3I026 - INTRODUCTION À L'INTELLIGENCE ARTIFICIELLE ET DATA SCIENCE

Vincent Guigue
Christophe Marsala

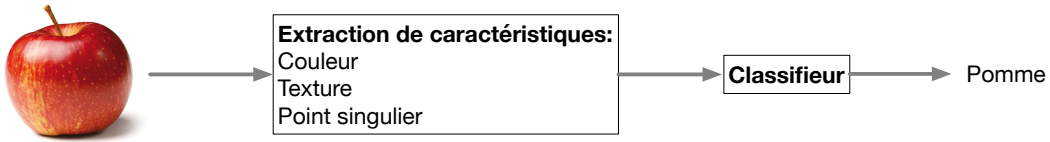
Sorbonne Université

SCIENCES SORBONNE UNIVERSITÉ

LIP6

MLIA Machine Learning & Deep Learning for Information Access

1. Représentation des entrées [Semaine 1 &2]



- Comment décrire un objet ? un signal ? une image ?...
- Représentation des données catégorielles dans une fonction de décision linéaire
- Normalisation des colonnes

2. k ppv : Premier algorithme d'apprentissage ! [Semaine 2 & 3]
   - Performant... Mais beaucoup trop lent !
3. Construction d'algorithmes de décision automatique [Semaine 3 & 4]
   - Perceptron
4. Evaluation de ces algorithmes

Déroulé sur la base Concrete (1)

Readme :

```
Name -- Data Type -- Measurement -- Description

Cement (component 1) -- kg in a m3 mixture -- Input
Blast Furnace Slag (component 2) -- kg in a m3 mixture -- Input
Fly Ash (component 3) -- kg in a m3 mixture -- Input
Water (component 4) -- kg in a m3 mixture -- Input
Superplasticizer (component 5) -- kg in a m3 mixture -- Input
Coarse Aggregate (component 6) -- kg in a m3 mixture -- Input
Fine Aggregate (component 7) -- kg in a m3 mixture -- Input
Age -- Day (1~365) -- Input
Concrete compressive strength -- MPa -- Output
```

Visualisation de X (premières lignes) :

```
[[ 540.    0.    0.  162.    2.5 1040.  676.   28.   79.99]
 [ 540.    0.    0.  162.    2.5 1055.  676.   28.   61.89]
 [ 332.5 142.5   0.  228.    0.   932.  594.  270.   40.27]
 [ 332.5 142.5   0.  228.    0.   932.  594.  365.   41.05]
 [ 198.6 132.4   0.  192.    0.   978.4 825.5 360.   44.3 ]
 [ 266.  114.    0.  228.    0.   932.  670.   90.   47.03]
 [ 380.   95.    0.  228.    0.   932.  594.  365.   43.7 ]
 [ 380.   95.    0.  228.    0.   932.  594.   28.   36.45]
 [ 266.  114.    0.  228.    0.   932.  670.   28.   45.85]
 [ 475.    0.    0.  228.    0.   932.  594.   28.   39.29]]
```

On ne peut pas visualiser un tableau avec plus d'une trentaine de valeurs...

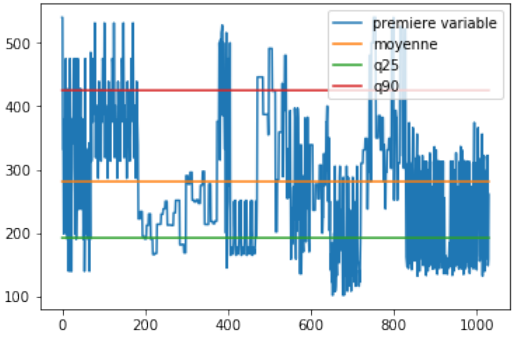
Premier contact : `pandas.describe`

	0	1	2	3	4	5	6	7	8
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

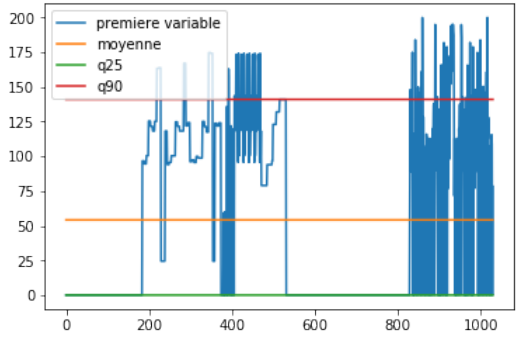
On ne peut pas visualiser un tableau avec plus d'une trentaine de valeurs...

Quelques détails un peu plus visuels :

Var0

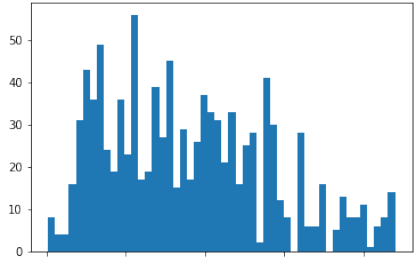


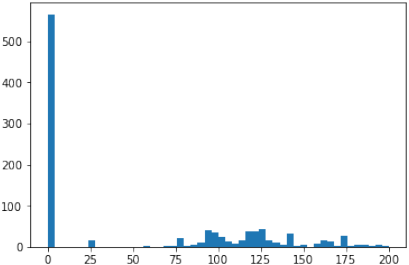
Var2

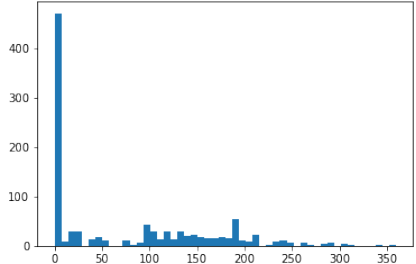


On dirait que les données sont regroupées par types de situations similaires

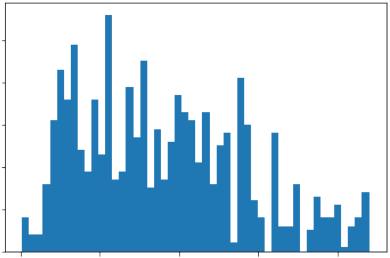
Déroulé sur la base Concrete (3)

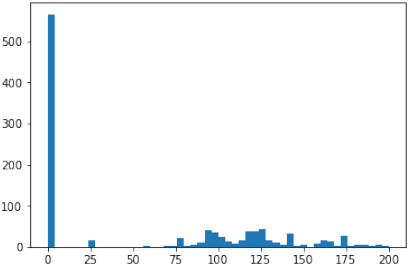


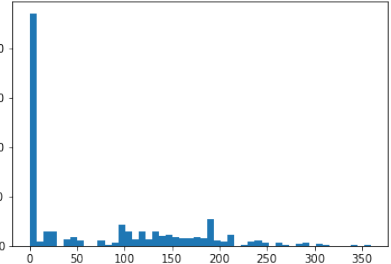


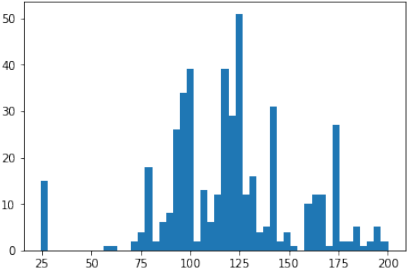


Déroulé sur la base Concrete (3)





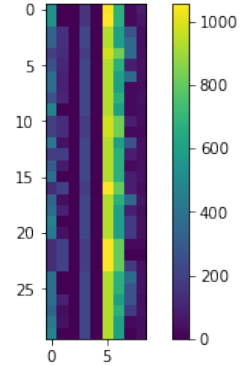




Différentes échelles, différentes distributions

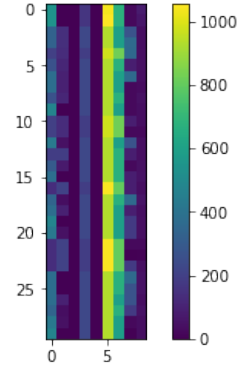
Que fait-on maintenant ? ? ?

Brutes :

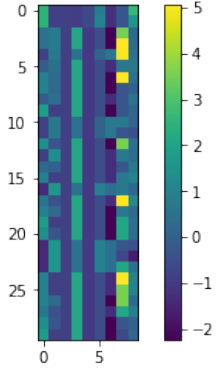


Que fait-on maintenant ? ? ?

Brutes :

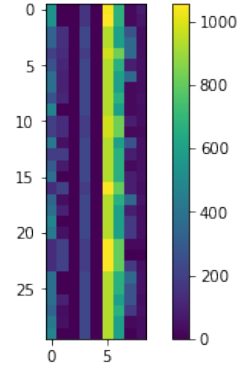


Norma.

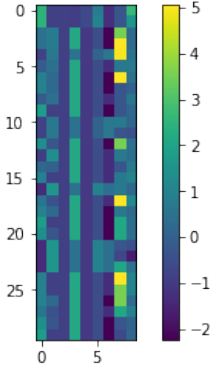


Que fait-on maintenant ? ? ?

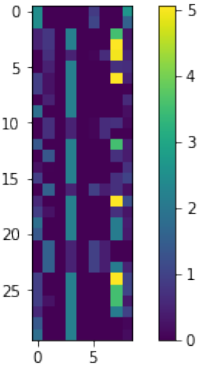
Brutes :



Norma.

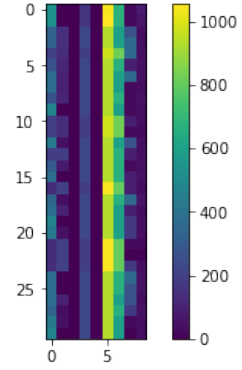


Norma. + 0

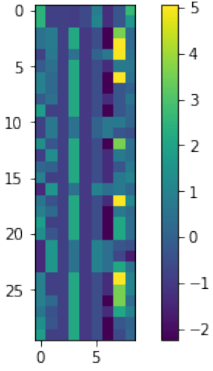


Que fait-on maintenant ? ? ?

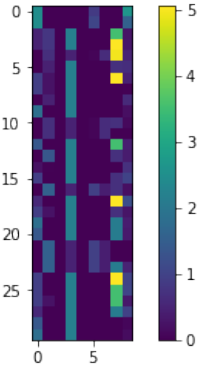
Brutes :



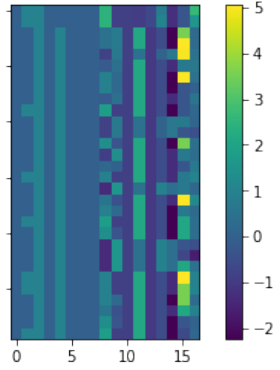
Norma.



Norma. + 0

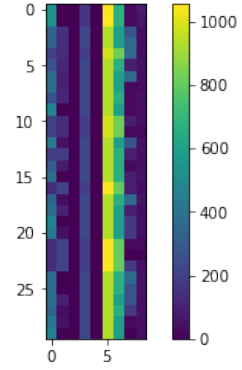


Bidouilles !

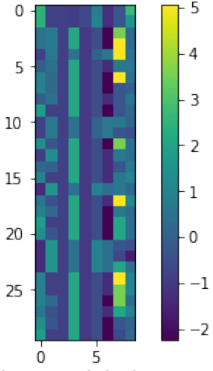


Que fait-on maintenant ? ? ?

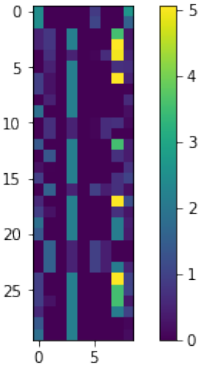
Brutes :



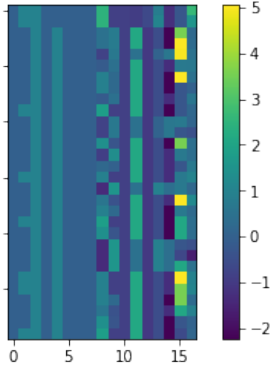
Norma.



Norma. + 0



Bidouilles !



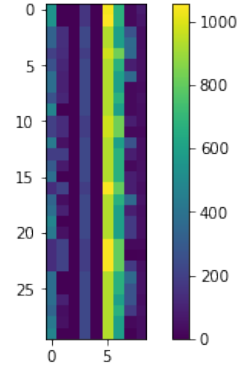
\+ Apprentissage d'un modèle linéaire :

Erreurs respectives :

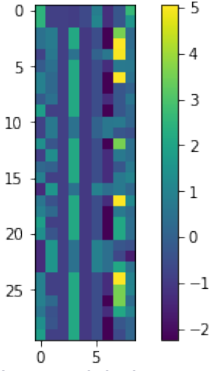
8.27	8.19	8.55	7.81

Que fait-on maintenant ? ? ?

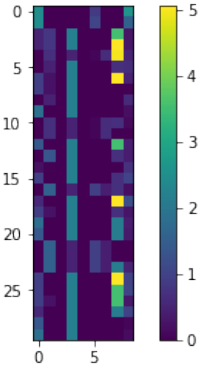
Brutes :



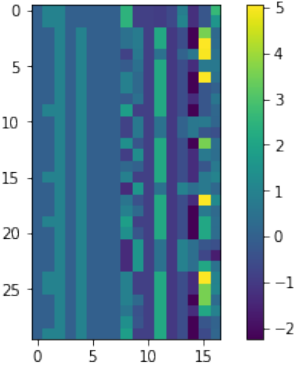
Norma.



Norma. + 0



Bidouilles !



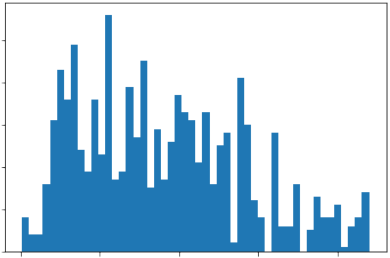
\+ Apprentissage d'un modèle linéaire :

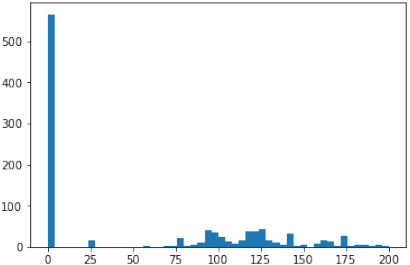
Erreurs respectives :

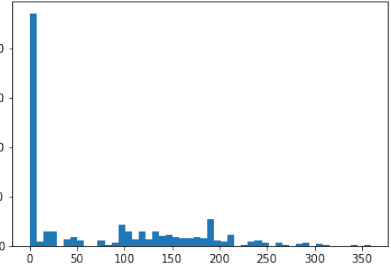
8.27 8.19 8.55 7.81

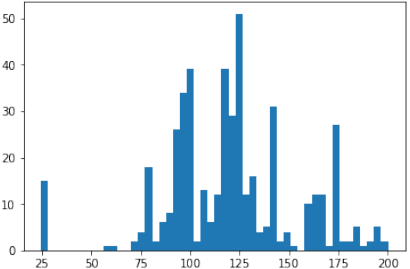
... A comparer avec ce que les gens avaient avant (modèle moyen) $\Rightarrow$ 13.46

(Retour) Déroulé sur la base Concrete (3)







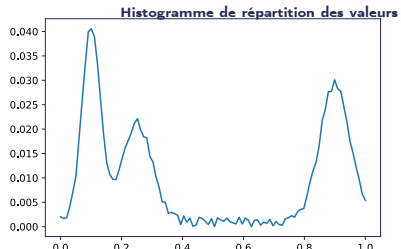


Différentes échelles, **différentes distributions**

- Si une variable X_i est discrète
- Si une variable X_i est continue... Mais avec des modes très prononcés

Reflexion sur les modèles linéaires : $f(\mathbf{x}) = \sum_j w_j x_j$ Comment coder x_j, quel impact sur f ?

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1i} = A & \cdots & x_{1d} \\ x_{21} & \cdots & x_{2i} = C & \cdots & x_{2d} \\ x_{31} & \cdots & x_{3i} = A & \cdots & x_{3d} \\ \vdots & \ddots & \vdots & & \\ x_{N1} & \cdots & x_{Ni} = B & \cdots & x_{Nd} \end{bmatrix} \in \mathbb{R}^{N \times d}$$



Conclusion

- Les histogrammes, c'est important
- Ne pas confondre les bidouilles et la triche...
  - Vérifier que vous êtes capable de traiter de nouveaux points
  - Notions d'apprentissage et de test
- Le dilemme du data-scientist :
  - Les performances se trouvent souvent dans les bidouilles
  - L'intérêt souvent dans les algorithmes d'apprentissage
- La problématique de l'évaluation...

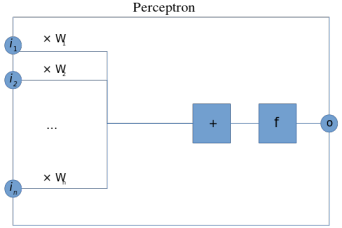
Modèle linéaire & Perceptron

data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

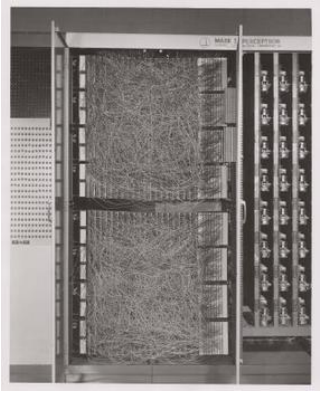
... = perceptron

Principe : initialisation aléatoire + correction en cas d'erreur

- 1957
- Frank Roseblatt



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

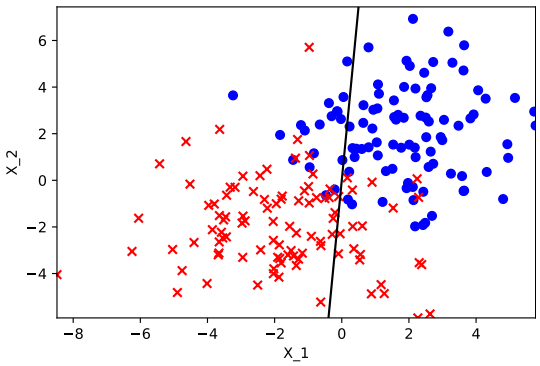
- Détecter une erreur
- Corriger
- Algorithme stochastique :
  paramètres & pièges

Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}$, $Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$

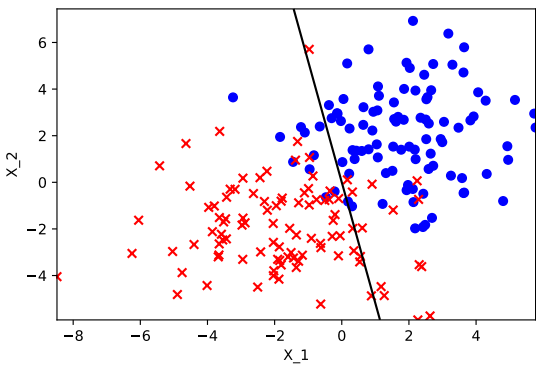
Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



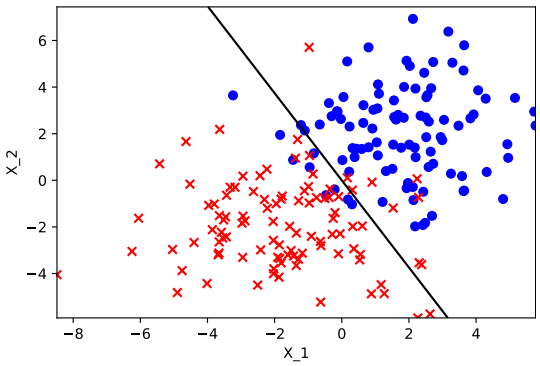
Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



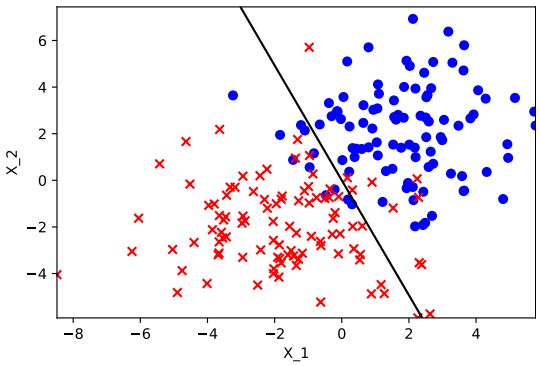
Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



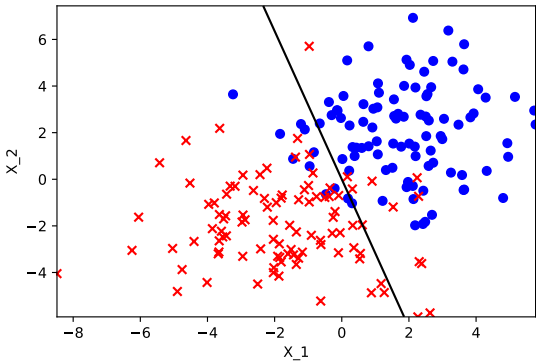
Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



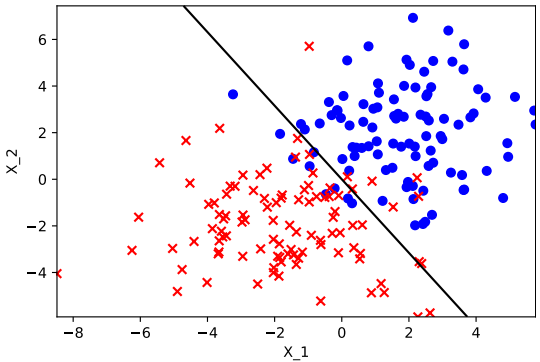
Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

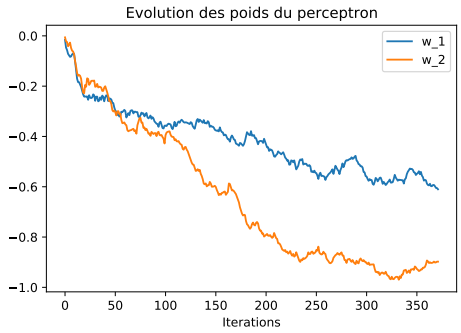
- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$



Soit des entrées étiquetées :
$X \in \mathbb{R}^{N \times d}, Y \in \{-1, 1\}^N$
Soit ϵ le pas de mise à jour (*learning rate*) et `niter_max` un paramètre de sécurité pour éviter les boucles infinies,

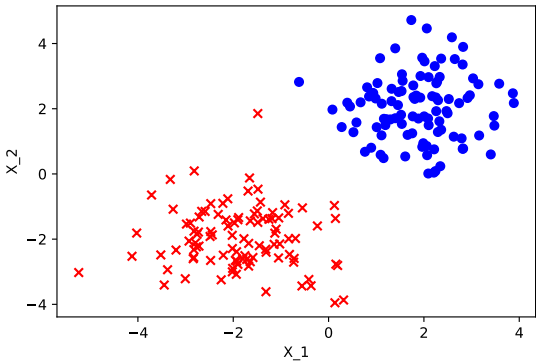
- Pour `niter_max` itérations
  - Initialiser $\mathbf{w}$
  - Tirer un point aléatoirement $\mathbf{x}_i$
  - Si le point est mal classé
    - $\mathbf{w} \leftarrow \mathbf{w} + \epsilon y_i \mathbf{x}_i$
  - Si $iteration \% N == 0$
    - critère de sortie = convergence = $\mathbf{w}$ ne bouge plus beaucoup
- Retourner $\mathbf{w}$

Quand s'arrêter ?



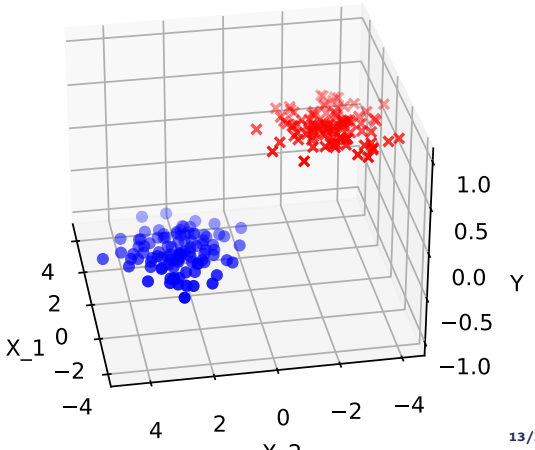
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



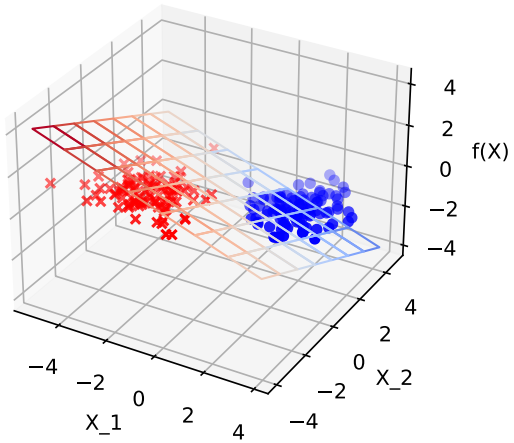
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



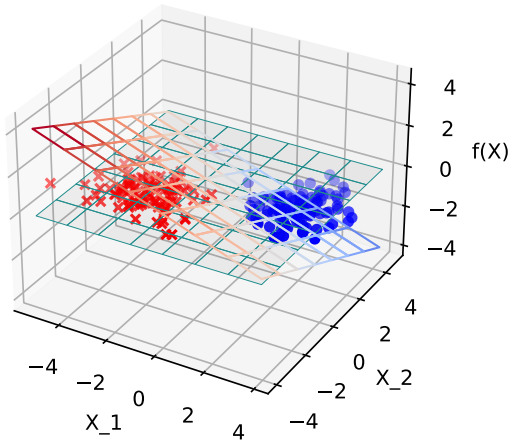
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



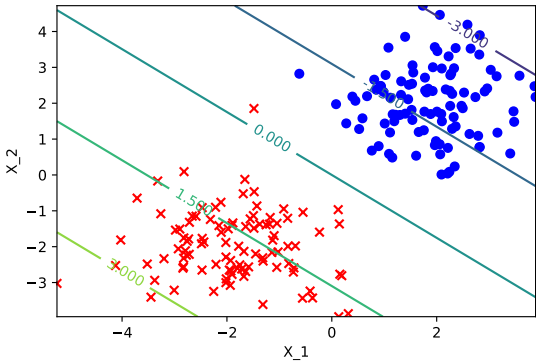
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

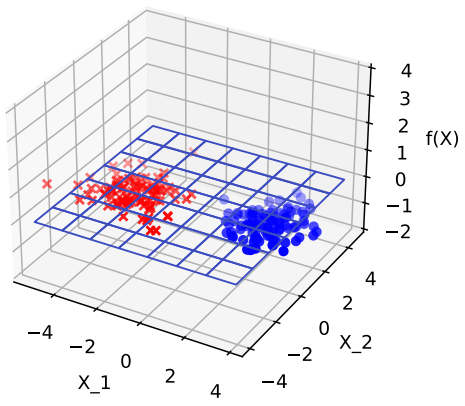
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.0 -0.0]

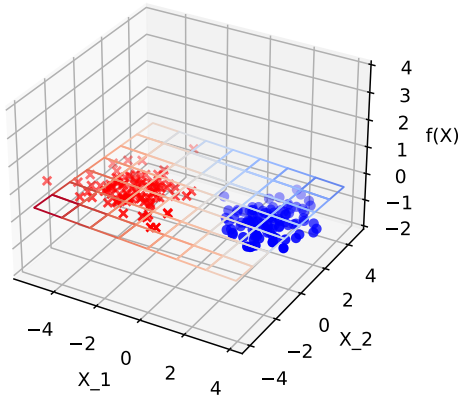
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.04 -0.05]

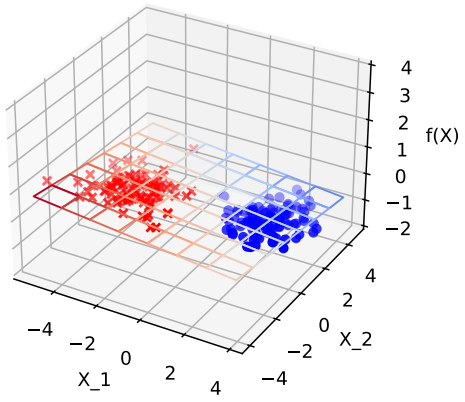
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.08 -0.1]

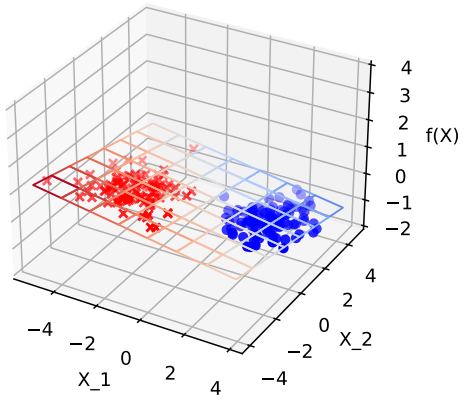
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.13 -0.15]

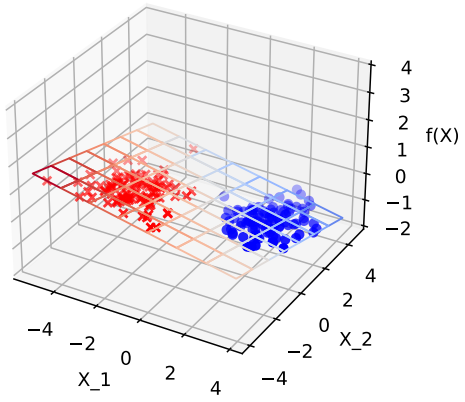
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.17 -0.19]

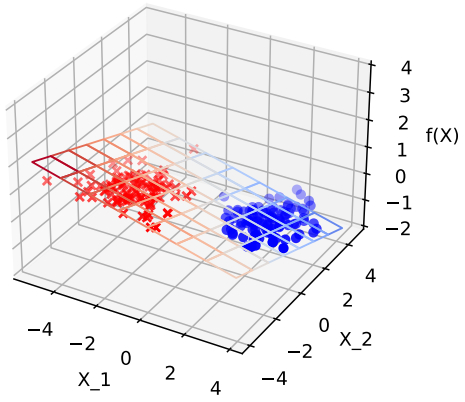
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.21 -0.24]

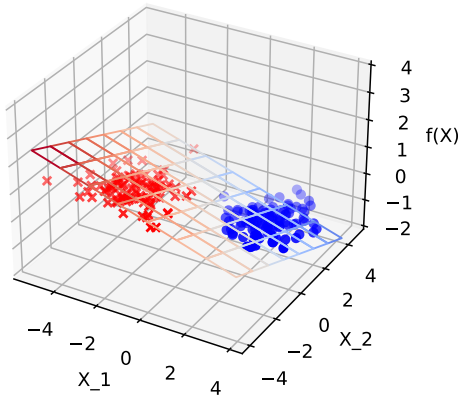
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.25 -0.29]

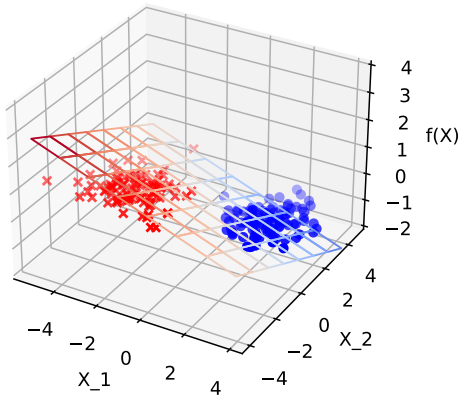
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.3 -0.34]

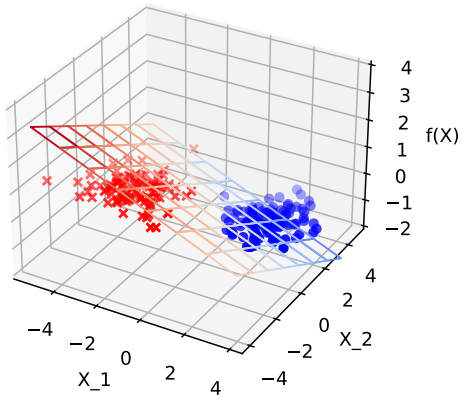
- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de **w**



data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

[-0.34 -0.39]

- Forme générale d'une fonction linéaire dans l'espace
- Forme de la frontière de décision
- Impact de la norme de $\mathbf{w}$



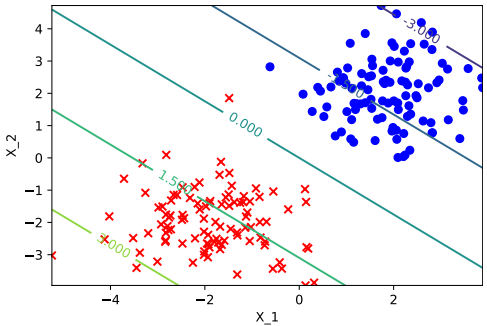
Extension non linéaire de l'algorithme

data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

Soit une matrice 2D d'observations :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \in \mathbb{R}^{N \times 2}$$

$$f(\mathbf{x}_i) = \mathbf{x}_i \cdot \mathbf{w} = \sum_j x_{ij} w_j$$



Frontière linéaire, passant par $(0, 0)$

data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

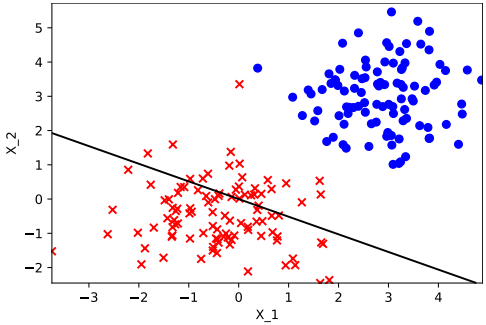
Si on décale les observations...

- rouge : $\mu = [0.5, 0.5]$
- bleu : $\mu = [3, 3]$

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \in \mathbb{R}^{N \times 2}$$

$$f(\mathbf{x}_i) = \mathbf{x}_i \cdot \mathbf{w} = x_{i1} w_1 + x_{i2} w_2$$

$\Rightarrow$ Frontière passant par $(0, 0)$



Meilleure frontière (après optimisation) ! !

$$\text{data} : \mathbf{x} = [x_1, \ldots, x_d], \text{ étiquette} : y = \{-1, 1\} \qquad f(\mathbf{x}) = \sum_j x_j w_j \approx y$$

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \in \mathbb{R}^{N \times 2}$$

$$f(\mathbf{x}_i) = \mathbf{x}_i \cdot \mathbf{w} = x_{i1} w_1 + x_{i2} w_2$$

$\Rightarrow$ Frontière passant par $(0, 0)$

Etudions la fonction...

data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

L'idée de base est **très** simple :
ajoutons des colonnes dans X et regardons la forme de la solution évoluer

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & 1 \\ \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & 1 \end{bmatrix}$$

$$f(\mathbf{x}_i) = x_{i1} w_1 + x_{i2} w_2 + w_3$$

La transformation est faisable sur n'importe quelle entrée

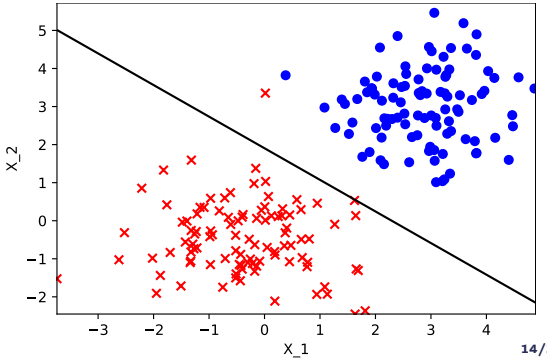
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

L'idée de base est **très** simple :
ajoutons des colonnes dans X et regardons la forme de la solution évoluer

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & 1 \\ \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & 1 \end{bmatrix}$$

$$f(\mathbf{x}_i) = x_{i1} w_1 + x_{i2} w_2 + w_3$$

La transformation est faisable sur n'importe quelle entrée



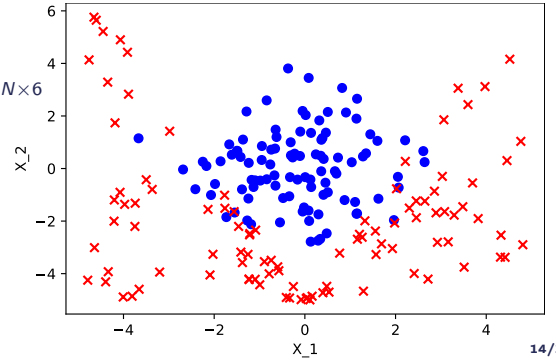
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

L'idée de base est **très** simple :
ajoutons des colonnes dans X et regardons la forme de la solution évoluer

$$X^* = \begin{bmatrix} x_{11} & x_{12} & x_{11}^2 & x_{12}^2 & x_{11}x_{12} & 1 \\ \vdots & \ddots & & & & \vdots \\ x_{N1} & x_{N2} & x_{N1}^2 & x_{N2}^2 & x_{N1}x_{N2} & 1 \end{bmatrix} \in \mathbb{R}^{N \times 6}$$

La fonction $f(\mathbf{x}^*) = \mathbf{w} \cdot \mathbf{x}^*$ correspond à une frontière linéaire dans l'espace 6D... Et une frontière non linéaire dans l'espace d'origine !

Sur un problème plus dur !



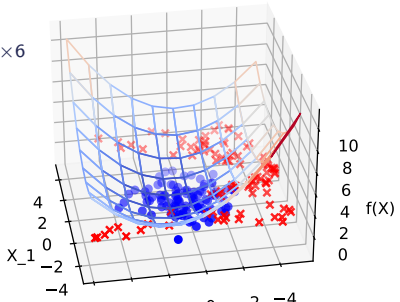
data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

L'idée de base est **très** simple :
ajoutons des colonnes dans X et regardons la forme de la solution évoluer

$$X^* = \begin{bmatrix} x_{11} & x_{12} & x_{11}^2 & x_{12}^2 & x_{11}x_{12} & 1 \\ \vdots & \ddots & & & & \vdots \\ x_{N1} & x_{N2} & x_{N1}^2 & x_{N2}^2 & x_{N1}x_{N2} & 1 \end{bmatrix} \in \mathbb{R}^{N\times 6}$$

La fonction $f(\mathbf{x}^*) = \mathbf{w} \cdot \mathbf{x}^*$ correspond à une frontière linéaire dans l'espace 6D... Et une frontière non linéaire dans l'espace d'origine !

[0.07 -0.27]



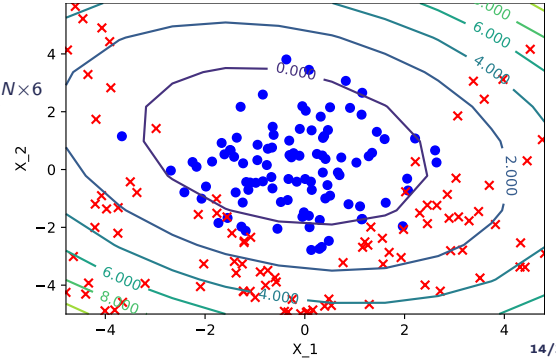
Sur un problème plus dur !

data : $\mathbf{x} = [x_1, \ldots, x_d]$, étiquette : $y = \{-1, 1\}$ $\quad f(\mathbf{x}) = \sum_j x_j w_j \approx y$

L'idée de base est **très** simple :
ajoutons des colonnes dans X et regardons la forme de la solution évoluer

$$X^* = \begin{bmatrix} x_{11} & x_{12} & x_{11}^2 & x_{12}^2 & x_{11}x_{12} & 1 \\ \vdots & \ddots & & & & \vdots \\ x_{N1} & x_{N2} & x_{N1}^2 & x_{N2}^2 & x_{N1}x_{N2} & 1 \end{bmatrix} \in \mathbb{R}^{N\times 6}$$

La fonction $f(\mathbf{x}^*) = \mathbf{w} \cdot \mathbf{x}^*$ correspond à une frontière linéaire dans l'espace 6D... Et une frontière non linéaire dans l'espace d'origine !

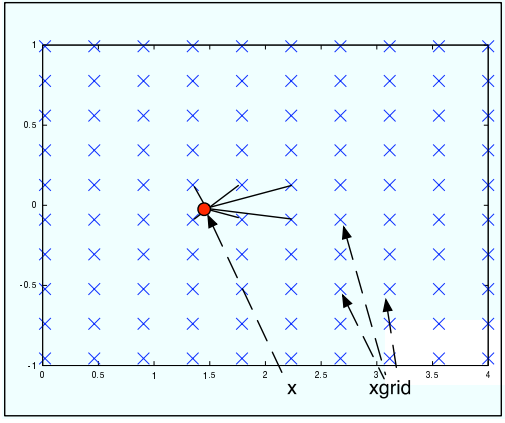


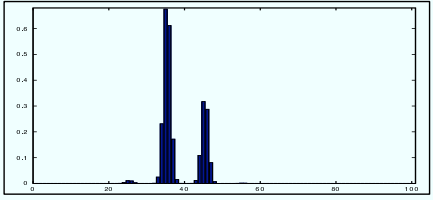
Sur un problème plus dur !

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & x_{11}^2 & x_{12}^2 & x_{11}x_{12} & 1 \\ \vdots & \ddots & & & & \vdots \\ x_{N1} & x_{N2} & x_{N1}^2 & x_{N2}^2 & x_{N1}x_{N2} & 1 \end{bmatrix} \in \mathbb{R}^{N\times 6}$$

1. Les plots sont sur X (ou sur les premières dimensions de X^*)
2. $\mathbf{w}$ est de la dimension de X^* (pas de X)
3. Pour traiter un nouveau point $\mathbf{x}$, il faut lui appliquer une transformation...
   Sinon, il n'est pas compatible en dimension avec $\mathbf{w}$

Vers des espaces encore plus compliqués...





Représentation en histogramme des similarités gaussiennes entre un point (rond rouge) et tous les points de la grille

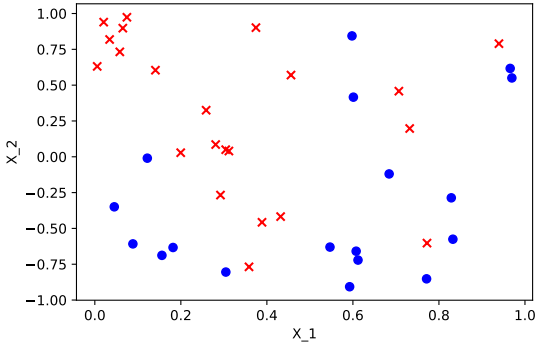
$$\phi(\mathbf{x})_j = \exp(-\frac{\|\mathbf{x} - \mathbf{xgrid_j}\|^2}{2\sigma^2})$$

Pour des problèmes encore plus compliqués...
Construire une zone d'influence autour d'un point d'apprentissage $\mathbf{x}_i$:

$$\forall \mathbf{x},\ k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$$

Idée : ajouter une colonne :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & k(\mathbf{x}_1, \mathbf{x}_i) \\ \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & k(\mathbf{x}_N, \mathbf{x}_i) \end{bmatrix}$$

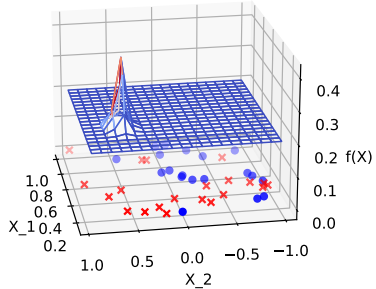


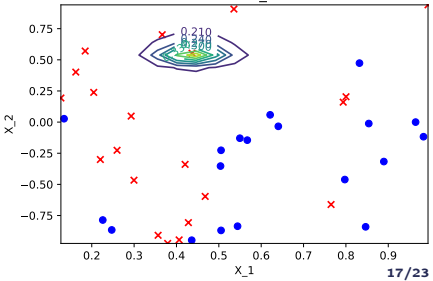
Pour des problèmes encore plus compliqués...
Construire une zone d'influence autour d'un point d'apprentissage $\mathbf{x}_i$:

$$\forall \mathbf{x},\ k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$$

Idée : ajouter une colonne :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & k(\mathbf{x}_1, \mathbf{x}_i) \\ \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & k(\mathbf{x}_N, \mathbf{x}_i) \end{bmatrix}$$





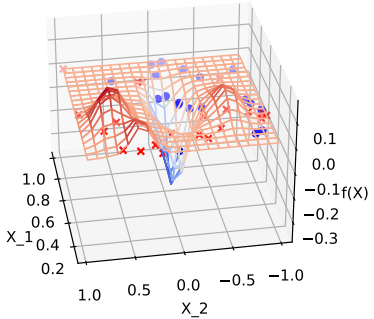
Pour des problèmes encore plus compliqués...
Construire une zone d'influence autour d'un point d'apprentissage $\mathbf{x}_i$:

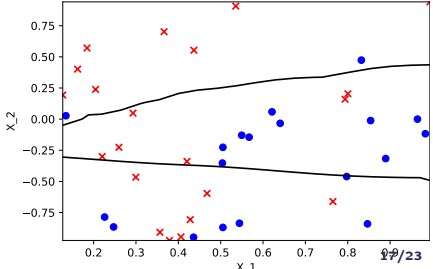
$$\forall \mathbf{x},\ k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$$

Idée : ajouter une colonne :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} x_{11} & x_{12} & k(\mathbf{x}_1, \mathbf{x}_i) \\ \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & k(\mathbf{x}_N, \mathbf{x}_i) \end{bmatrix}$$

puis plusieurs colonnes...





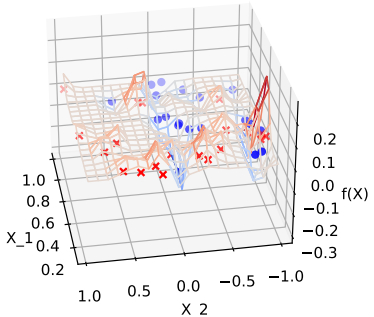
Pour des problèmes encore plus compliqués...
Construire une zone d'influence autour d'un point d'apprentissage $\mathbf{x}_i$:

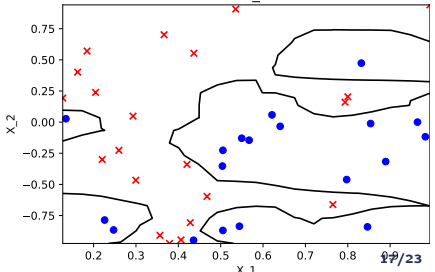
$$\forall \mathbf{x},\ k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right)$$

Idée : ajouter une colonne :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

puis autant de colonnes que de points d'apprentissage...
Les $\mathbf{w} \in \mathbb{R}^N$ viennent pondérer les Gaussiennes centrées sur les points d'apprentissage.





- Il faut toujours vérifier que vous êtes capable de projeter les nouveaux points... Avec les gaussiennes, ce n'est pas évident (mais ça marche) :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix} \Rightarrow X^* = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}, \qquad \mathbf{w} \in \mathbb{R}^N$$

$$k : \mathbf{x} \rightarrow \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \end{bmatrix}$$

- Plus l'espace est complexe, plus le risque de sur-apprentissage est grand
- Un noyau Gaussien est capable de bien classé 100% des points d'apprentissage
- La transformation est chère en mémoire et en calcul (et le cout est quadratique en N)
- Il faut régler le paramètre σ en plus des paramètres d'apprentissage

Soit une base d'apprentissage constituée de N échantillons $(\mathbf{x}_i, y_i)_{i=1,\dots,N}$

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$

- k produit scalaire $\approx$ mesure de similarité

On dit qu'une application

$$(\ |\ ) : \begin{array}{ccc} E \times E & \to & \mathbb{R} \\ (x, y) & \mapsto & (x \mid y) \end{array}$$

est un **produit scalaire** si elle est :

- *bilinéaire* : ϕ est linéaire relativement à chaque argument (l'autre étant fixé) ;
- *symétrique* : $\forall (x, y) \in E^2 \quad (y \mid x) = (x \mid y)$;
- *positive* : $\forall x \in E \quad (x \mid x) \geqslant 0$;
- *définie* : $(x \mid x) = 0 \Rightarrow x = 0$.

Kernels ou noyaux classiques

- Polynomial d'ordre 2

$$\mathbf{x} \rightarrow [x_1, \ldots, x_1^2, \ldots, x_1 x_2, \ldots]$$

- Polynomial

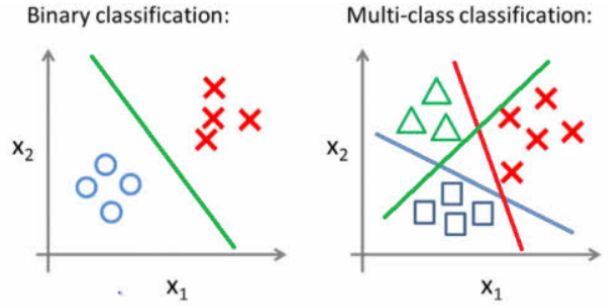
$$k(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x} \cdot \mathbf{x}_i)^n$$

- Gaussien

$$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2})$$

$\Rightarrow$ Reflexion sur la dimension des espaces induits

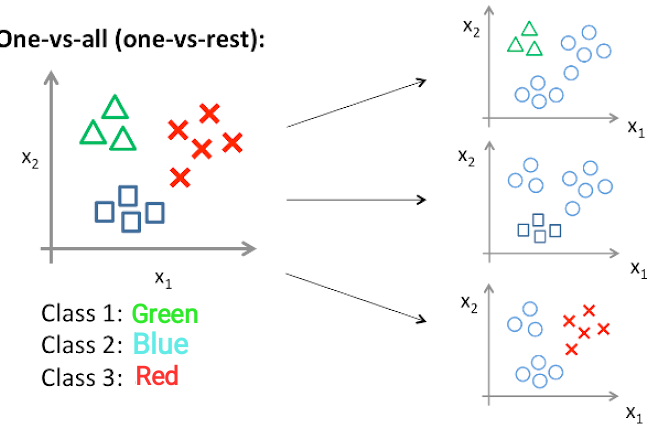
La plupart des problèmes de la vie réelle sont multi-classes :



Comment les traiter avec un perceptron qui attend des étiquettes dans $\{-1, 1\}$?

⇒ En multipliant les classifieurs !

One-vs-all (one-vs-rest):



On obtient autant de $\{\mathbf{w}_c\}$ que de classes... Quid de l'inférence ?

Evaluation

- Pas facile...
- Apprentissage / test
- Validation croisée

Conclusion

Vous savez tout

1. Récupérer des données
2. Les décrire
3. Les classer / scorer / catégoriser

... Mais que va-t-on faire dans le reste de l'UE ?