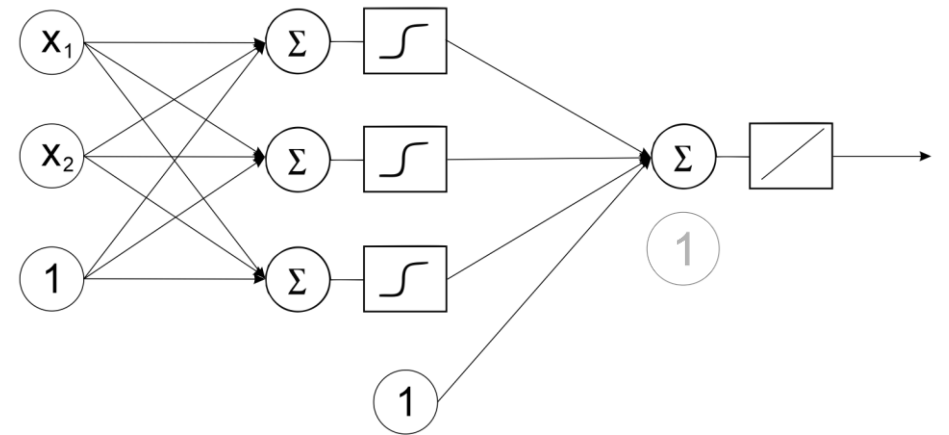


Principes

Intuition : mettre des perceptrons en série et en parallèle ...



Perceptron multi couches pour la classification

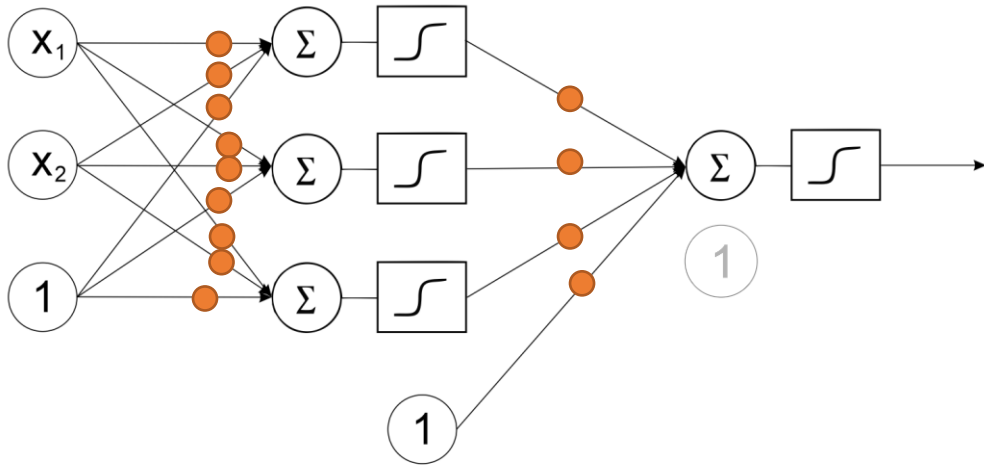


Perceptron multi couches pour la régression

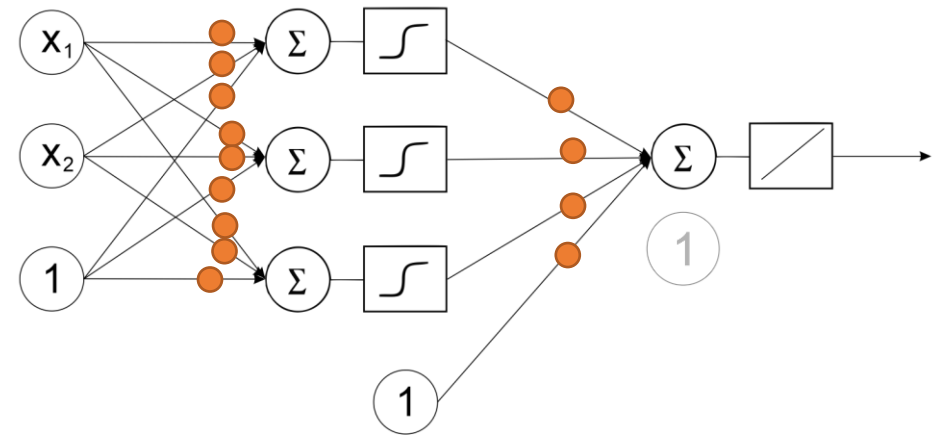
Principes

Intuition : mettre des perceptrons en série et en parallèle ...

Paramètres : ensemble des poids



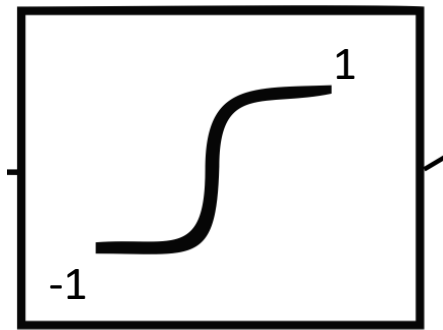
Perceptron multi couches pour la classification



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:

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

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 :

$$\delta_j^L = (1 - (x_j^L)^2) \times (x_j^L - y_j)$$

- En déduire pour tous les autres neurones de l'avant dernière couche à la première :

$$\delta_i^{l-1} = (1 - (x_i^{l-1})^2) \times \sum_{j=1}^{d^l} (w_{ij}^l \times \delta_j^l)$$

- 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 :

$$\delta_i^{l-1} = (1 - (x_i^{l-1})^2) \times \sum_{j=1}^{d^l} (w_{ij}^l \times \delta_j^l)$$

- Puis mettre à jour tous les w_{ij}^l :

$$w_{ij}^l \leftarrow w_{ij}^l - \alpha x_i^{l-1} \delta_j^l$$

RBF Networks

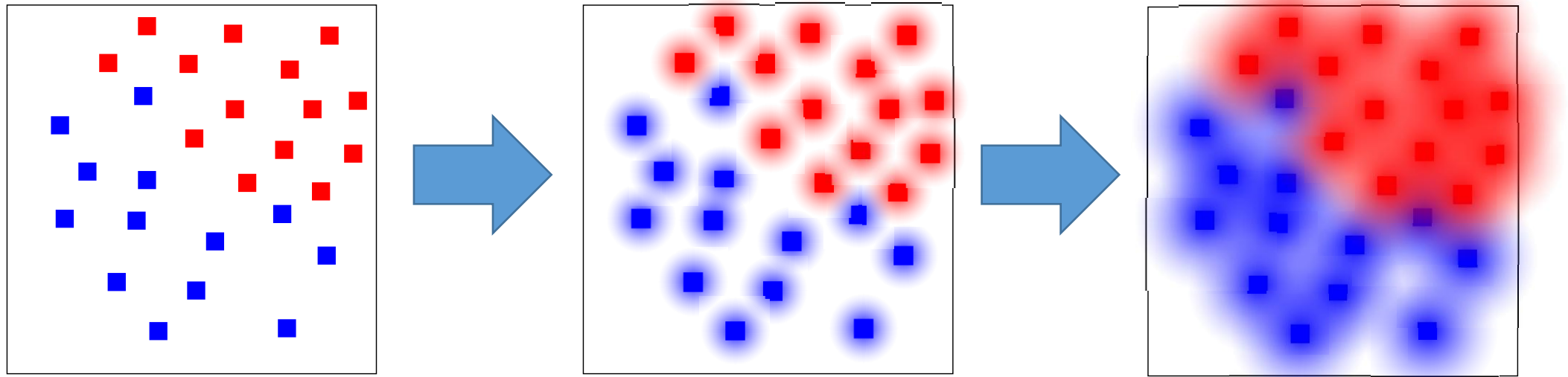
Intuition

Retours sur l'apprentissage « par cœur »

Intuition

Retours sur l'apprentissage « par cœur »

Conserver les exemples et attribuer une 'zone d'influence'



Principes

Régression RBF Naïf :

$$output(x) = \sum_{n=1}^N w_n e^{-\gamma ||X - X_n||^2}$$

Classification RBF Naïf :

$$output(x) = sign(\sum_{n=1}^N w_n e^{-\gamma ||X - X_n||^2})$$

Principes

Régression RBF Naïf :

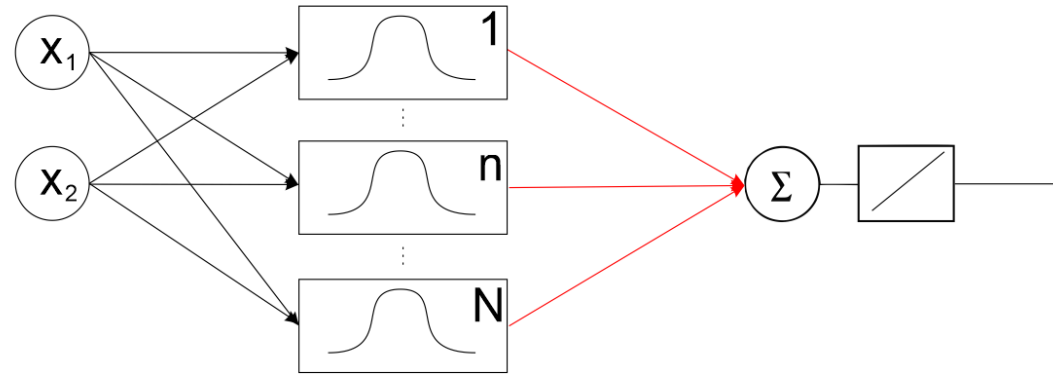
$$output(x) = \sum_{n=1}^N w_n e^{-\gamma ||X - X_n||^2}$$

Classification RBF Naïf :

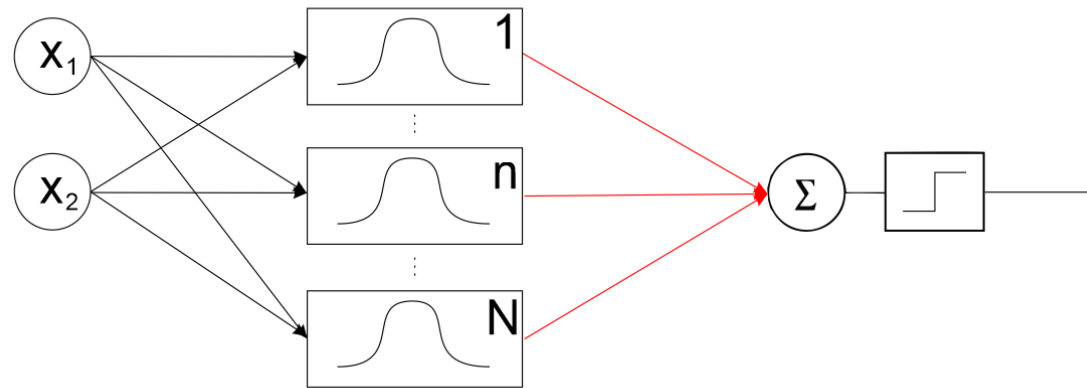
$$output(x) = sign(\sum_{n=1}^N w_n e^{-\gamma ||X - X_n||^2})$$

Principes

Régression RBF Naïf :



Classification RBF Naïf :



Principes :

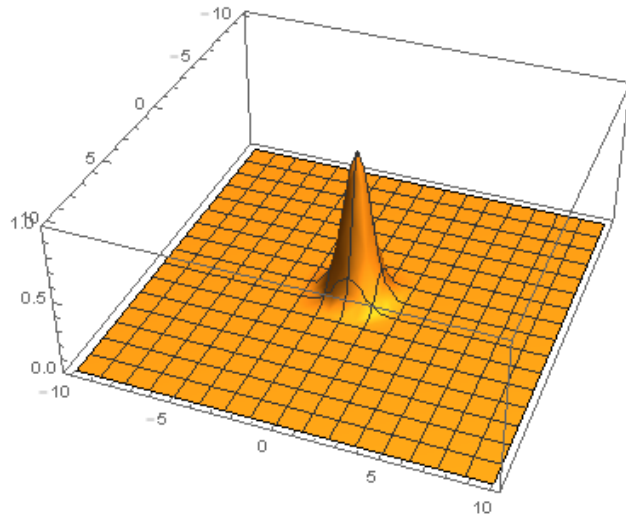
Trouver W pour un RBF naïf :

$$\text{Soit } \phi = \begin{bmatrix} e^{-\gamma \|X_1 - X_1\|^2} & \dots & e^{-\gamma \|X_1 - X_N\|^2} \\ \vdots & \ddots & \vdots \\ e^{-\gamma \|X_N - X_1\|^2} & \dots & e^{-\gamma \|X_N - X_N\|^2} \end{bmatrix} \text{ et } Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix}$$

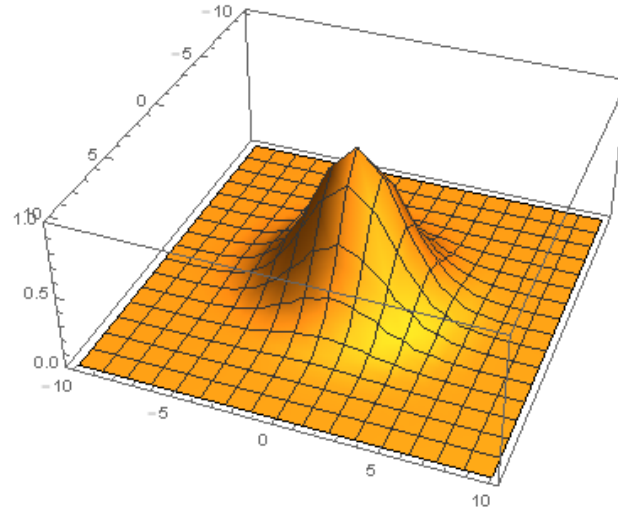
$$\text{Alors } W = \phi^{-1} Y$$

Principes :

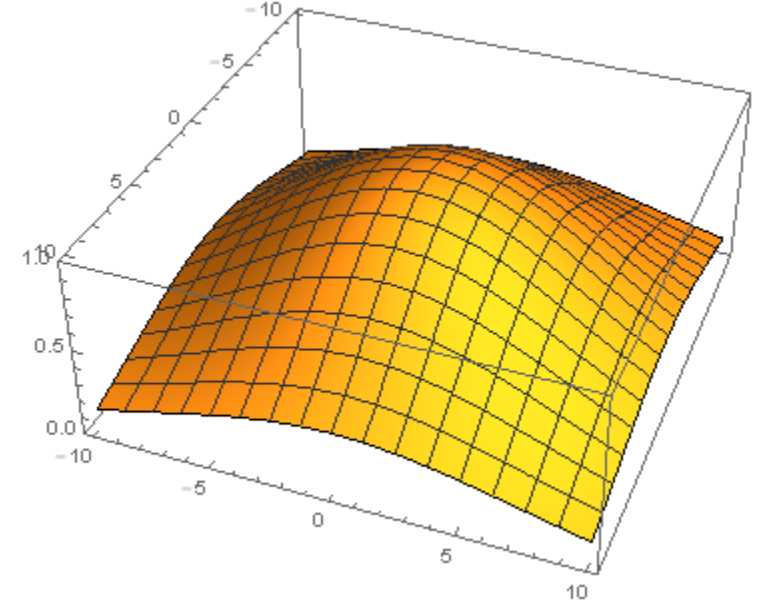
Impact du choix de Gamma :



$$\gamma = 1$$



$$\gamma = 0,1$$



$$\gamma = 0,01$$

Principes

Plus on a d'exemples à disposition, mieux c'est ?

Principes

En a d'exem disposition, mieux c'est ?



Principes

Plus on a d'exemples à disposition, mieux c'est ?

Nombre de w_i = nombre d'exemples !



Principes

Plus on a d'exemples à disposition, mieux c'est ?

Nombre de w_i = nombre d'exemples !

Mauvais signe pour la généralisation.



Intuition

Ne pas prendre tous les exemples !

Intuition

Ne pas prendre tous les exemples !

Elire des 'représentants'

Principes

k-Means

Méthode exacte : NP-Difficile !

Algorithme de LLoyd

Répéter :

1 :

$$\mu_k = \frac{1}{|S_k|} \sum_{x_n \in S_k} X_n$$

2 :

$$S_k = \{X_n \mid \forall l, \|X_n - \mu_k\| \leq \|X_n - \mu_l\|\}$$

RBF utilisant K centres

Trouver W pour un RBF utilisant K Centres :

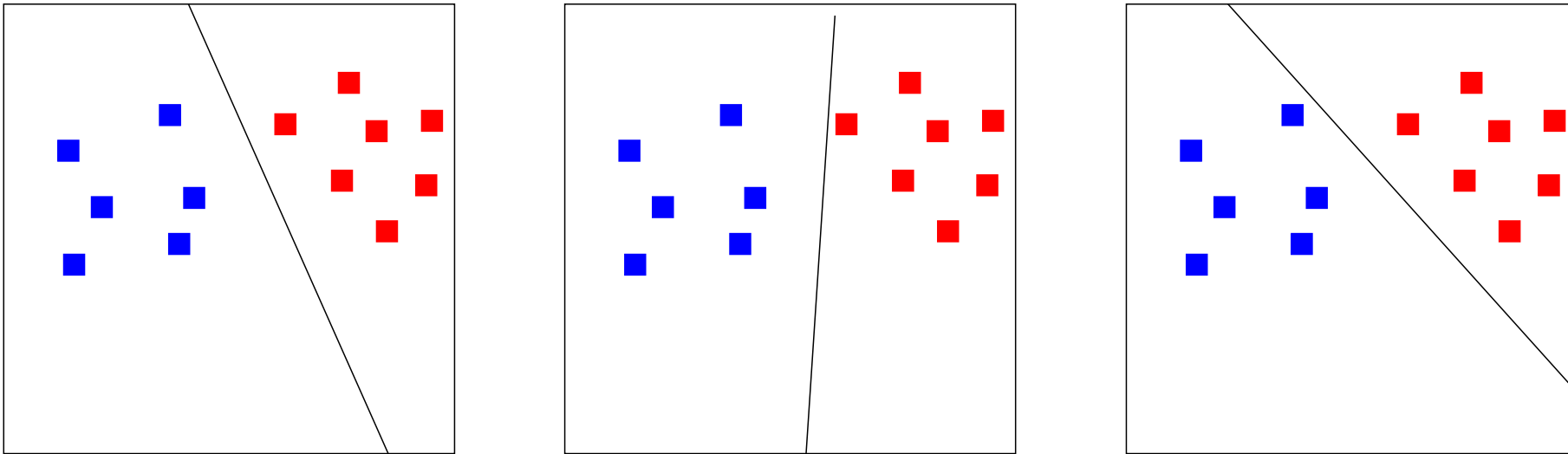
$$\text{Soit } \phi = \begin{bmatrix} e^{-\gamma \|x_1 - \mu_1\|^2} & \dots & e^{-\gamma \|x_1 - \mu_K\|^2} \\ \vdots & \ddots & \vdots \\ e^{-\gamma \|x_N - \mu_1\|^2} & \dots & e^{-\gamma \|x_N - \mu_K\|^2} \end{bmatrix} \text{ et } Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

$$\text{Alors } W = (\phi^T \phi)^{-1} \phi^T Y$$

SVM

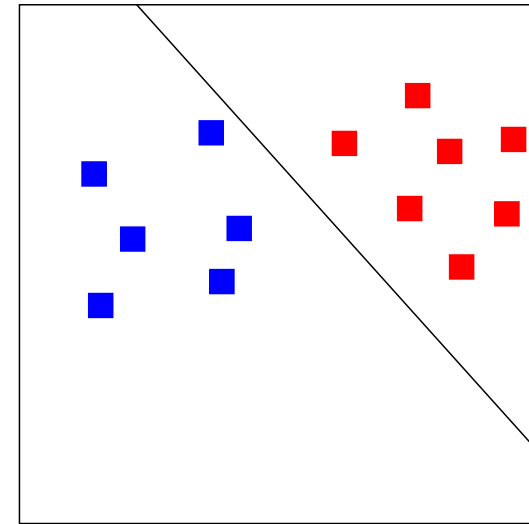
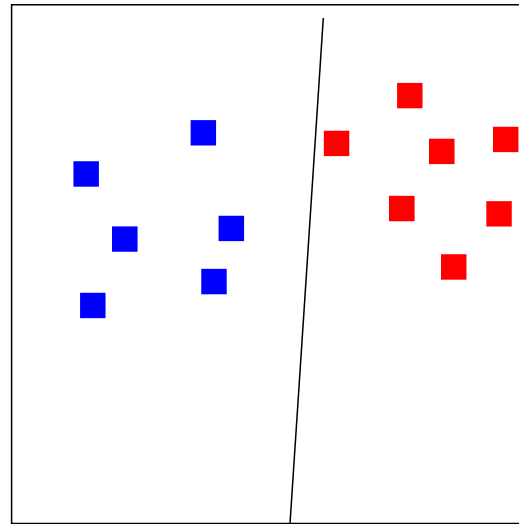
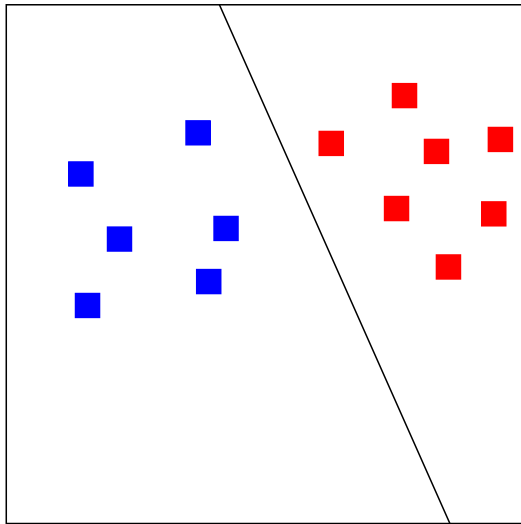
Séparation(s) linéaire(s)

Plusieurs (une infinité) de séparations linéaires possibles ...

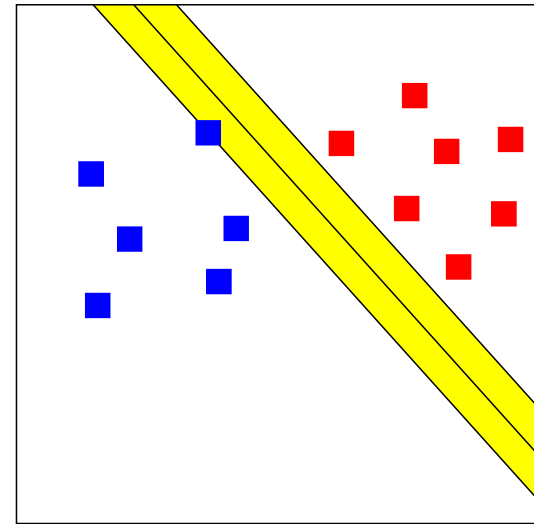
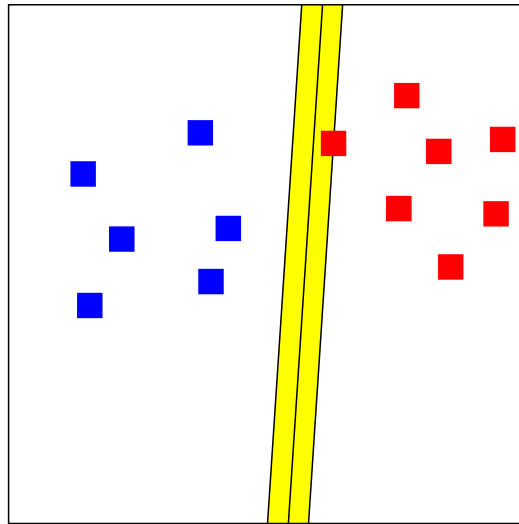
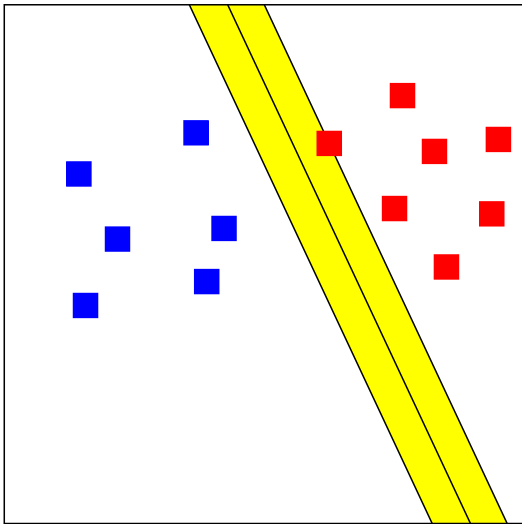


Séparation(s) linéaire(s)

Y en a-t-il une meilleure ?

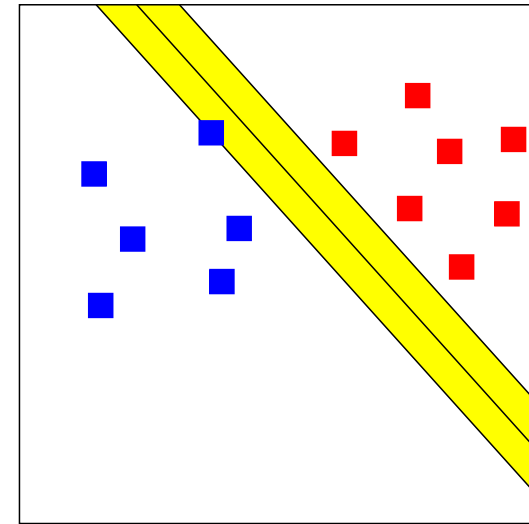
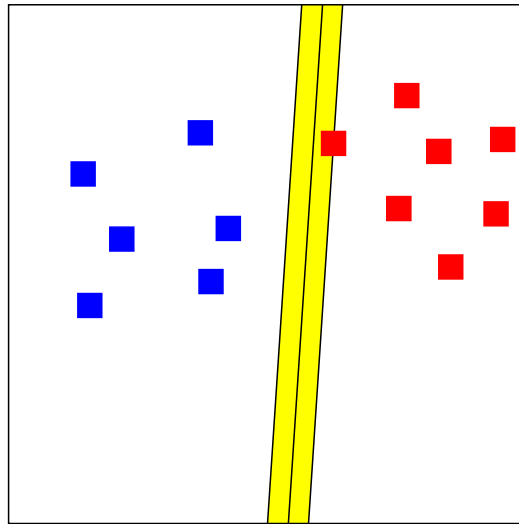
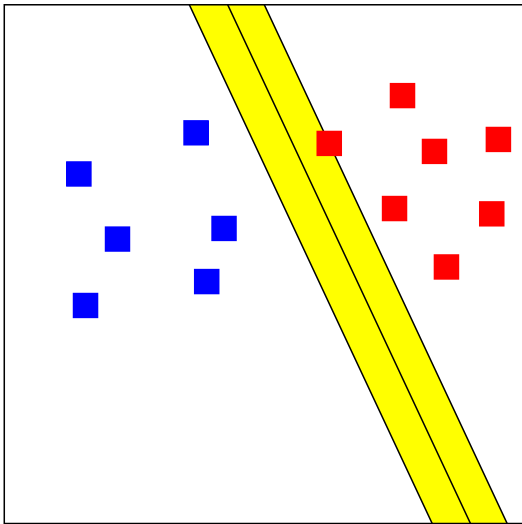


Notion de marge...



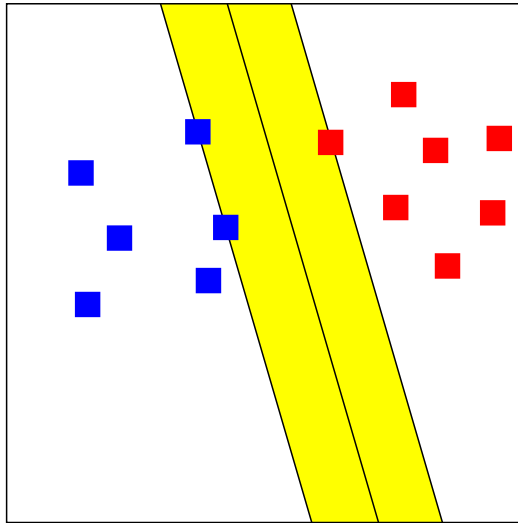
Notion de marge...

Y en a-t-il une meilleure ?



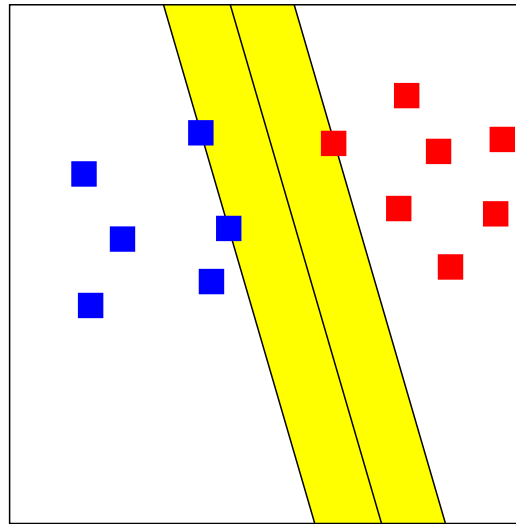
Notion de marge...

Y en a-t-il une meilleure ?



Notion de marge...

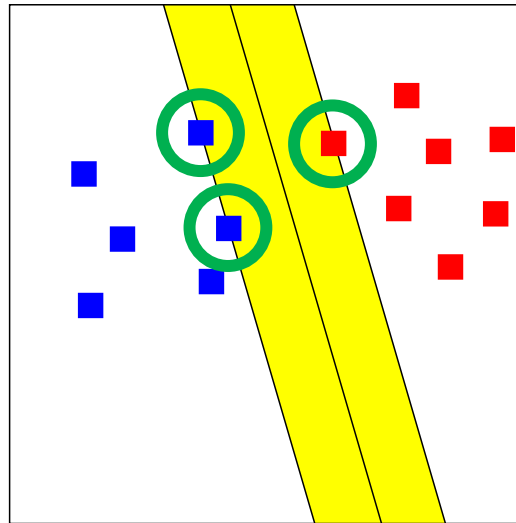
Nouveau problème : trouver les W qui maximisent la marge !



Notion de marge...

Nouveau problème : trouver les W qui maximisent la marge !

C.à.D. : Trouver les vecteurs supports :



Programmation quadratique à la rescousse !

Minimiser :

$$\frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 X_1^T X_1 & \cdots & y_1 y_N X_1^T X_N \\ \vdots & \ddots & \vdots \\ y_N y_1 X_N^T X_1 & \cdots & y_N y_N X_N^T X_N \end{bmatrix} \alpha + [-1 \quad \dots \quad -1] \alpha$$

Sous contraintes :

$$Y^T \alpha = 0$$

Avec :

$$\alpha \geq 0$$

Programmation quadratique à la rescousse !

Une fois α obtenu, on peut retrouver W :

$$W = \sum_{n=1}^N \alpha_n y_n X_n$$

Programmation quadratique à la rescousse !

Une fois α obtenu, on peut retrouver W :

$$W = \sum_{n=1}^N \alpha_n y_n X_n$$

Attention, il nous manque w_0 !

Programmation quadratique à la rescousse !

Pour trouver w_0 :

1 – Choisir un X_n tq $\alpha_n > 0$ c.à.d un **Vecteur Support** !

2 – Sachant que $y_n(W^T X_n + w_0) = 1$

3 – Cela nous donne :

$$w_0 = \frac{1}{y_n} - \sum_i w_i X_{ni}$$

Programmation quadratique à la rescousse !

Une fois α obtenu, on peut retrouver W :

$$W = \sum_{n=1}^N \alpha_n y_n X_n$$

N exemples \Rightarrow N paramètres ?

Programmation quadratique à la rescousse !

Une fois α obtenu, on peut retrouver W :

$$W = \sum_{n=1}^N \alpha_n y_n X_n$$

N exemples \Rightarrow N paramètres ?

Si X_i n'est pas un vecteur support, alors $\alpha_i = 0$!

Programmation quadratique à la rescousse !

Une fois α obtenu, on peut retrouver W :

$$W = \sum_{n=1}^N \alpha_n y_n X_n$$

N exemples \Rightarrow N paramètres ?

Si X_i n'est pas un vecteur support, alors $\alpha_i = 0$!

Ainsi, nous avons autant de paramètres dans notre modèle que de vecteur support \Rightarrow bonne généralisation !

Machine à noyaux

Retour sur les SVMs

Si nos exemples sont de grande dimension,

$$\begin{bmatrix} y_1 y_1 X_1^T X_1 & \cdots & y_1 y_N X_1^T X_N \\ \vdots & \ddots & \vdots \\ y_N y_1 X_N^T X_1 & \cdots & y_N y_N X_N^T X_N \end{bmatrix}$$

Sera difficile à calculer !

Retour sur les SVMs

Projection des entrées dans un autre espace (le retour) :

$$\begin{bmatrix} y_1 y_1 \mathbf{z}_1^T \mathbf{z}_1 & \cdots & y_1 y_N \mathbf{z}_1^T \mathbf{z}_N \\ \vdots & \ddots & \vdots \\ y_N y_1 \mathbf{z}_N^T \mathbf{z}_1 & \cdots & y_N y_N \mathbf{z}_N^T \mathbf{z}_N \end{bmatrix}$$

Si l'espace est de dimension supérieure à l'espace de départ, cela devrait être encore pire !

Retour su

Projection des

Si l'espace est
devrait être en



art, cela

Retour sur les SVMs

Projection des entrées dans un autre espace (le retour) :

$$\begin{bmatrix} y_1 y_1 \mathbf{z}_1^T \mathbf{z}_1 & \cdots & y_1 y_N \mathbf{z}_1^T \mathbf{z}_N \\ \vdots & \ddots & \vdots \\ y_N y_1 \mathbf{z}_N^T \mathbf{z}_1 & \cdots & y_N y_N \mathbf{z}_N^T \mathbf{z}_N \end{bmatrix}$$

Si l'espace est de dimension supérieure à l'espace de départ, cela devrait être encore pire !

Cela dépend du type de transformation !

Retour sur les SVMs

Cela dépend du type de transformation !

Nous n'avons besoin que de l'existence de la possibilité d'effectuer produit scalaire dans le nouvel espace !

$$\begin{bmatrix} y_1 y_1 K(X_1, X_1) & \cdots & y_1 y_N K(X_1, X_N) \\ \vdots & \ddots & \vdots \\ y_N y_1 K(X_N, X_1) & \cdots & y_N y_N K(X_N, X_N) \end{bmatrix}$$

Différents noyaux :

Noyau Polynomial de degré Q :

$$K(x_n, x_m) = (1 + x_n^T x_m)^Q$$

Noyau à Base Radiale :

$$K(x_n, x_m) = e^{-x_n^2} e^{-x_m^2} e^{2x_n x_m}$$

Equivalent à une
projection dans un
espace de
dimension infinie !

Noyaux :

Degré Q :

$$K(x_n, x_m) = (1 + x_n^T x_m)^Q$$

Sans augmentation
du nombre de
paramètres !

Noyau à Base Radiale :

$$K(x_n, x_m) = e^{-x_n^2} e^{-x_m^2} e^{2x_n x_m}$$

Conclusion

Recommandation et pièges



Généraliser
 \neq



Minimiser l'erreur sur les exemples



Recommandation et pièges



Quel modèle choisir ?

Privilégier les modèles simples avant tout !



L'explication (le modèle), la plus simple est la plus probable !

Quels autres modèles ?

Tous les modèles que nous avons vus sont utiles et continuent d'être améliorés !

Quels autres modèles ?

Le monde de l'apprentissage artificiel est très vaste ...



Overture

Apprentissage par renforcement

Très utile pour le Jeu (vidéo) !

- Peut être utilisé avec les modèles neuronaux que nous avons vu !
- Idéal dans les cas où les exemples étiquetés sont rares et arrivent au cours du temps.
- Q-Learning : indépendant du modèle

Metaheuristiques

Nous ne faisons qu'estimer des paramètres !

On peut utiliser un algorithme génétique pour trouver les poids d'un réseau de neurones à la place de la descente de gradient !

etc.

DeepLearning

La nouvelle classe de techniques 'à la mode', notamment les Convolutional Neural Networks,

Intrinsèquement liée au 'Big Data'

Etat des lieux :

<http://slideshot.epfl.ch/play/khnnunGF0elc>

Prochaines étapes de votre parcours :

Recommandations quasi-exhaustives pour le ML :

<http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf>

« Best MOOC Ever ! » :

<http://work.caltech.edu/telecourse.html>

Amusez-vous avec les frameworks !

Google a encore frappé !

https://www.tensorflow.org/get_started/get_started

Microsoft a suivi !

<https://cntk.codeplex.com/>

Facebook aussi !

<https://github.com/facebook/fbcunn>

N'oublions pas IBM !

<http://www.ibm.com/smarterplanet/us/en/ibmwatson/>

L'historique :

<http://scikit-learn.org/stable/>

Amusez-vous avec les frameworks !

Amazon s'y est mis aussi !

<https://github.com/amznlabs/amazon-dsstne>