

# Optimisation et réseaux de neurones

23 février 2024



---

---

# Table des matières

<b>1 Fonctions de plusieurs variables et optimisation</b>	<b>5</b>
I Définition et exemples . . . . .	5
1 Définition . . . . .	5
2 Deux variables . . . . .	6
3 Trois variables . . . . .	6
4 $n$ variables . . . . .	7
II Graphe . . . . .	7
1 Définition . . . . .	7
2 Tranches . . . . .	9
3 Minimum, maximum . . . . .	11
4 Recherche élémentaire d'un minimum . . . . .	14
III Lignes de niveau . . . . .	16
1 Définition . . . . .	16
2 Exemples . . . . .	18
3 Surfaces quadratiques . . . . .	19
4 Régression linéaire . . . . .	24
IV Dérivées partielles . . . . .	26
1 Définition . . . . .	26
2 Calculs . . . . .	26
3 Interprétation géométrique . . . . .	27
V Gradient . . . . .	28
1 Définition . . . . .	28
2 Tangentes aux lignes de niveau . . . . .	29
3 Lignes de plus forte pente . . . . .	31
4 Dérivée directionnelle . . . . .	31
5 Surface de niveau . . . . .	33
6 Calcul approché . . . . .	34
7 Minimum et maximum . . . . .	35
VI Descente de gradient classique . . . . .	40
1 Où est le minimum ? . . . . .	40
2 Exemple en deux variables . . . . .	42
3 Exemples en une variable . . . . .	43
4 Algorithme du gradient . . . . .	47
VII Optimisation . . . . .	48
1 Faire varier le pas . . . . .	48
2 Régression linéaire $y = ax + b$ . . . . .	49
VIII Descente de gradient stochastique . . . . .	52
1 Petits pas à petits pas . . . . .	53
2 Différentes fonctions d'erreurs . . . . .	57
3 Descente par lots . . . . .	57
IX Accélérations . . . . .	61

1	Moment . . . . .	61
2	Nesterov . . . . .	62
3	Vocabulaire . . . . .	62
<b>2</b>	<b>Réseaux de neurones</b>	<b>63</b>
I	Perceptron . . . . .	63
1	Perceptron linéaire . . . . .	63
2	Biais – Perceptron affine . . . . .	67
II	Théorie du perceptron . . . . .	72
1	OU, ET, OU exclusif . . . . .	72
2	Séparation linéaire . . . . .	75
3	Vocabulaire . . . . .	76
III	Réseau de neurones . . . . .	77
1	Couches de neurones . . . . .	77
2	Exemples . . . . .	80
IV	Théorie des réseaux de neurones . . . . .	84
1	OU exclusif . . . . .	84
2	Ensemble réalisable . . . . .	85
3	Approximation d'ensembles . . . . .	88
V	Théorème d'approximation universelle . . . . .	88
1	Fonctions marches . . . . .	89
2	Fonctions créneaux . . . . .	90
3	Fonctions en escalier . . . . .	92
4	Théorème d'approximation universelle (une variable) . . . . .	93
5	Théorème d'approximation universelle (deux variables et plus) . . . . .	94
VI	Principe de la rétropropagation . . . . .	94
1	Objectif du réseau . . . . .	95
2	Descente de gradient . . . . .	97
3	Prédiction . . . . .	99
VII	Exemples . . . . .	100
1	Le « ou exclusif » . . . . .	100
2	Le perceptron et la règle de Hebb . . . . .	102
3	Une fonction marche . . . . .	103
4	Plus de neurones . . . . .	105
5	Apprentissage . . . . .	108
VIII	Sur-apprentissage et autres soucis . . . . .	110
1	Modèle insuffisant . . . . .	110
2	Minimum local . . . . .	111
3	Sous-apprentissage . . . . .	112
4	Sur-apprentissage . . . . .	113
<b>A</b>	<b>Compétences attendues à l'issue de ce cours</b>	<b>115</b>

# Fonctions de plusieurs variables et optimisation

De nombreux phénomènes dépendent de plusieurs paramètres, par exemple le volume d'un gaz dépend de la température et de la pression ; l'altitude  $z$  d'un terrain dépend des coordonnées  $(x, y)$  du lieu.

Dans les réseaux de neurones, les fonctions de plusieurs variables interviennent de deux manières :

- lors de l'utilisation d'un réseau. C'est la partie la plus facile et la plus fréquente. On utilise un réseau déjà bien paramétré pour répondre à une question (Est-ce une photo de chat ? Tourner à droite ou à gauche ? Quel pion déplacer au prochain coup ?). La réponse est un calcul direct obtenu en évaluant une fonction de plusieurs variables.
- lors de la paramétrisation du réseau. C'est la partie difficile et le but de ce cours. Quels paramètres choisir pour définir ce réseau afin qu'il réponde au problème ? Ces paramètres seront choisis comme minimum d'une fonction de plusieurs variables. Une des difficultés est qu'il peut y avoir des milliers de paramètres à gérer.

## I- Définition et exemples

### 1. Définition

Nous allons étudier les fonctions de deux variables, mais aussi de trois variables et plus généralement de  $n$  variables. Ces fonctions sont donc de la forme

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

où  $n$  est un entier naturel supérieur ou égal à 1.

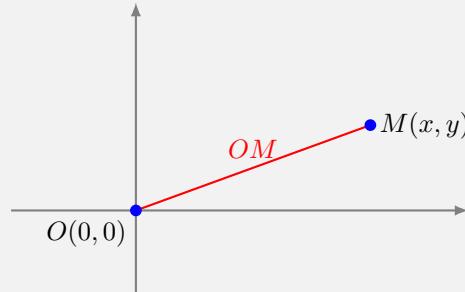
Un élément de l'ensemble de départ est un vecteur de type  $x = (x_1, \dots, x_n)$ . À chacun de ces vecteurs,  $f$  associe un nombre réel  $f(x_1, \dots, x_n)$ . On pourrait aussi limiter l'ensemble de départ à une partie  $E$  de  $\mathbb{R}^n$ .

## 2. Deux variables

Lorsque  $n = 2$ , on préfère noter les variables  $(x, y)$  plutôt que  $(x_1, x_2)$ . Voici quelques exemples.

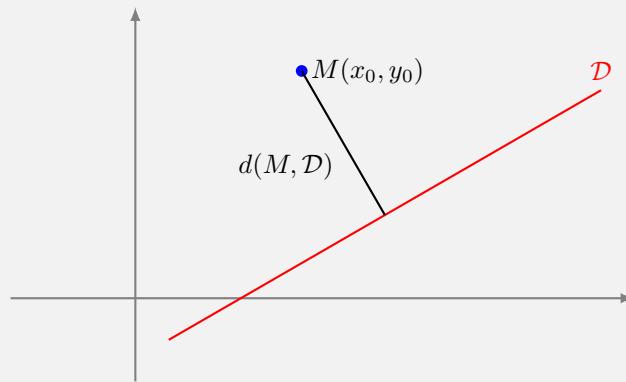
### Exemple 2.1

- $f(x, y) = 2x + 3y^2 + 1$ .
- $f(x, y) = \cos(xy)$ .
- $f(x, y) = \sqrt{x^2 + y^2}$ .  $f$  renvoie la distance entre un point  $M(x, y)$  et l'origine  $O(0, 0)$ .



- L'équation physique  $PV = nRT$  implique  $T = \frac{1}{nR}PV$  : la température d'un gaz s'exprime en fonction de son volume et de la pression ( $n$  et  $R$  sont des constantes).
- La distance entre une droite fixée  $\mathcal{D}$  d'équation  $ax + by + c = 0$  et un point  $M(x_0, y_0)$  est donnée par une fonction de deux variables :

$$d(x_0, y_0) = d(M, \mathcal{D}) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}.$$

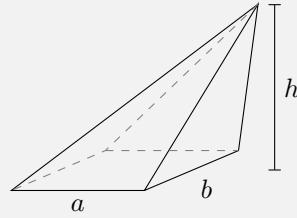


## 3. Trois variables

### Exemple 3.1

1.  $f(x, y, z) = ax + by + cz + d$ . Cette fonction  $f$  est une fonction affine ( $a, b, c, d$  sont des constantes).
2.  $f(x, y, z) = x^2 + y^2 + z^2$  qui donne la distance au carré entre un point  $M(x, y, z)$  et l'origine  $O(0, 0, 0)$ .
3. Le volume d'un cône à base rectangulaire dépend des longueurs des côtés  $a$  et  $b$  de la base et de la hauteur  $h$  :

$$V = f(a, b, h) = \frac{1}{3}abh.$$



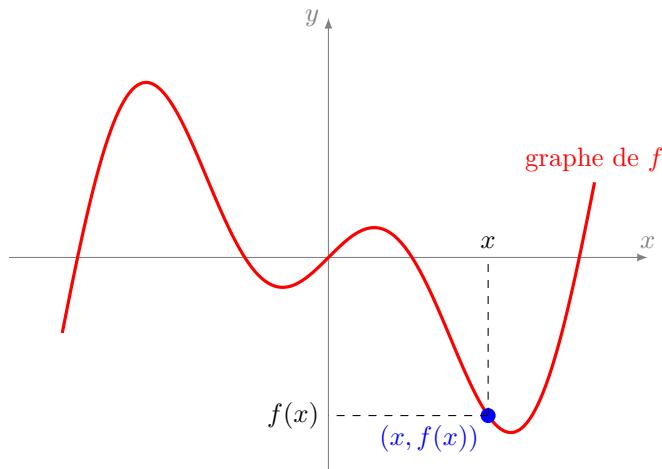
#### 4. $n$ variables

##### Exemple 4.1

1.  $f(x_1, \dots, x_n) = a_1x_1 + \dots + a_nx_n + a_0$  une fonction affine (les  $a_i$  sont des constantes).
2.  $f(x_1, \dots, x_n) = \sqrt{(x_1 - a_1)^2 + \dots + (x_n - a_n)^2}$  exprime la distance entre les points  $M(x_1, \dots, x_n)$  et  $A(a_1, \dots, a_n)$  dans  $\mathbb{R}^n$ .

## II- Graphe

Le cas le plus simple, et déjà connu, est celui des fonctions d'une seule variable  $f : \mathbb{R} \rightarrow \mathbb{R}$ . C'est l'ensemble de tous les points du plan de la forme  $(x, f(x))$ . Voici le graphe de la fonction  $x \mapsto x \cos x$ .



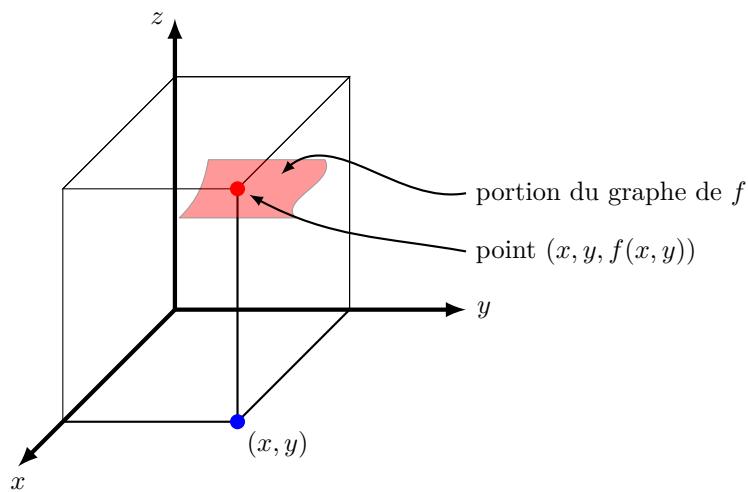
#### 1. Définition

##### Définition II.1

Le **graphe**  $\mathcal{G}_f$  d'une fonction de deux variables  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  est l'ensemble des points de  $\mathbb{R}^3$  ayant pour coordonnées  $(x, y, f(x, y))$ , pour  $(x, y)$  parcourant  $\mathbb{R}^2$ . Le graphe est donc :

$$\mathcal{G}_f = \{(x, y, z) \in \mathbb{R}^3 \mid (x, y) \in \mathbb{R}^2 \text{ et } z = f(x, y)\}.$$

Dans le cas de deux variables, le graphe d'une fonction est une surface tracée dans l'espace.



### Exemple 1.1

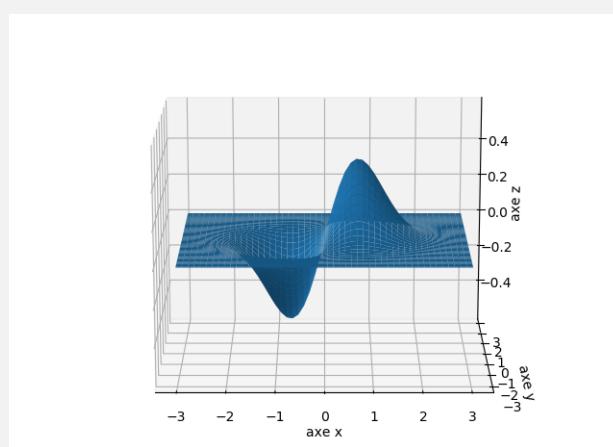
On souhaite tracer le graphe de la fonction définie par :

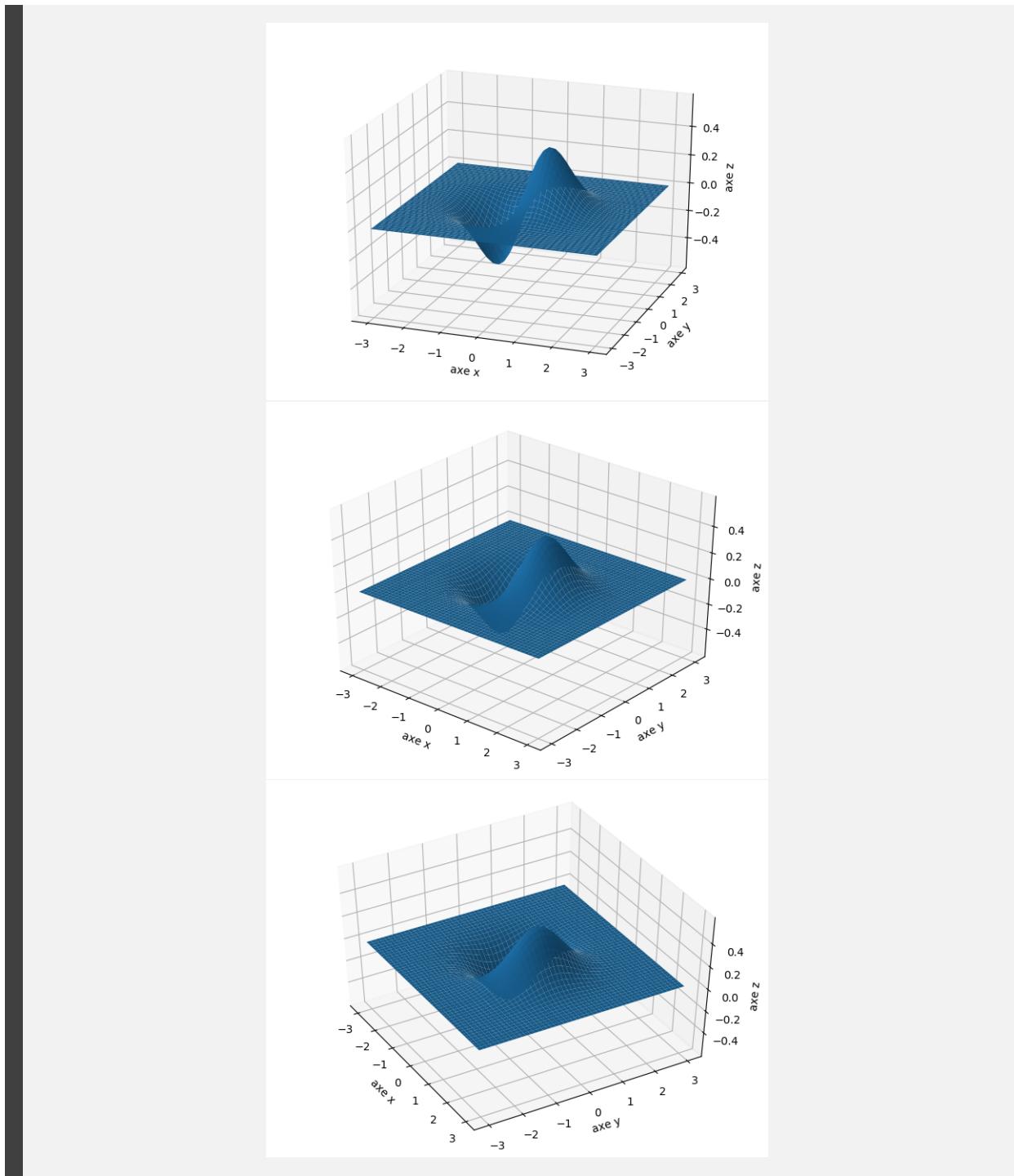
$$f(x, y) = xe^{-x^2-y^2}.$$

On commence par tracer quelques points à la main :

- si  $(x, y) = (0, 0)$  alors  $f(x, y) = f(0, 0) = 0$  donc le point de coordonnées  $(0, 0, 0)$  appartient au graphe.
- Comme  $f(1, 0) = 1/e$  alors le point de coordonnées  $(1, 0, 1/e)$  appartient au graphe.
- Pour n'importe quel  $y$ , on a  $f(0, y) = 0$  donc la droite de l'espace d'équation  $(x = 0 \text{ et } z = 0)$  est incluse dans le graphe.
- Notons  $r = \sqrt{x^2 + y^2}$  la distance entre le point de coordonnées  $(x, y)$  et l'origine  $(0, 0)$  alors on a la formule  $f(x, y) = xe^{-r^2}$ . Pour un point éloigné de l'origine,  $r$  est grand, donc  $e^{-r^2}$  est très petit, et  $f(x, y)$  est très proche de 0.

Voici différentes vues de ce graphe.





## 2. Tranches

Afin de tracer le graphe d'une fonction de deux variables, on peut découper la surface en « tranches ». On fixe par exemple une valeur  $y_0$  et on trace dans le plan ( $xOz$ ) le graphe de la fonction d'une variable

$$f|_{y_0} : x \mapsto f(x, y_0).$$

Géométriquement, cela revient à tracer l'intersection du graphe de  $f$  et du plan d'équation ( $y = y_0$ ). On recommence pour plusieurs valeurs de  $y_0$ , ce qui nous donne des tranches du graphe de  $f$  et nous donne une bonne idée du graphe complet de  $f$ .

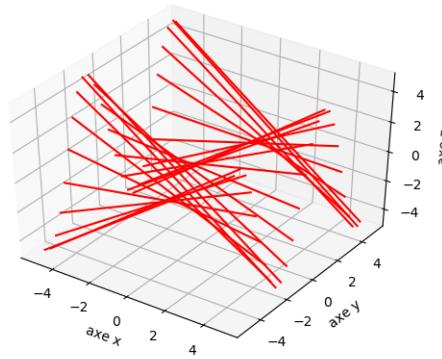
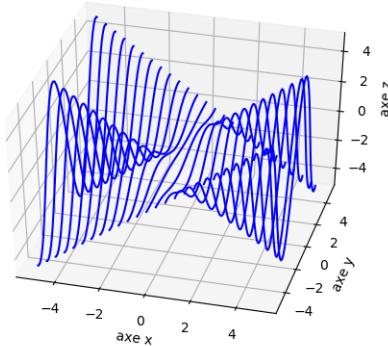
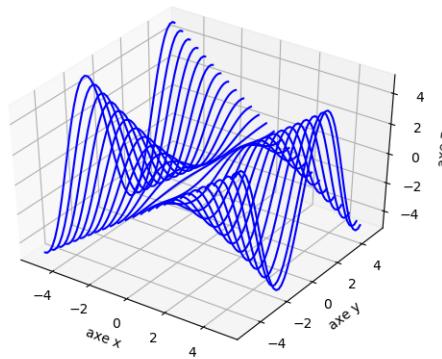
On peut faire le même travail en fixant des valeurs  $x_0$  avec les fonctions :

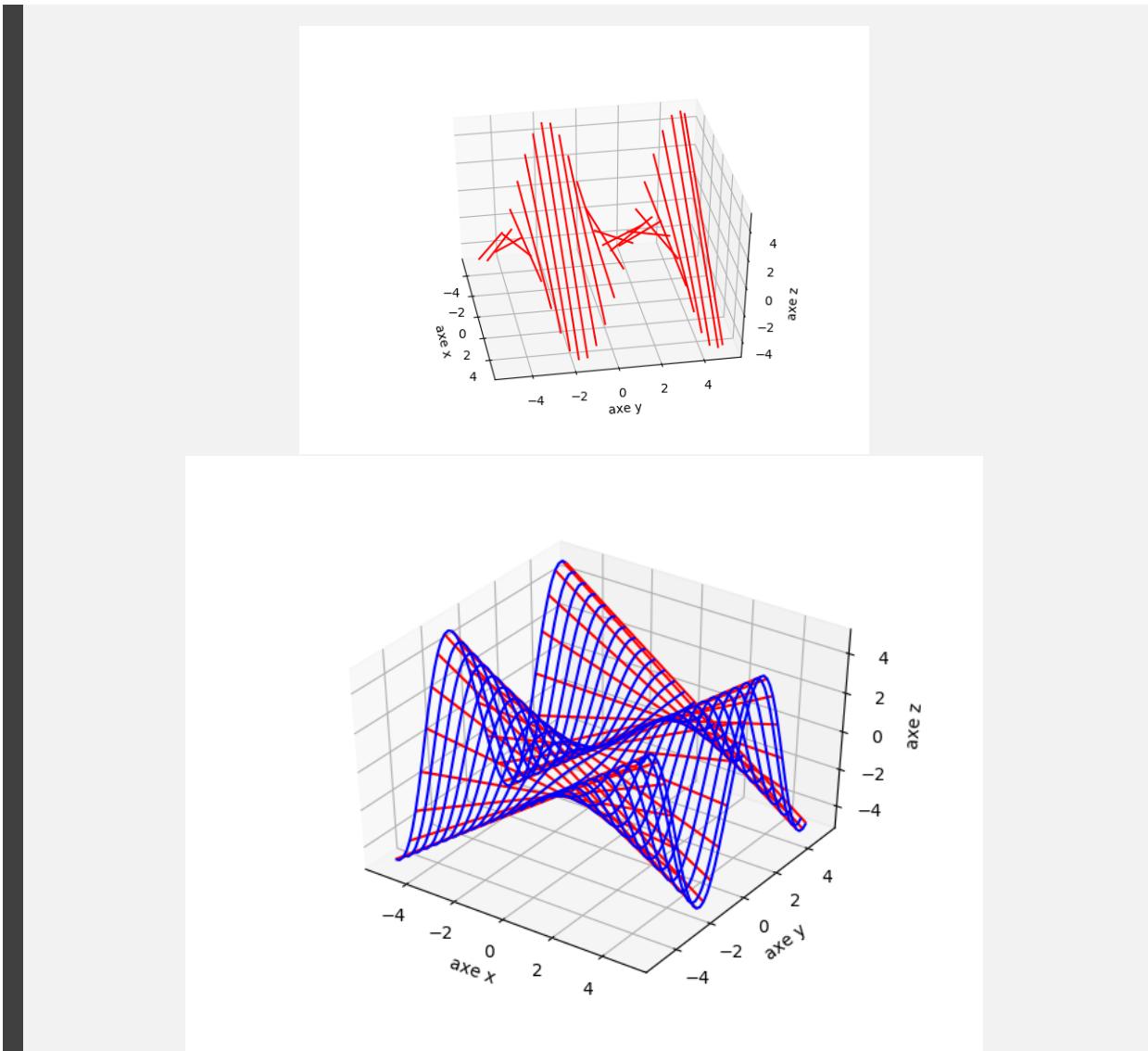
$$f|_{x_0} : y \mapsto f(x_0, y).$$

**Exemple 2.1**

On souhaite tracer le graphe de la fonction définie par :

$$f(x, y) = x \sin(y).$$





En haut les tranches pour lesquelles  $x$  est constant (deux points de vue), au milieu les tranches pour lesquelles  $y$  est constant (deux points de vue). Lorsque l'on rassemble les tranches (à  $x$  constant et à  $y$  constant), on reconstitue la surface (dernière figure).

### 3. Minimum, maximum

Pour des fonctions de deux variables (ou plus) il existe une notion de minimum et de maximum.

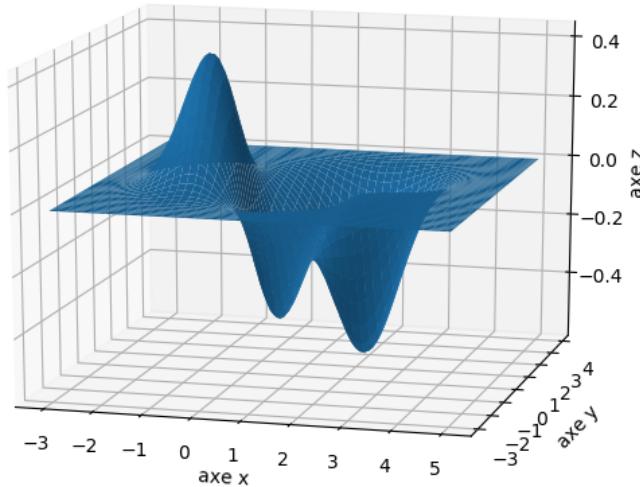
#### Définition II.2

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  une fonction.

- $f$  atteint un **minimum global** en  $(x_0, y_0) \in \mathbb{R}^2$  si pour tout  $(x, y) \in \mathbb{R}^2$ , on a  $f(x, y) \geq f(x_0, y_0)$ .
- $f$  atteint un **minimum local** en  $(x_0, y_0) \in \mathbb{R}^2$  si il existe un intervalle ouvert  $I$  contenant  $x_0$  et un intervalle ouvert  $J$  contenant  $y_0$  tels que pour tout  $(x, y) \in I \times J$ , on a  $f(x, y) \geq f(x_0, y_0)$ .

### Exemple 3.1

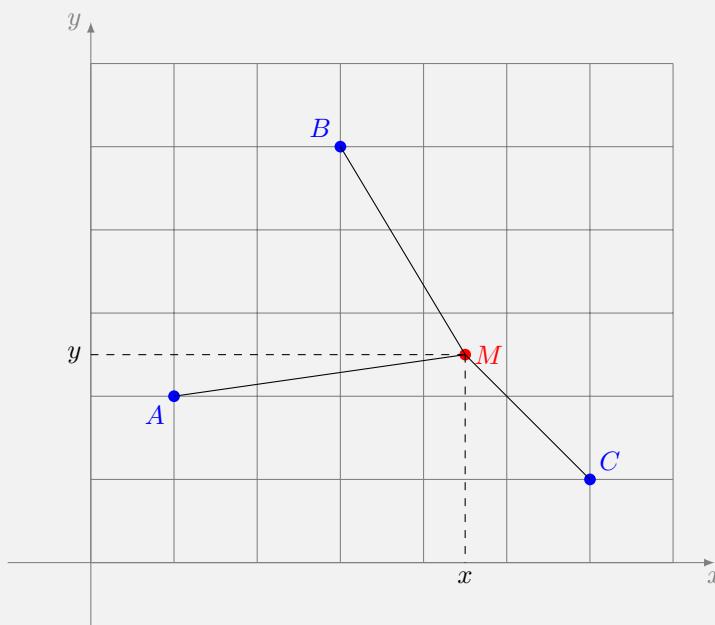
Voici l'exemple d'une fonction qui admet deux minimums locaux. L'un est aussi un minimum global. Elle admet un maximum local qui est aussi global.



Trouver les bons paramètres d'un réseau de neurones nous amènera à trouver le minimum d'une fonction de plusieurs (et même de centaines de) variables. Voyons un exemple en deux variables.

### Exemple 3.2

Étant donnés trois points  $A(1, 2)$ ,  $B(3, 5)$  et  $C(6, 1)$ , il s'agit de trouver un point  $M(x, y)$  qui « approche au mieux » ces trois points. Il faut expliciter une fonction à minimiser pour définir correctement le problème. Nous décidons de prendre la somme des carrés des distances.



Il s'agit donc de minimiser la fonction  $f$  suivante, qui correspond à une fonction distance

(aussi appelée fonction erreur ou bien fonction coût) :

$$f(x, y) = MA^2 + MB^2 + MC^2 = (x - 1)^2 + (y - 2)^2 + (x - 3)^2 + (y - 5)^2 + (x - 6)^2 + (y - 1)^2.$$

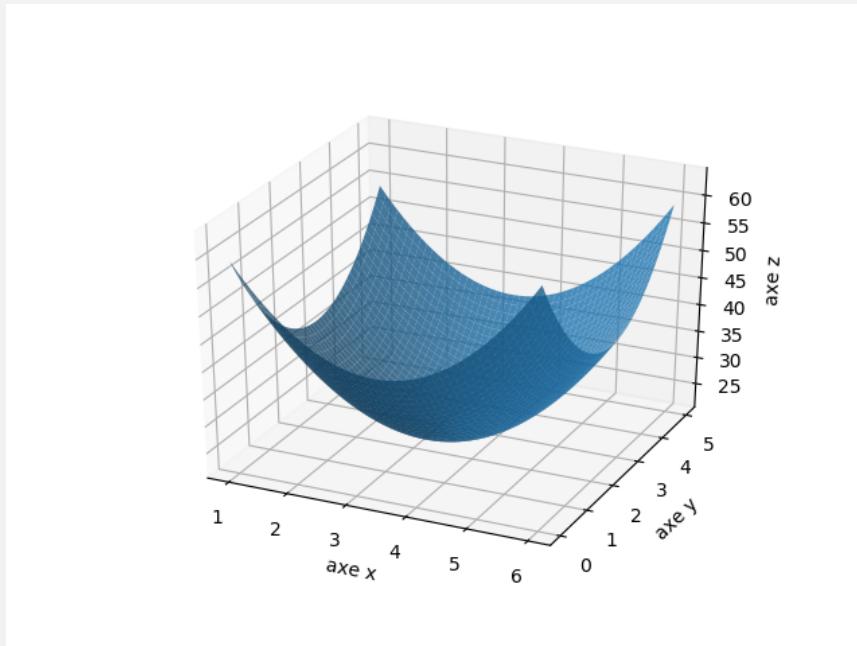
En développant on trouve :

$$f(x, y) = 3x^2 + 3y^2 - 20x - 16y + 76.$$

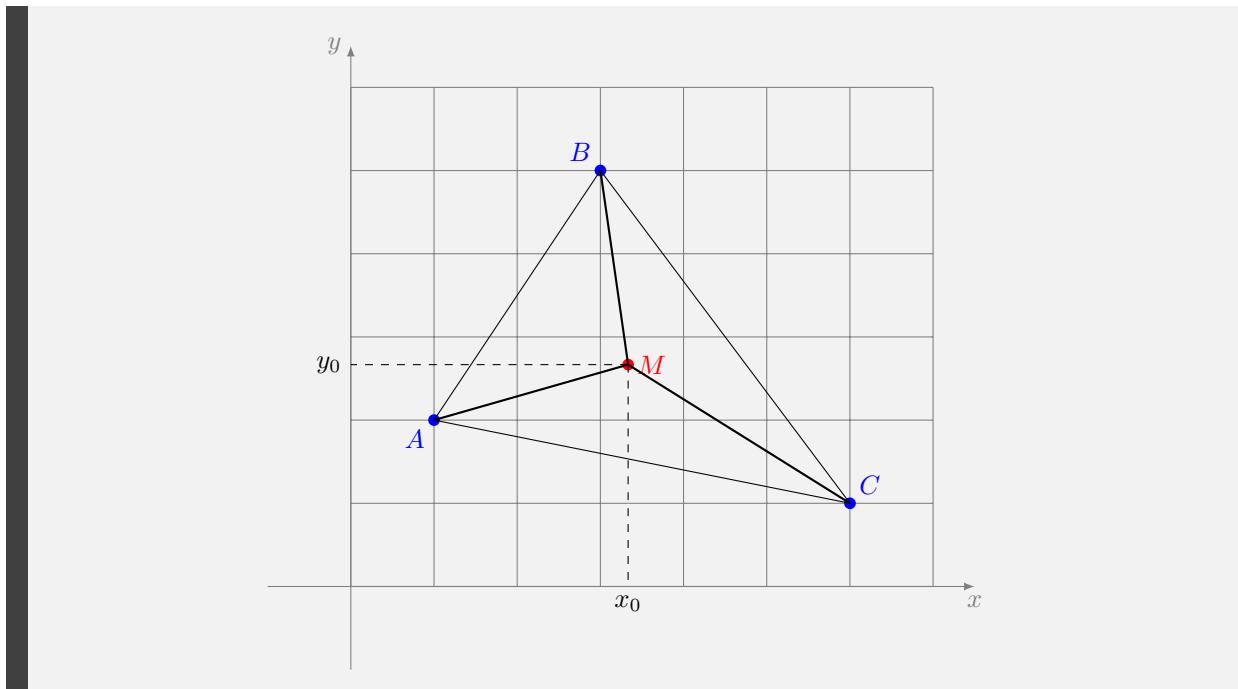
Le graphe de  $f$  nous suggère qu'il existe un unique minimum qui est le minimum global de  $f$ . Par recherche graphique ou par les méthodes décrites dans la section suivante, on trouverait une solution approchée. En fait la solution géométrique exacte est l'isobarycentre des points (autrement dit le centre de gravité du triangle  $ABC$ ), ainsi :

$$(x_0, y_0) = \left( \frac{10}{3}, \frac{8}{3} \right) \simeq (3.33, 2.66)$$

pour lequel  $f$  atteint son minimum  $z_0 = f(x_0, y_0) = \frac{64}{3} \simeq 21.33$ .



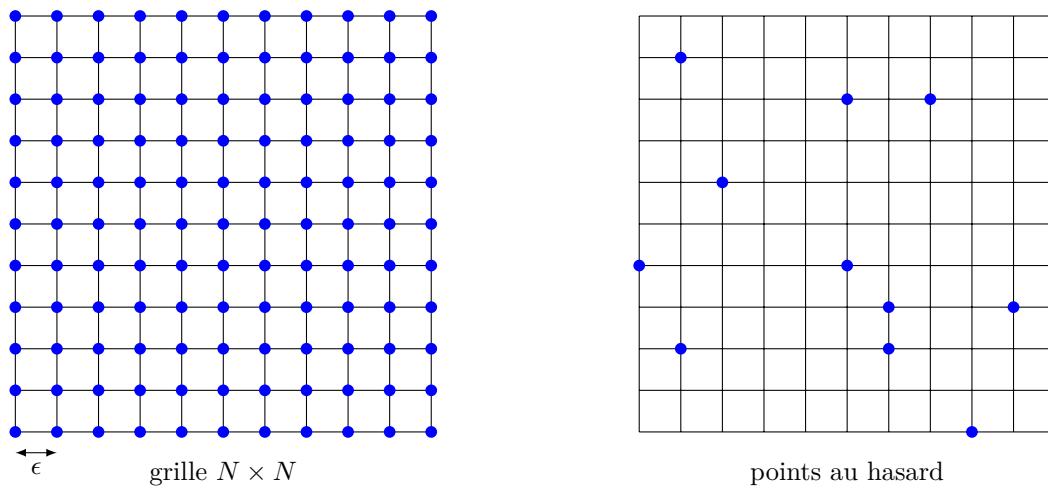
Le point qui convient le mieux à notre problème et en lequel notre fonction distance est minimale est donc le point de coordonnées  $(\frac{10}{3}, \frac{8}{3})$ . Attention, un autre choix de la fonction distance  $f$  pourrait conduire à une autre solution (voir l'exemple de la prochaine section).



#### 4. Recherche élémentaire d'un minimum

Voici trois techniques pour trouver les valeurs approchées des coordonnées du point en lequel une fonction de plusieurs variables atteint son minimum. Ces techniques sont valables quelque soit le nombre de variables même si ici elles ne sont décrites que dans le cas de deux variables.

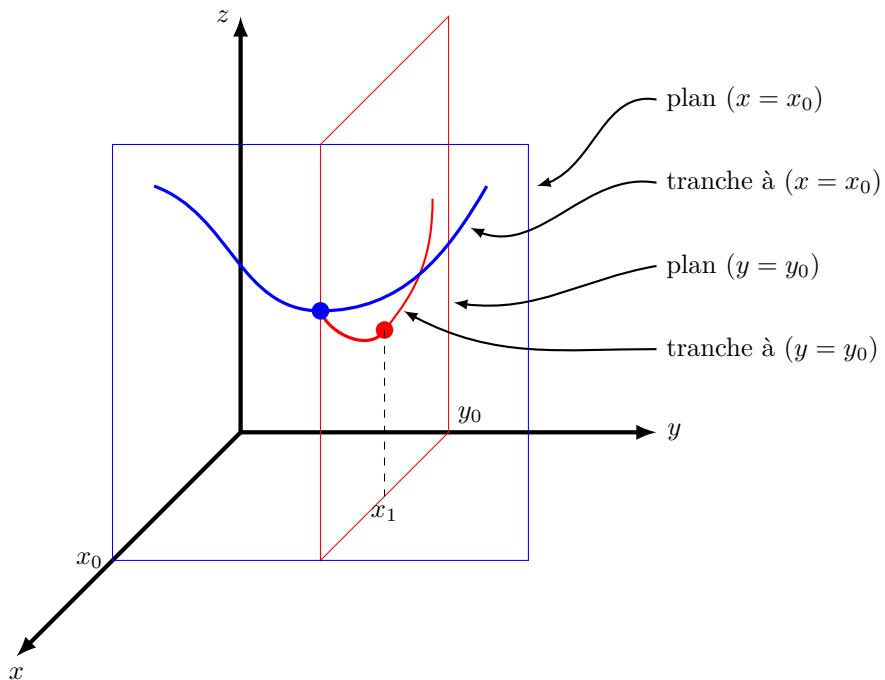
**Recherche sur une grille.** On calcule  $f(x, y)$  pour  $(x, y)$  parcourant une grille. On retient le point  $(x_0, y_0)$  en lequel  $z_0 = f(x_0, y_0)$  est le plus petit. Si on le souhaite, on peut affiner la grille autour de ce point, en diminuant le pas  $\epsilon$  pour améliorer l'approximation. C'est une technique qui demande  $N^2$  calculs pour une grille de largeur  $N$  (et même  $N^n$  pour une fonction de  $n$  variables) ce qui peut être énorme.



**Recherche au hasard.** Cela peut sembler incongru mais choisir quelques coordonnées  $(x, y)$  au hasard, calculer chaque valeur  $z = f(x, y)$  et comme auparavant retenir le point  $(x_0, y_0)$  correspondant au  $z_0$  minimal n'est pas ridicule! Un ordinateur peut tester plusieurs millions de points en quelques secondes. Bien sûr il y a de fortes chances de ne trouver qu'une solution approchée. C'est aussi une technique que l'on retrouvera plus tard : partir d'un point au hasard pour ensuite construire une suite de points convergeant vers un minimum. Et si cela n'est pas concluant, il faudra repartir d'un autre point tiré au hasard.

**Recherche par tranche.** L'idée est de se ramener à des fonctions d'une seule variable. En effet, pour les fonctions d'une variable, on sait qu'il faut chercher les minimums là où la dérivée s'annule.

On part d'une valeur  $x_0$  (au hasard!). On cherche le minimum sur la tranche  $x = x_0$ , c'est-à-dire que l'on cherche le minimum de la fonction d'une variable  $y \mapsto f(x_0, y)$ . On trouve une valeur  $y_0$  qui réalise un minimum. On change alors de direction en étudiant maintenant la tranche  $y = y_0$  (pour le  $y_0$  que l'on vient d'obtenir), on obtient l'abscisse  $x_1$  du minimum de la fonction d'une variable  $x \mapsto f(x, y_0)$ . On recommence depuis le début à partir de ce  $x_1$ . On obtient ainsi une suite de points  $(x_i, y_i)$  avec des valeurs  $z_i = f(x_i, y_i)$  de plus en plus petites. On peut espérer tendre vers un minimum.



Sur la figure ci-dessus, on part d'une tranche choisie au hasard, donnée par  $x = x_0$ . Cette tranche définit le graphe d'une fonction d'une variable. On se déplace sur cette courbe jusqu'à atteindre le minimum de cette tranche, en une valeur  $y_0$ . On considère la tranche perpendiculaire donnée par  $y = y_0$ . On se déplace sur la courbe jusqu'à atteindre le minimum de cette tranche, en une valeur  $x_1$ . On pourrait continuer avec une nouvelle tranche  $x = x_1$ , etc.

Les descriptions données ici sont assez informelles, d'une part parce qu'il est difficile d'énoncer des théorèmes qui garantissent d'atteindre un minimum local et d'autre part parce qu'aucune technique ne garantit d'atteindre un minimum global. Lorsque l'on étudiera le gradient, nous obtiendrons une méthode plus efficace.

### Exemple 4.1

On reprend le problème précédent, à savoir trouver un point  $M$  qui approche au mieux les trois points  $A$ ,  $B$  et  $C$ , mais cette fois on choisit la « vraie » distance comme fonction d'erreur :

$$g(x, y) = MA + MB + MC = \sqrt{(x - 1)^2 + (y - 2)^2} + \sqrt{(x - 3)^2 + (y - 5)^2} + \sqrt{(x - 6)^2 + (y - 1)^2}.$$

On applique ces trois techniques pour chercher le minimum de  $g$  en se limitant au carré  $[0, 6] \times [0, 6]$ .

1. Avec une grille  $N \times N$ . Par exemple pour  $N = 100$ , on évalue  $g$  en 10 000 points. On trouve  $(x_{\min}, y_{\min}) \simeq (2.91, 2.97)$  pour une valeur  $z_{\min} \simeq 7.84$ .

2. Avec un tirage aléatoire de 1000 points, on trouve par exemple :  $(x_{\min}, y_{\min}) \simeq (2.88, 2.99)$  et  $z_{\min} \simeq 7.84$ . Le résultat est similaire à la méthode précédente, bien qu'on ait effectué 10 fois moins de calculs.

3. Par les tranches. On part de la tranche ( $x = 0$ ). On pose  $x_0 = 0$ , la fonction à étudier est donc

$$g|_{x_0}(y) = \sqrt{1 + (y - 2)^2} + \sqrt{9 + (y - 5)^2} + \sqrt{36 + (y - 1)^2}.$$

C'est une fonction de la seule variable  $y$  pour laquelle on possède des techniques efficaces de recherche de minimum. On trouve que le minimum de  $g|_{x_0}$  est atteint en  $y_0 \simeq 2.45$ . On recommence avec cette fois la tranche ( $y = y_0$ ) et on cherche le minimum de la fonction  $g|_{y_0}(x) = g(x, y_0)$ . On trouve que cette fonction atteint son minimum en  $x_1 \simeq 2.84$ . On recommence ce processus jusqu'à atteindre la précision souhaitée. Ainsi en 5 étapes on obtient une valeur approchée assez précise du minimum  $(x_{\min}, y_{\min}) \simeq (2.90579, 2.98464)$  et  $z_{\min} \simeq 7.83867$ .

La première conclusion à tirer de ce qui précède est que pour résoudre un problème il faut définir correctement une fonction d'erreur, c'est-à-dire celle que l'on cherche à minimiser. La solution trouvée dépend de cette fonction d'erreur choisie. Enfin, la méthode des tranches est une méthode efficace pour trouver un minimum d'une fonction, mais nous en découvrirons une encore meilleure en utilisant le gradient.

### III- Lignes de niveau

#### 1. Définition

##### Définition III.1

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  une fonction de deux variables. La **ligne de niveau**  $z = c \in \mathbb{R}$  est l'ensemble de tous les points  $(x, y)$  vérifiant  $f(x, y) = c$  :

$$L_c = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) = c\}.$$

La ligne de niveau  $c$  est une courbe du plan  $\mathbb{R}^2$ .

On peut aussi définir une **courbe de niveau**, c'est l'ensemble des points de l'espace obtenus comme intersection du graphe  $\mathcal{G}_f$  et du plan  $z = c$  qui est horizontal et « d'altitude »  $c$ . Ce sont donc tous les points  $(x, y, f(x, y))$  avec  $f(x, y) = c$ . On obtient la courbe de niveau en translatant la ligne de niveau d'une altitude  $c$ .

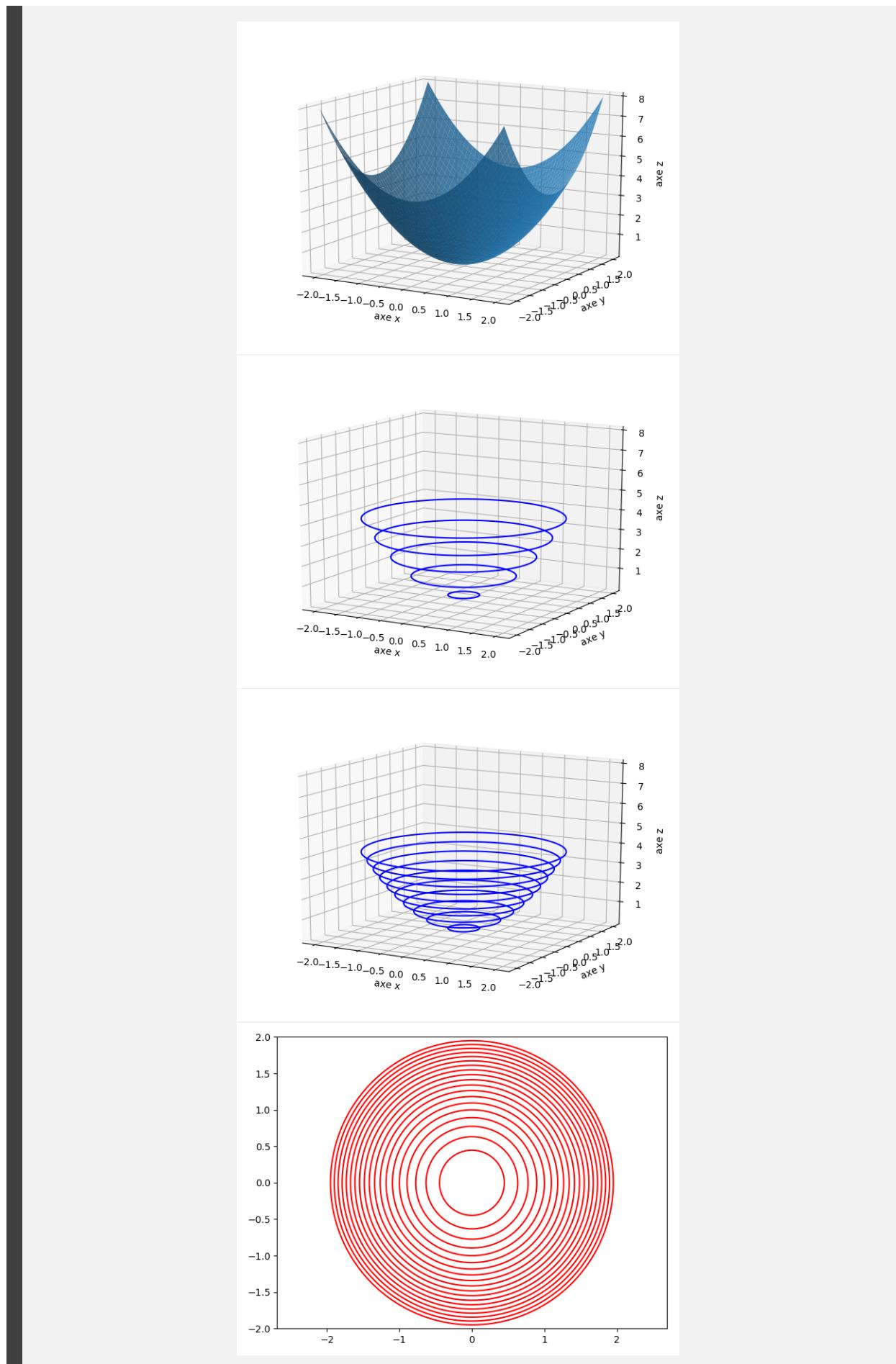
##### Exemple 1.1

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  définie par  $f(x, y) = x^2 + y^2$ .

- Si  $c < 0$ , la ligne de niveau  $L_c$  est vide (aucun point n'est d'altitude négative).
- Si  $c = 0$ , la ligne de niveau  $L_0$  se réduit à  $\{(0, 0)\}$ .
- Si  $c > 0$ , la ligne de niveau  $L_c$  est le cercle du plan de centre  $(0, 0)$  et de rayon  $\sqrt{c}$ . On « remonte »  $L_c$  à l'altitude  $z = c$  : la courbe de niveau est alors le cercle horizontal de l'espace de centre  $(0, 0, c)$  et de rayon  $\sqrt{c}$ .

Le graphe est alors une superposition de cercles horizontaux de l'espace de centre  $(0, 0, c)$  et de rayon  $\sqrt{c}$ ,  $c > 0$ .

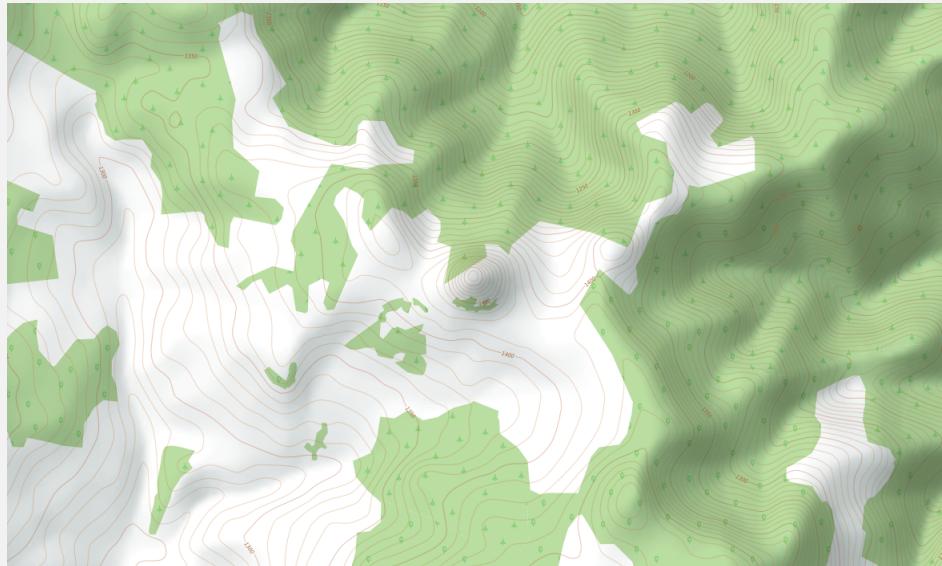
Ci-dessous : (a) la surface, (b) 5 courbes de niveau, (c) 10 courbes de niveau, (d) les lignes de niveau dans le plan.



## 2. Exemples

### Exemple 2.1

Sur une carte topographique, les lignes de niveau représentent les courbes ayant la même altitude.

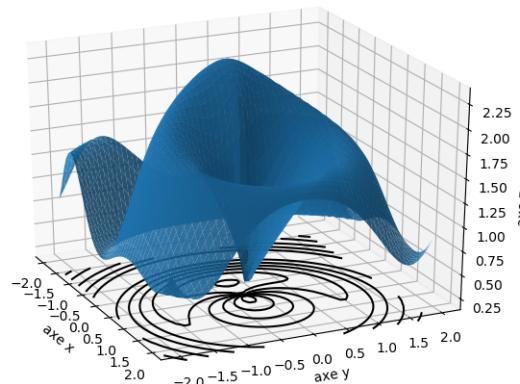


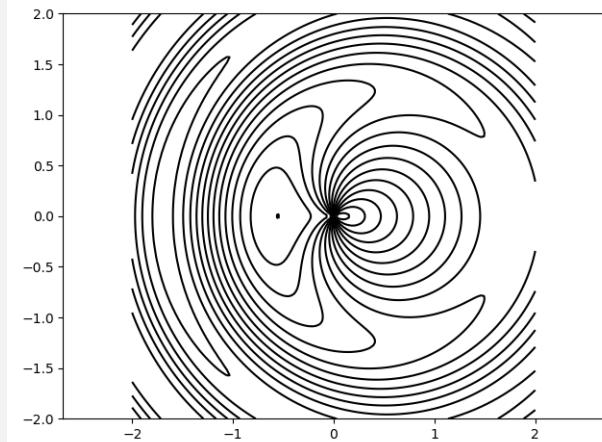
- Ici, une carte *Open Street Map* avec au centre le mont Gerbier de Jonc (source de la Loire, 1551 m).
- Les lignes de niveau correspondent à des altitudes équidistantes de 10 m (par exemple, pour  $c = 1400$ ,  $c = 1410$ ,  $c = 1420\ldots$ ).
- Lorsque les lignes de niveau sont très espacées, le terrain est plutôt plat ; lorsque les lignes sont rapprochées le terrain est pentu.
- Par définition, si on se promène en suivant une ligne de niveau, on reste toujours à la même altitude !

### Exemple 2.2

Voici le graphe et les lignes de niveau de la fonction  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  définie par

$$f(x, y) = \frac{\sin(r^2 - x)}{r} + 1 \quad \text{où } r = \sqrt{x^2 + y^2}.$$





### 3. Surfaces quadratiques

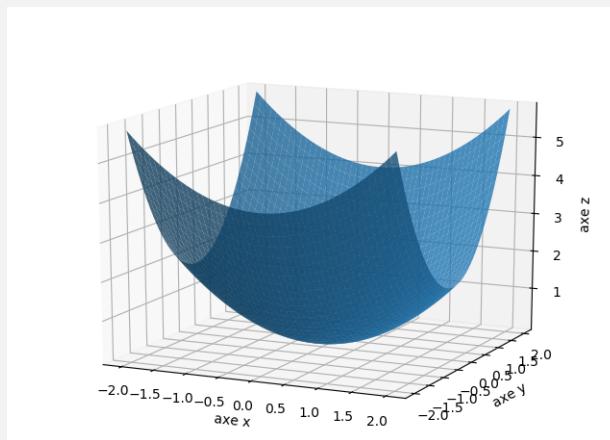
Ce sont des exemples à bien comprendre car ils seront importants pour la suite du cours.

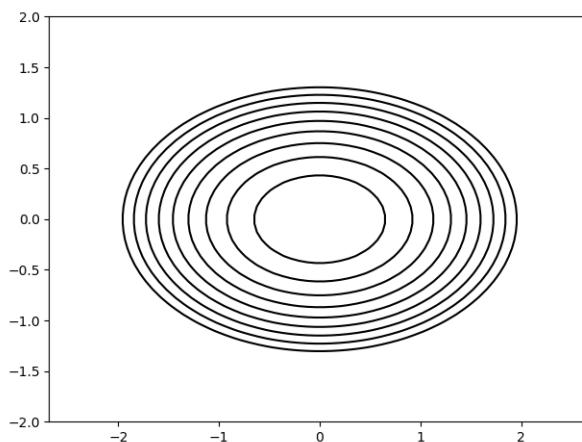
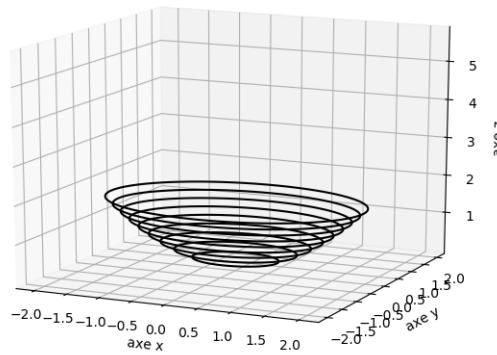
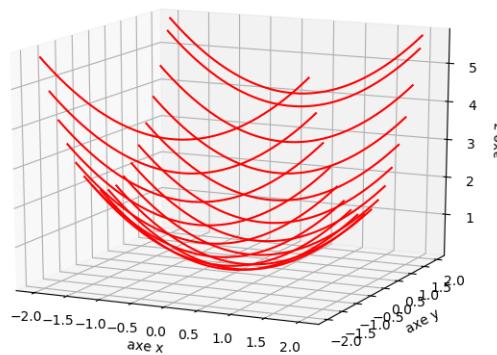
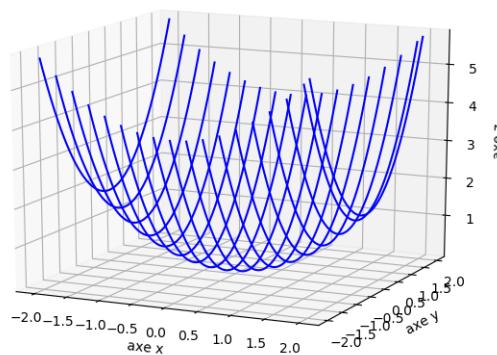
#### Exemple 3.1

$$f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$$

- Les tranches sont des paraboles.
- Les lignes de niveau sont des ellipses.
- Le graphe est donc un **paraboloïde elliptique**.

Ci-dessous : (a) la surface, (b) les tranches avec  $x$  constant, (c) les tranches avec  $y$  constant, (d) les courbes de niveau, (e) les lignes de niveau dans le plan.



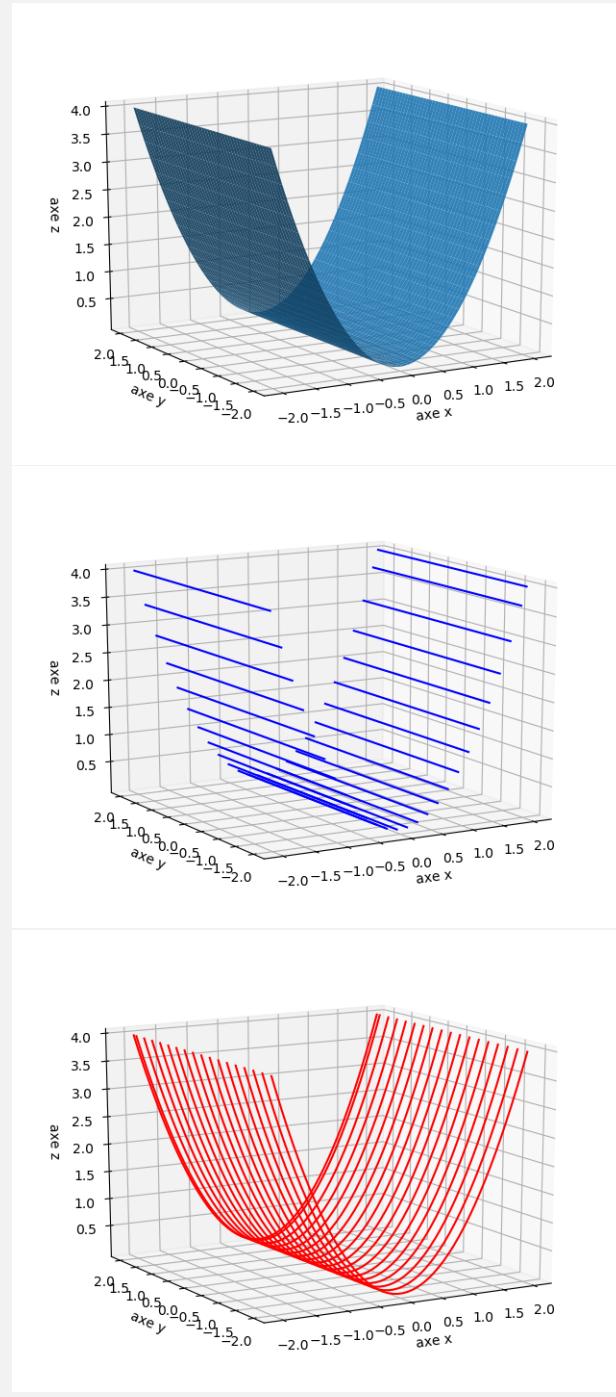


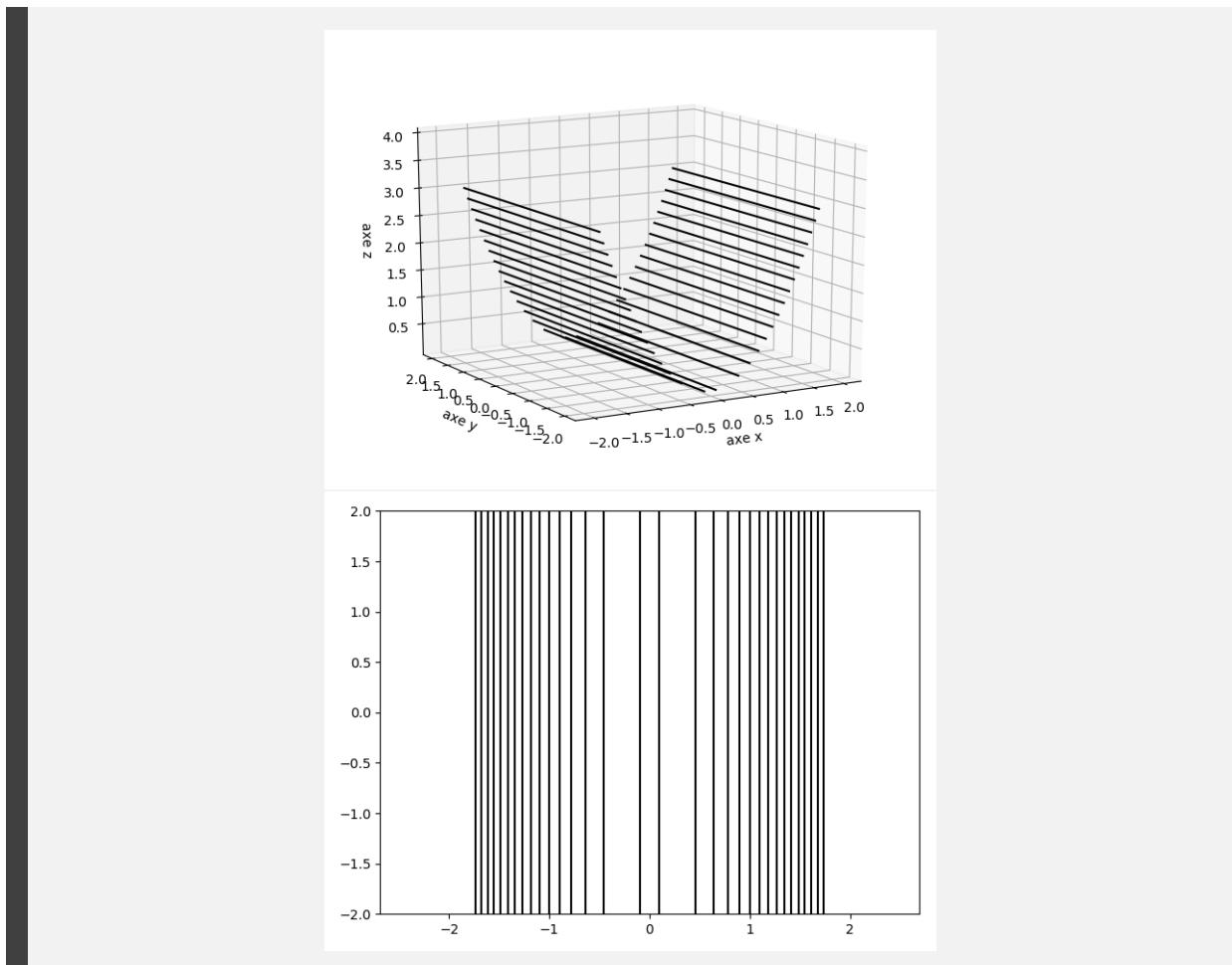
**Exemple 3.2**

$$f(x, y) = x^2$$

- Les tranches obtenues en coupant selon des plans  $y = y_0$  sont des paraboles. Dans l'autre direction, ce sont des droites.
- Les lignes de niveau sont des droites.
- Le graphe est donc un **cylindre parabolique**.

Ci-dessous : (a) la surface, (b) les tranches avec  $x$  constant, (c) les tranches avec  $y$  constant, (d) les courbes de niveau, (e) les lignes de niveau dans le plan.



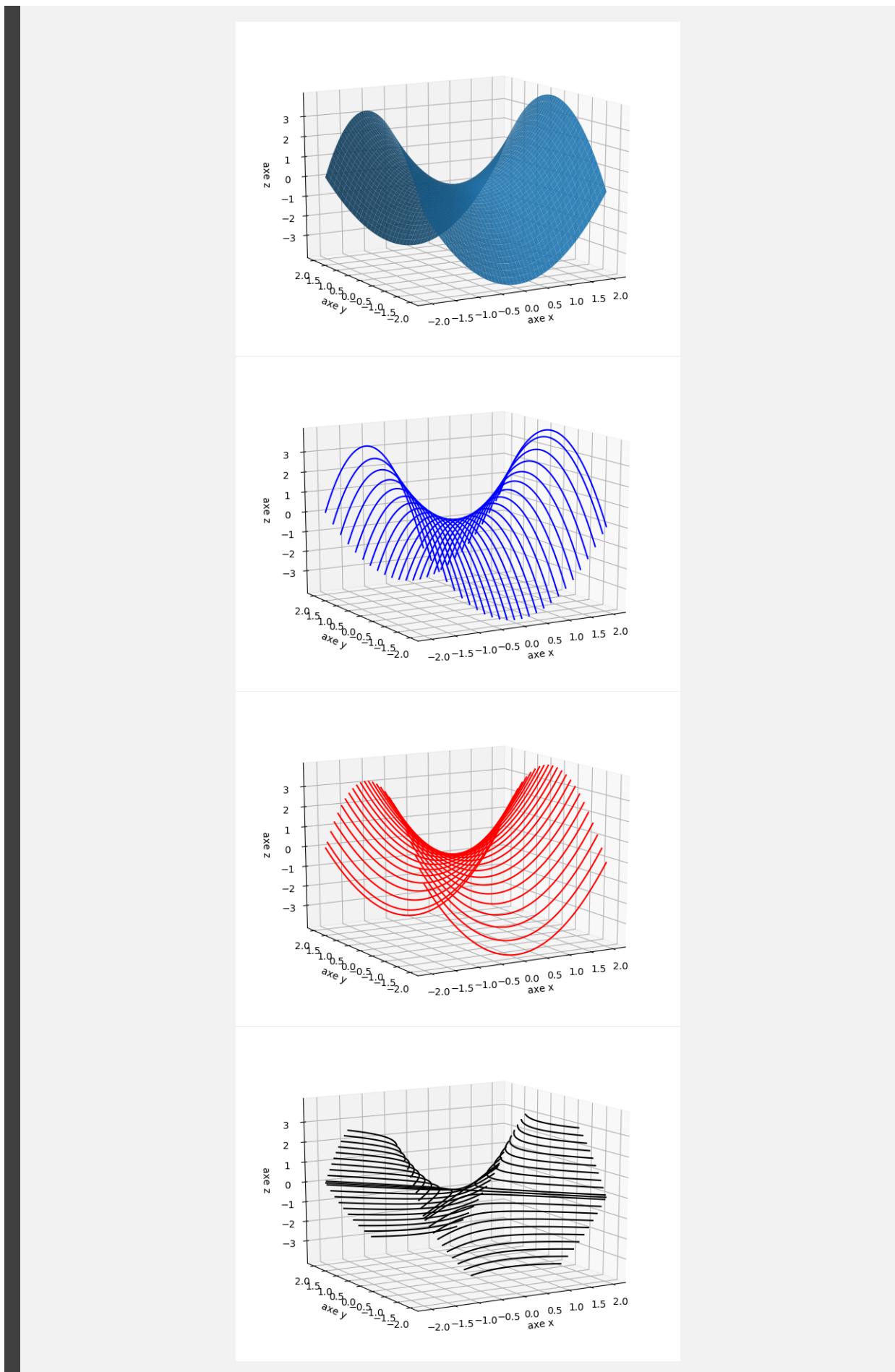


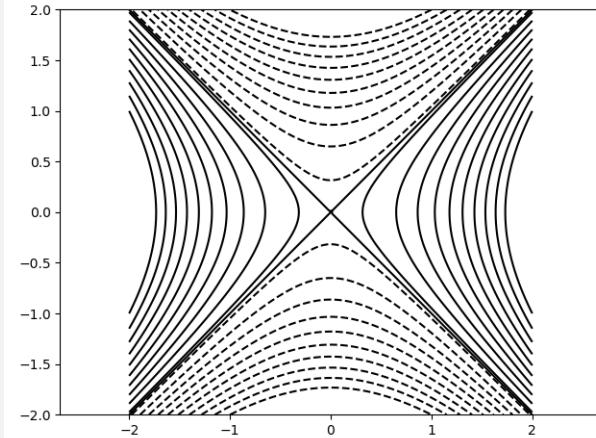
### Exemple 3.3

$$f(x, y) = x^2 - y^2$$

- Les tranches sont des paraboles, tournées vers le haut ou vers le bas selon la direction de la tranche.
- Les lignes de niveau sont des hyperboles.
- Le graphe est donc un **paraboloïde hyperbolique** que l'on appelle aussi la **selle de cheval**.
- Un autre nom pour cette surface est un **col** (en référence à un col en montagne). En effet le point  $(0, 0, 0)$ , est le point de passage le moins haut pour passer d'un versant à l'autre de la montagne.

Ci-dessous : (a) la surface, (b) les tranches avec  $x$  constant, (c) les tranches avec  $y$  constant, (d) les courbes de niveau, (e) les lignes de niveau dans le plan (en pointillé les lignes de niveau négatif).





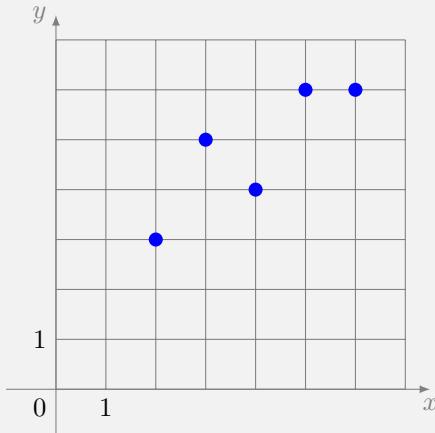
#### 4. Régression linéaire

##### Exemple 4.1

On se donne des points du plan, comme ci-dessous. On s'aperçoit qu'ils sont à peu près alignés et on souhaite trouver l'équation d'une droite :

$$y = ax + b$$

qui les approche au mieux.



Formalisons un peu le problème : on se donne  $N$  points  $A_i(x_i, y_i)$ ,  $i = 1, \dots, N$ . Pour une droite  $\mathcal{D}$  d'équation  $y = ax + b$ , la distance entre  $A_i$  et la droite  $\mathcal{D}$  est donnée par la formule :

$$d(A_i, \mathcal{D}) = \frac{|ax_i - y_i + b|}{\sqrt{1 + a^2}}.$$

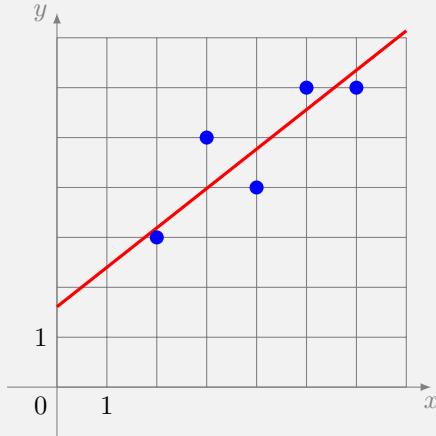
Pour se débarrasser des valeurs absolues et des racines carrées, on élève au carré et on décide que la droite qui approche au mieux tous les points  $A_i$  est la droite qui minimise la fonction

$$f(a, b) = \sum_{i=1}^N d(A_i, \mathcal{D})^2 = \frac{1}{1 + a^2} \sum_{i=1}^N (ax_i - y_i + b)^2.$$

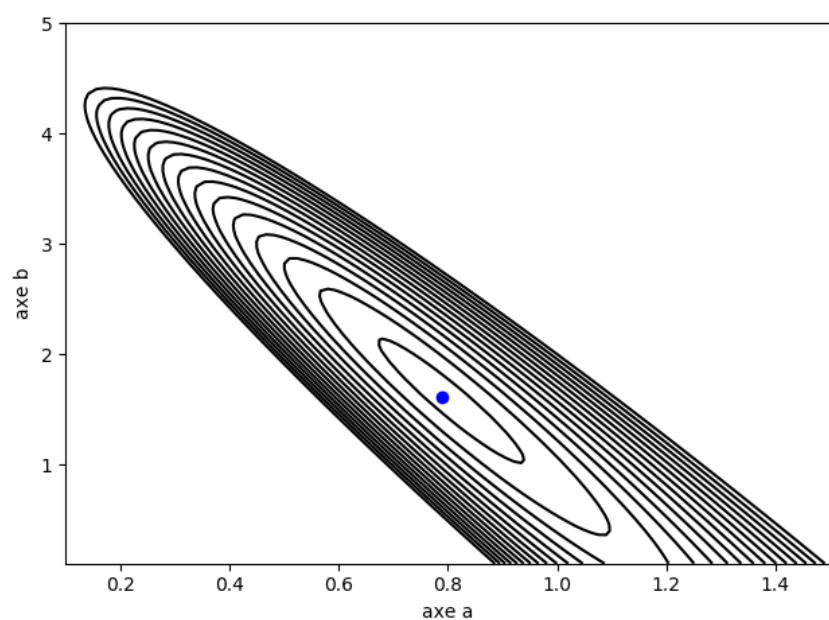
Pour les 5 points du dessin initial :  $A_1(2, 3)$ ,  $A_2(3, 5)$ ,  $A_3(4, 4)$ ,  $A_4(5, 6)$  et  $A_5(6, 6)$ , il s'agit donc de trouver  $(a, b)$  qui minimise la fonction

$$f(a, b) = \frac{1}{1 + a^2} ((2a - 3 + b)^2 + (3a - 5 + b)^2 + (4a - 4 + b)^2 + (5a - 6 + b)^2 + (6a - 6 + b)^2).$$

On trace le graphe de  $f$ , les lignes de niveau de  $f$ , et on utilise les techniques à notre disposition pour trouver qu'un minimum global est réalisé en  $(a, b) \simeq (0.8, 1.6)$ , ce qui permet de tracer une droite  $y = ax + b$  solution.



Lorsque l'on dessine les lignes de niveau, on s'aperçoit que le minimum se trouve dans une région plate et allongée. Cela signifie que, bien que le minimum  $(a_{\min}, b_{\min})$  soit unique, il existe beaucoup de  $(a, b)$  tels que  $f(a, b)$  soit proche de la valeur minimale  $f(a_{\min}, b_{\min})$ . De plus, ces points  $(a, b)$  peuvent être assez éloignés de la solution  $(a_{\min}, b_{\min})$  (par exemple tous les points de la zone ovale autour du minimum). Ce qui signifie que beaucoup de droites très différentes approchent la solution optimale.



## IV- Dérivées partielles

Pour une fonction de plusieurs variables, il existe une dérivée pour chacune des variables, qu'on appelle dérivée partielle.

### 1. Définition

#### Définition IV.1

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ . La **dérivée partielle**  $\frac{\partial f}{\partial x}(x_0, y_0)$  de  $f$  par rapport à la variable  $x$  au point  $(x_0, y_0) \in \mathbb{R}^2$  est la dérivée en  $x_0$  de la fonction d'une variable  $x \mapsto f(x, y_0)$ .

De même  $\frac{\partial f}{\partial y}(x_0, y_0)$  est la dérivée partielle de  $f$  par rapport à la variable  $y$  au point  $(x_0, y_0)$ .

Comme d'habitude et sauf mention contraire, nous supposerons que toutes les dérivées partielles existent.

Autrement dit, en revenant à la définition de la dérivée comme une limite :

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h, y_0) - f(x_0, y_0)}{h} \quad \text{et} \quad \frac{\partial f}{\partial y}(x_0, y_0) = \lim_{h \rightarrow 0} \frac{f(x_0, y_0 + h) - f(x_0, y_0)}{h}.$$

Plus généralement, pour une fonction  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  de plusieurs variables,  $\frac{\partial f}{\partial x_i}(x_1, \dots, x_n)$  est la dérivée partielle de  $f$  par rapport à la variable  $x_i$  au point  $(x_1, \dots, x_n) \in \mathbb{R}^n$ . C'est la dérivée en  $x_i$  de la fonction d'une variable  $x_i \mapsto f(x_1, \dots, x_n)$  où l'on considère fixes les variables  $x_j$  pour  $j \neq i$ .

**Notations.**

$$\frac{\partial f}{\partial x}(x, y) \quad \text{et} \quad \frac{\partial f}{\partial y}(x, y)$$

sont les analogues de l'écriture  $\frac{df}{dx}(x)$  pour l'écriture de la dérivée lorsqu'il n'y a qu'une seule variable. Le symbole «  $\partial$  » se lit « d rond ». Une autre notation est  $\partial_x f(x, y)$ ,  $\partial_y f(x, y)$  ou bien encore  $f'_x(x, y)$ ,  $f'_y(x, y)$ .

### 2. Calculs

Le calcul d'une dérivée partielle n'est pas plus compliqué que le calcul d'une dérivée.

**Méthode.** Pour calculer une dérivée partielle par rapport à une variable, il suffit de dériver par rapport à cette variable en considérant les autres variables comme des constantes.

#### Exemple 2.1

Calculer les dérivées partielles de la fonction  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  définie par  $f(x, y) = x^2 e^{3y}$ .

*Solution.*

Pour calculer la dérivée partielle  $\frac{\partial f}{\partial x}$ , par rapport à  $x$ , on considère que  $y$  est une constante et on dérive  $x^2 e^{3y}$  comme si c'était une fonction de la variable  $x$  uniquement :

$$\frac{\partial f}{\partial x}(x, y) = 2x e^{3y}.$$

Pour l'autre dérivée  $\frac{\partial f}{\partial y}$ , on considère que  $x$  est une constante et on dérive  $x^2 e^{3y}$  comme si c'était une fonction de  $y$  :

$$\frac{\partial f}{\partial y}(x, y) = 3x^2 e^{3y}.$$

**Exemple 2.2**

Pour  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  définie par  $f(x, y, z) = \cos(x + y^2)e^{-z}$  on a :

$$\begin{aligned}\frac{\partial f}{\partial x}(x, y, z) &= -\sin(x + y^2)e^{-z}, \\ \frac{\partial f}{\partial y}(x, y, z) &= -2y \sin(x + y^2)e^{-z}, \\ \frac{\partial f}{\partial z}(x, y, z) &= -\cos(x + y^2)e^{-z}.\end{aligned}$$

**Exemple 2.3**

Soit  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  définie par  $f(x_1, \dots, x_n) = x_1^2 + x_2^2 + \dots + x_n^2$ , alors pour  $i = 1, \dots, n$  :

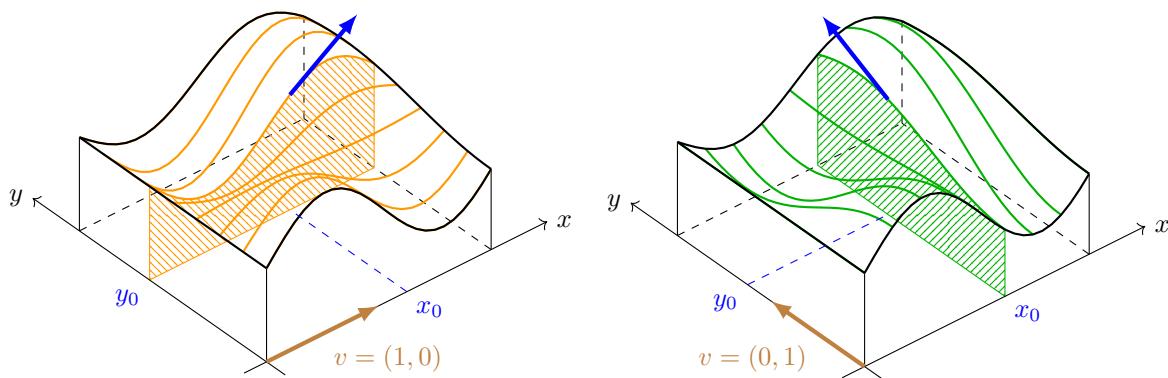
$$\frac{\partial f}{\partial x_i}(x_1, \dots, x_n) = 2x_i.$$

**3. Interprétation géométrique**

Pour une fonction d'une variable, la dérivée est la pente de la tangente au graphe de la fonction (le graphe étant alors une courbe). Pour une fonction de deux variables  $(x, y) \mapsto f(x, y)$ , les dérivées partielles indiquent les pentes au graphe de  $f$  selon certaines directions (le graphe étant ici une surface). Plus précisément :

- $\frac{\partial f}{\partial x}(x_0, y_0)$  est la pente du graphe de  $f$  en  $(x_0, y_0)$  suivant la direction de l'axe ( $Ox$ ). En effet cette pente est celle de la tangente à la courbe  $z = f(x, y_0)$  et est donnée par la dérivée de  $x \mapsto f(x, y_0)$  en  $x_0$ , c'est donc bien  $\frac{\partial f}{\partial x}(x_0, y_0)$ .
- $\frac{\partial f}{\partial y}(x_0, y_0)$  est la pente du graphe de  $f$  en  $(x_0, y_0)$  suivant la direction de l'axe ( $Oy$ ).

Sur la figure de gauche, la dérivée partielle  $\frac{\partial f}{\partial x}$  indique la pente de la tranche parallèle à l'axe ( $Ox$ ). Sur la figure de droite, la dérivée partielle  $\frac{\partial f}{\partial y}$  indique la pente de la tranche parallèle à l'axe ( $Oy$ ).



## V- Gradient

Le gradient est un vecteur dont les coordonnées sont les dérivées partielles. Il a de nombreuses applications géométriques car il donne l'équation des tangences aux courbes et surfaces de niveau. Surtout, il indique la direction dans laquelle la fonction varie le plus vite.

Le gradient est un vecteur qui remplace la notion de dérivée pour les fonctions de plusieurs variables. On sait que la dérivée permet de décider si une fonction est croissante ou décroissante. De même, le vecteur gradient indique la direction dans laquelle la fonction croît ou décroît le plus vite. Nous allons voir comment calculer de façon algorithmique le gradient grâce à la « différentiation automatique ».

### 1. Définition

#### Définition V.1

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  une fonction admettant des dérivées partielles. Le **gradient** de  $f$  en  $(x_0, y_0) \in \mathbb{R}^2$ , noté  $\overrightarrow{\text{grad}} f(x_0, y_0)$ , est le vecteur :

$$\overrightarrow{\text{grad}} f(x_0, y_0) = \begin{pmatrix} \frac{\partial f}{\partial x}(x_0, y_0) \\ \frac{\partial f}{\partial y}(x_0, y_0) \end{pmatrix}.$$

Les physiciens et les anglo-saxons notent souvent  $\nabla f(x, y)$  pour  $\overrightarrow{\text{grad}} f(x, y)$ . Le symbole  $\nabla$  se lit « nabla ».

Plus généralement, pour  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , le gradient de  $f$  en  $(x_1, \dots, x_n) \in \mathbb{R}^n$  est le vecteur :

$$\overrightarrow{\text{grad}} f(x_1, \dots, x_n) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x_1, \dots, x_n) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x_1, \dots, x_n) \end{pmatrix}.$$

#### Exemple 1.1

- $f(x, y) = x^2y^3$ ,  $\overrightarrow{\text{grad}} f(x, y) = \begin{pmatrix} 2xy^3 \\ 3x^2y^2 \end{pmatrix}$ . Au point  $(x_0, y_0) = (2, 1)$ ,  $\overrightarrow{\text{grad}} f(2, 1) = \begin{pmatrix} 4 \\ 12 \end{pmatrix}$ .
- $f(x, y, z) = x^2 \sin(yz)$ ,  $\overrightarrow{\text{grad}} f(x, y, z) = \begin{pmatrix} 2x \sin(yz) \\ x^2 z \cos(yz) \\ x^2 y \cos(yz) \end{pmatrix}$ .
- $f(x_1, \dots, x_n) = x_1^2 + x_2^2 + \dots + x_n^2$ ,  $\overrightarrow{\text{grad}} f(x_1, \dots, x_n) = \begin{pmatrix} 2x_1 \\ \vdots \\ 2x_n \end{pmatrix}$ .

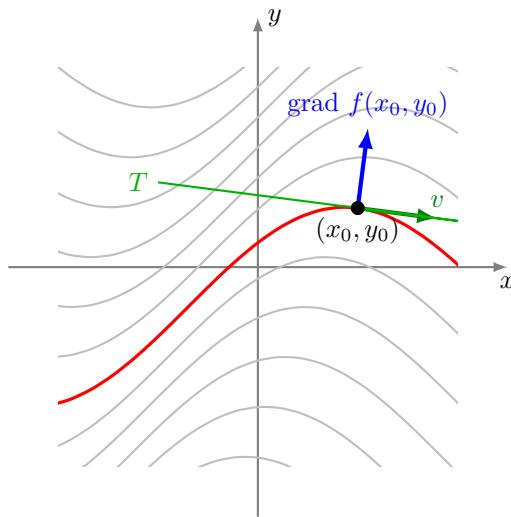
## 2. Tangentes aux lignes de niveau

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  une fonction différentiable. On considère les lignes de niveau  $f(x, y) = k$ .

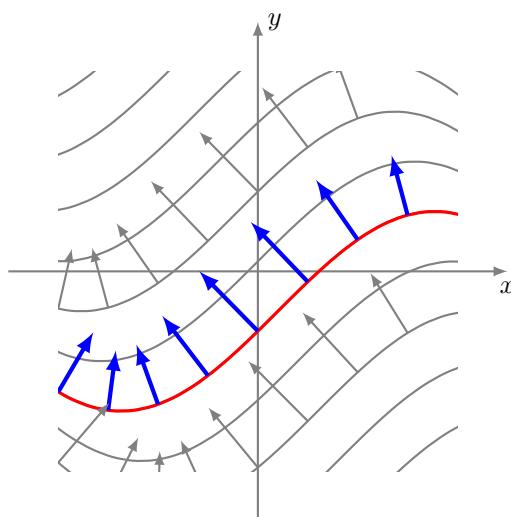
### Proposition 1.1

Le vecteur gradient  $\overrightarrow{\text{grad}} f(x_0, y_0)$  est orthogonal à la ligne de niveau de  $f$  passant au point  $(x_0, y_0)$ .

Sur ce premier dessin, sont dessinés la ligne de niveau passant par le point  $(x_0, y_0)$ , un vecteur tangent  $v$  en ce point et la tangente à la ligne de niveau. Le vecteur gradient est un vecteur du plan qui est orthogonal à la ligne de niveau en ce point.



À chaque point du plan, on peut associer un vecteur gradient. Ce vecteur gradient est orthogonal à la ligne de niveau passant par ce point. Nous verrons juste après comment savoir s'il est orienté « vers le haut » ou « vers le bas ».



Dans le cadre de notre étude, nous nous intéressons à l'équation de la tangente.

**Proposition 1.2**

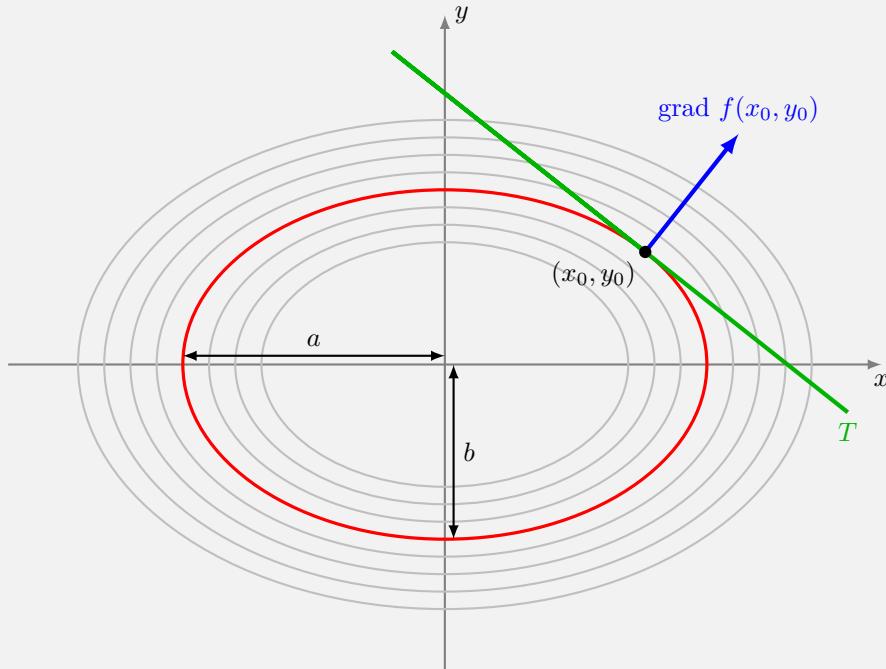
Au point  $(x_0, y_0)$ , l'équation de la tangente à la ligne de niveau de  $f$  est :

$$\frac{\partial f}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0)(y - y_0) = 0$$

pourvu que le gradient de  $f$  en ce point ne soit pas le vecteur nul.

**Exemple 2.1 : Tangentes à une ellipse**

Trouver les tangentes à l'ellipse  $\mathcal{E}$  d'équation  $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ .



Cette ellipse  $\mathcal{E}$  est la ligne de niveau  $f(x, y) = 1$  de la fonction  $f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2}$ . Les dérivées partielles en  $(x_0, y_0)$  sont :

$$\frac{\partial f}{\partial x}(x_0, y_0) = \frac{2x_0}{a^2} \quad \text{et} \quad \frac{\partial f}{\partial y}(x_0, y_0) = \frac{2y_0}{b^2}.$$

L'équation de la tangente à l'ellipse  $\mathcal{E}$  en ce point est donc :

$$\frac{2x_0}{a^2}(x - x_0) + \frac{2y_0}{b^2}(y - y_0) = 0.$$

Mais comme  $\frac{x_0^2}{a^2} + \frac{y_0^2}{b^2} = 1$ , l'équation de la tangente se simplifie en  $\frac{x_0}{a^2}x + \frac{y_0}{b^2}y = 1$ .

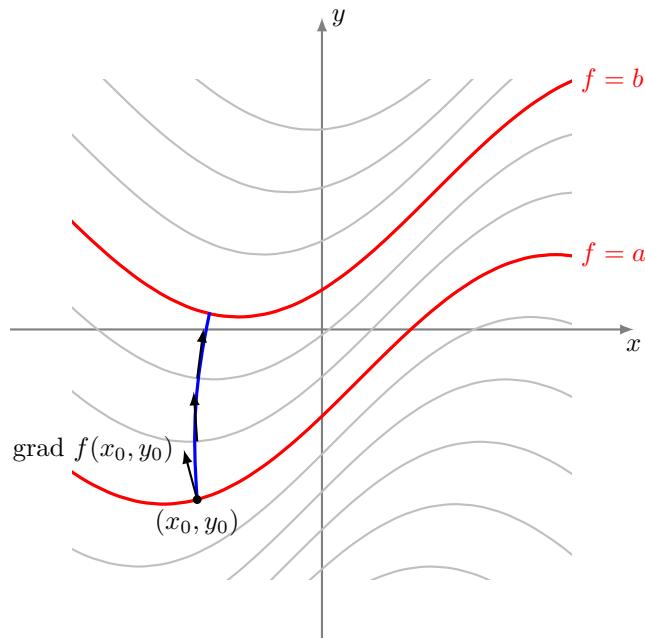
### 3. Lignes de plus forte pente

Considérons les lignes de niveau  $f(x, y) = k$  d'une fonction  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ . On se place en un point  $(x_0, y_0)$ . On cherche dans quelle direction se déplacer pour augmenter au plus vite la valeur de  $f$ .

#### Proposition 1.3

Le vecteur gradient  $\overrightarrow{\text{grad}} f(x_0, y_0)$  indique la direction de plus grande pente à partir du point  $(x_0, y_0)$ .

Autrement dit, si l'on veut, à partir d'un point donné  $(x_0, y_0)$  de niveau  $a$ , passer au niveau  $b > a$  le plus vite possible alors il faut démarrer en suivant la direction du gradient  $\overrightarrow{\text{grad}} f(x_0, y_0)$ .



Comme illustration, un skieur de descente, voulant optimiser sa course, choisira en permanence de s'orienter suivant la plus forte pente, c'est-à-dire dans le sens opposé au gradient.

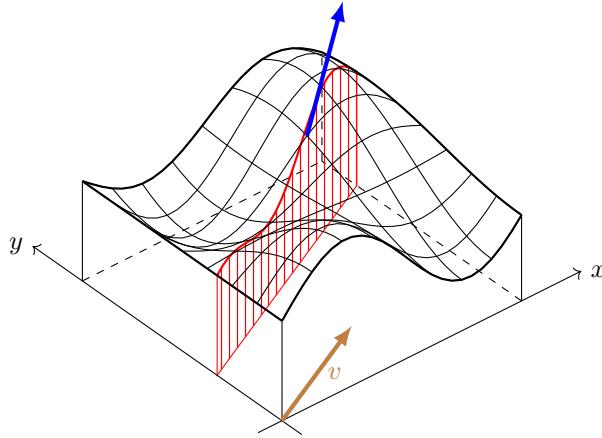
### 4. Dérivée directionnelle

Pour prouver que le gradient indique la ligne de la plus grande pente, nous avons besoin de généraliser la notion de dérivée partielle. Ce passage est plus technique et peut être ignoré en première lecture.

Soit  $v = \begin{pmatrix} h \\ k \end{pmatrix}$  un vecteur du plan. La **dérivée directionnelle** de  $f$  suivant le vecteur  $v$  en  $(x_0, y_0)$  est le nombre :

$$D_v f(x_0, y_0) = h \frac{\partial f}{\partial x}(x_0, y_0) + k \frac{\partial f}{\partial y}(x_0, y_0).$$

La dérivée directionnelle correspond à la pente de la fonction pour la tranche dirigée par le vecteur  $v$ .



Remarque : pour  $v = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  alors  $D_v f(x_0, y_0) = \frac{\partial f}{\partial x}(x_0, y_0)$  et pour  $v = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  alors  $D_v f(x_0, y_0) = \frac{\partial f}{\partial y}(x_0, y_0)$ .

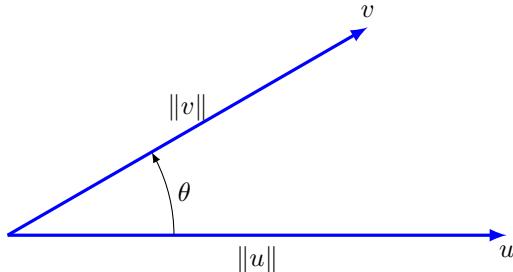
On rappelle que le **produit scalaire** de deux vecteurs  $u = \begin{pmatrix} x \\ y \end{pmatrix}$  et  $v = \begin{pmatrix} x' \\ y' \end{pmatrix}$  est donné par

$$\langle u | v \rangle = xx' + yy'.$$

On sait que le produit scalaire se calcule aussi géométriquement par :

$$\langle u | v \rangle = \|u\| \cdot \|v\| \cdot \cos(\theta)$$

où  $\theta$  est l'angle entre  $u$  et  $v$ .



Ainsi, on peut réécrire la dérivée directionnelle sous la forme :

$$D_v f(x_0, y_0) = \overrightarrow{\text{grad}} f(x_0, y_0) | v.$$

On peut maintenant prouver que le gradient indique la ligne de plus grande pente.

*Démonstration.* La dérivée suivant le vecteur non nul  $v$  au point  $(x_0, y_0)$  décrit la variation de  $f$  autour de ce point lorsqu'on se déplace dans la direction  $v$ . La direction selon laquelle la croissance est la plus grande est celle du gradient de  $f$ . En effet,

$$D_v f(x_0, y_0) = \overrightarrow{\text{grad}} f(x_0, y_0) | v = \|\overrightarrow{\text{grad}} f(x_0, y_0)\| \cdot \|v\| \cdot \cos \theta$$

où  $\theta$  est l'angle entre le vecteur  $\overrightarrow{\text{grad}} f(x_0, y_0)$  et le vecteur  $v$ . Le maximum est atteint lorsque l'angle  $\theta = 0$ , c'est-à-dire lorsque  $v$  pointe dans la même direction que  $\overrightarrow{\text{grad}} f(x_0, y_0)$ .  $\square$

## 5. Surface de niveau

Les résultats présentés ci-dessus pour les fonctions de deux variables se généralisent aux fonctions de trois variables ou plus. Commençons avec trois variables et une fonction  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ . Rappelons qu'un plan de  $\mathbb{R}^3$  passant par  $(x_0, y_0, z_0)$  et de vecteur normal  $n = (a, b, c)$  a pour équation cartésienne :

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0.$$

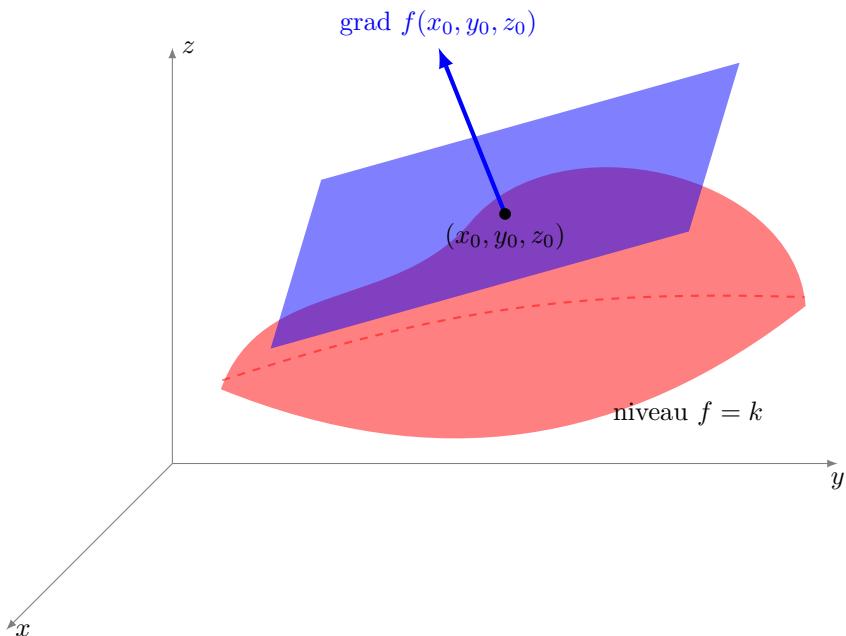
De même qu'il existe une droite tangente pour les lignes de niveau, il existe un **plan tangent** à une surface de niveau.

### Proposition 1.4

Le vecteur gradient  $\overrightarrow{\text{grad}} f(x_0, y_0, z_0)$  est orthogonal à la surface de niveau de  $f$  passant au point  $(x_0, y_0, z_0)$ . Autrement dit, l'équation du plan tangent à la surface de niveau de  $f$  en  $(x_0, y_0, z_0)$  est

$$\frac{\partial f}{\partial x}(x_0, y_0, z_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0, z_0)(y - y_0) + \frac{\partial f}{\partial z}(x_0, y_0, z_0)(z - z_0) = 0$$

pourvu que le gradient de  $f$  en ce point ne soit pas le vecteur nul.



Plus généralement pour  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $\overrightarrow{\text{grad}} f(x_1, \dots, x_n)$  est orthogonal à l'espace tangent à l'hypersurface de niveau  $f = k$  passant par le point  $(x_1, \dots, x_n) \in \mathbb{R}^n$  et ce vecteur gradient  $\overrightarrow{\text{grad}} f(x_1, \dots, x_n)$  indique la direction de plus grande pente à partir du point  $(x_1, \dots, x_n)$ .

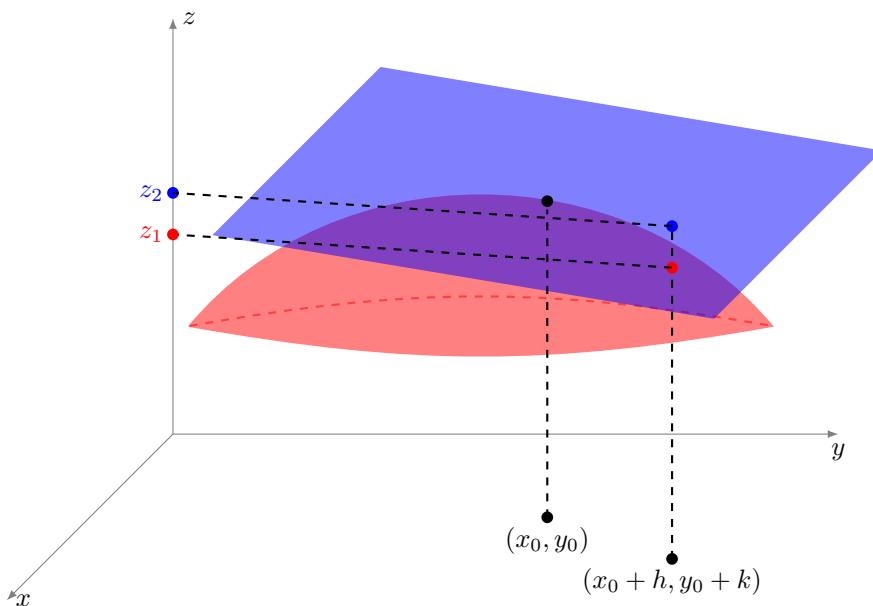
## 6. Calcul approché

Rappelez-vous que la dérivée nous a permis de faire des calculs approchés, par exemple pour estimer  $\sqrt{1.01}$  sans calculatrice (voir le chapitre « Dérivée »). Voici, en deux variables, l'analogue de la formule pour une variable :

$$f(x_0 + h, y_0 + k) \simeq f(x_0, y_0) + h \frac{\partial f}{\partial x}(x_0, y_0) + k \frac{\partial f}{\partial y}(x_0, y_0).$$

Cette approximation est valable pour  $h$  et  $k$  petits.

L'interprétation géométrique est la suivante : on approche le graphe de  $f$  en  $(x_0, y_0)$  par le plan tangent au graphe en ce point. Sur la figure ci-dessous sont représentés : le graphe de  $f$ , le plan tangent au-dessus du point  $(x_0, y_0)$ . La valeur  $z_1 = f(x_0 + h, y_0 + k)$  est la valeur exacte donnée par le point de la surface au dessus de  $(x_0 + h, y_0 + k)$ . On approche cette valeur par  $z_2 = f(x_0, y_0) + h \frac{\partial f}{\partial x}(x_0, y_0) + k \frac{\partial f}{\partial y}(x_0, y_0)$  donnée par le point du plan tangent au dessus de  $(x_0 + h, y_0 + k)$ .



### Exemple 6.1

Valeur approchée de  $f(1.002, 0.997)$  si  $f(x, y) = x^2y$ .

*Solution.* Ici  $(x_0, y_0) = (1, 1)$ ,  $h = 2 \times 10^{-3}$ ,  $k = -3 \times 10^{-3}$ ,  $\frac{\partial f}{\partial x}(x, y) = 2xy$ ,  $\frac{\partial f}{\partial y}(x, y) = x^2$ , donc  $\frac{\partial f}{\partial x}(x_0, y_0) = 2$ ,  $\frac{\partial f}{\partial y}(x_0, y_0) = 1$ . Ainsi

$$f(1 + h, 1 + k) \simeq f(1, 1) + 2h + k$$

donc

$$f(1.002, 0.997) \simeq 1 + 2 \times 2 \times 10^{-3} - 3 \times 10^{-3} \simeq 1.001.$$

Avec une calculatrice, on trouve  $f(1.002, 0.997) = 1.000992$  : l'approximation est bonne.

## 7. Minimum et maximum

### Définition V.2

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ .

- La fonction  $f$  admet un **minimum local** en  $(x_0, y_0)$  s'il existe un disque  $D$  centré en ce point tel que

$$f(x, y) \geq f(x_0, y_0) \quad \text{pour tout } (x, y) \in D.$$

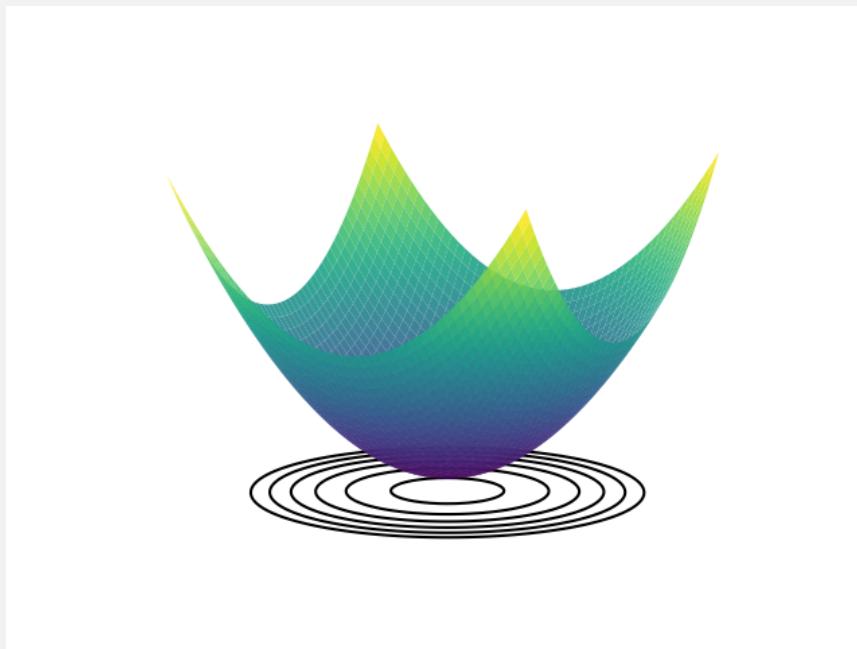
- La fonction  $f$  admet un **maximum local** en  $(x_0, y_0)$  pour lequel

$$f(x, y) \leq f(x_0, y_0) \quad \text{pour tout } (x, y) \in D.$$

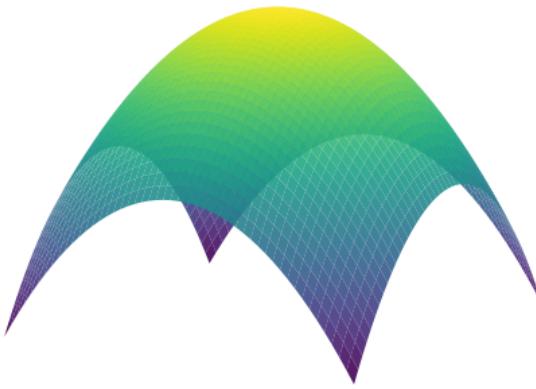
- On parle d'un **extremum local** pour un minimum ou un maximum local.

### Exemple 7.1

L'exemple type de minimum est celui de la fonction  $f(x, y) = x^2 + y^2$  en  $(0, 0)$ . Voici son graphe et ses lignes de niveau.



La fonction  $f(x, y) = -x^2 - y^2$  admet, elle, un maximum en  $(0, 0)$ .



### Proposition 1.5

Soit  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ . Si  $f$  admet un minimum ou un maximum local en  $(x_0, y_0)$  alors le gradient est le vecteur nul en ce point, autrement dit :

$$\frac{\partial f}{\partial x}(x_0, y_0) = 0 \quad \text{et} \quad \frac{\partial f}{\partial y}(x_0, y_0) = 0.$$

*Démonstration.* Prenons le cas d'un minimum local. La fonction d'une variable  $x \mapsto f(x, y_0)$  admet aussi un minimum en  $x_0$  donc sa dérivée est nulle en  $x_0$ , c'est-à-dire  $\frac{\partial f}{\partial x}(x_0, y_0) = 0$ . De même  $y \mapsto f(x_0, y)$  admet un minimum en  $y_0$  donc  $\frac{\partial f}{\partial y}(x_0, y_0) = 0$ .  $\square$

Dans la suite du cours nous chercherons les points pour lesquels une fonction donnée présente un minimum local. D'après la proposition précédente, ces points sont à chercher parmi les points en lesquels le gradient s'annule. On dira que  $(x_0, y_0)$  est un **point critique** de  $f$  si les deux dérivées partielles  $\frac{\partial f}{\partial x}(x_0, y_0)$  et  $\frac{\partial f}{\partial y}(x_0, y_0)$  s'annulent simultanément.

### Exemple 7.2

Chercher les points en lesquels  $f(x, y) = x^2 - y^3 + xy$  peut atteindre son minimum.

**Recherche des points critiques.** On calcule

$$\frac{\partial f}{\partial x}(x, y) = 2x + y \quad \text{et} \quad \frac{\partial f}{\partial y}(x, y) = -3y^2 + x.$$

On cherche les points  $(x, y)$  en lesquels les deux dérivées partielles s'annulent. Par l'annulation de la première dérivée, on a  $2x + y = 0$  donc  $y = -2x$ . Par l'annulation de la seconde dérivée, on a  $-3y^2 + x = 0$  ce qui donne par substitution  $-12x^2 + x = 0$ , ainsi  $x(-12x + 1) = 0$ . Donc soit  $x = 0$  et alors on a  $y = 0$ , soit  $x = \frac{1}{12}$  et alors  $y = -\frac{1}{6}$ . Bilan : il y a deux points critiques :

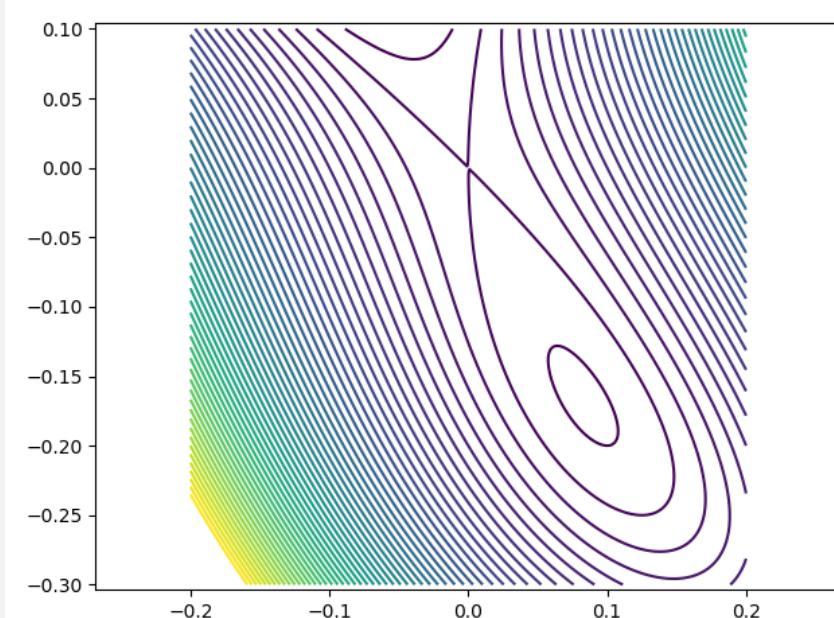
$$(0, 0) \quad \text{et} \quad \left( \frac{1}{12}, -\frac{1}{6} \right).$$

**Étude du point critique  $(0, 0)$ .** On a  $f(0, 0) = 0$  mais on remarque que  $f(0, y) = -y^3$

qui peut être négatif ou positif (selon le signe de  $y$  proche de 0), donc en  $(0, 0)$  il n'y a ni minimum ni maximum.

**Étude du point critique**  $(\frac{1}{12}, -\frac{1}{6})$ . Il existe un critère (que l'on ne décrira pas ici) qui permet de dire qu'en ce point  $f$  admet un minimum local.

Sur le dessin ci-dessous, le minimum est situé à l'intérieur du petit ovale, l'autre point critique en  $(0, 0)$  correspond à l'intersection de la ligne de niveau  $f = 0$  avec elle-même.

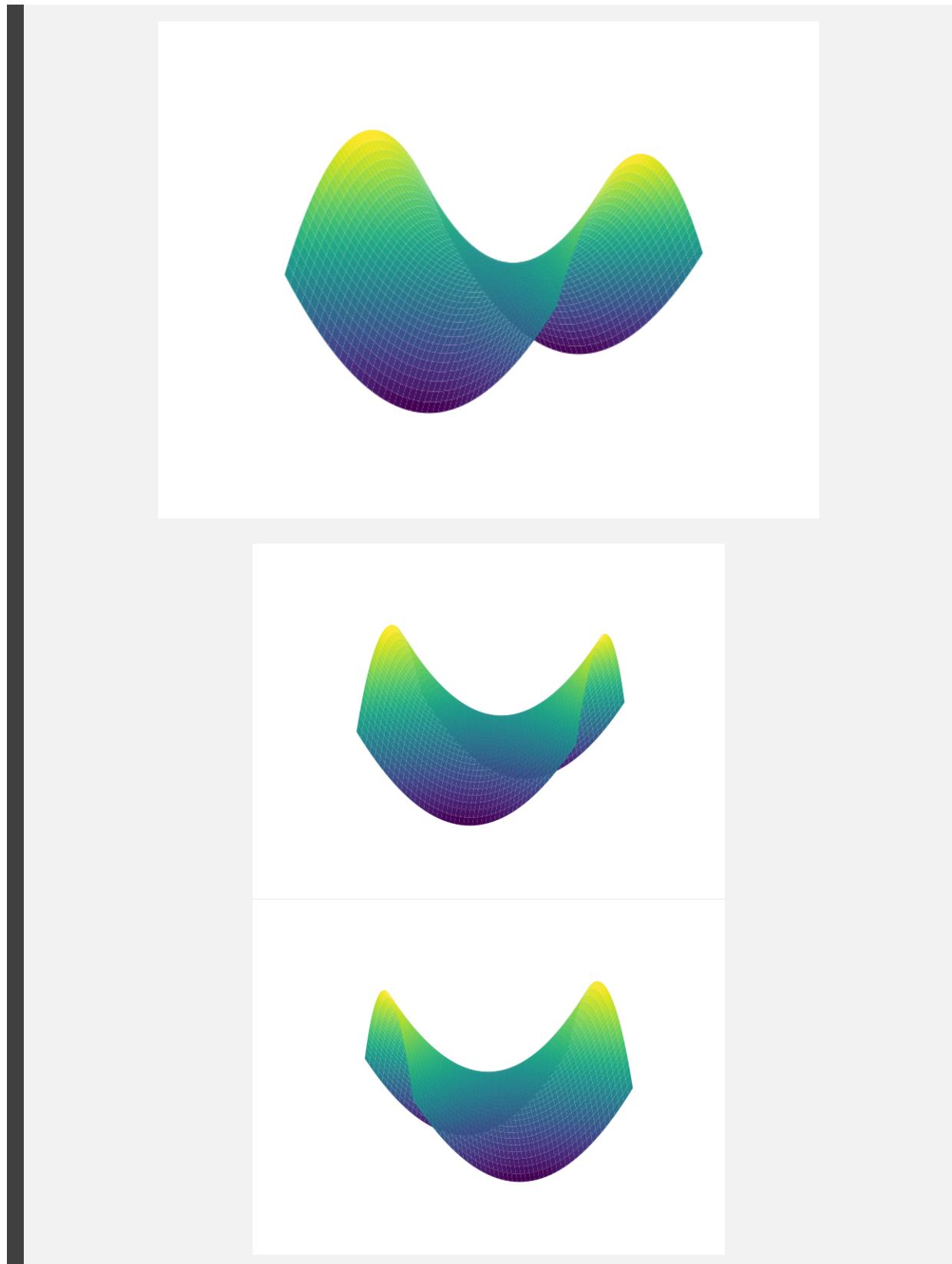


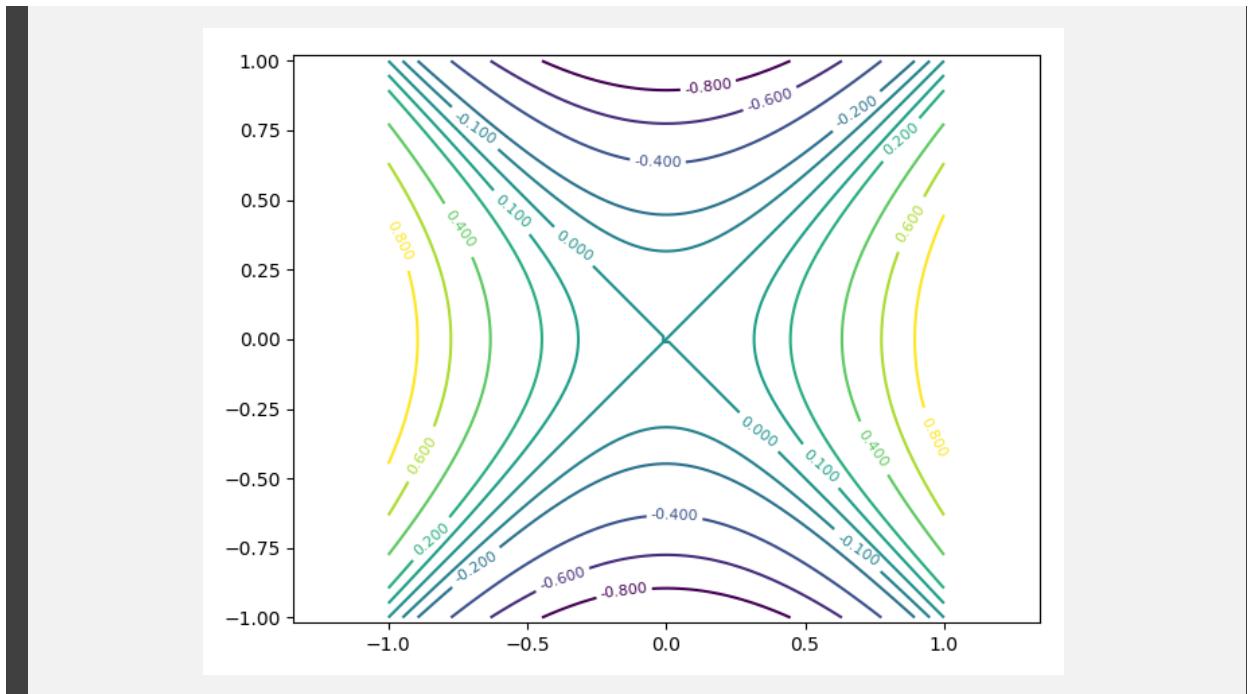
Sur l'exemple précédent, nous avons assez facilement calculé les points critiques à partir des deux équations à deux inconnues. Il faut prendre garde que ce n'est pas un système linéaire et que dans le cas d'une fonction plus compliquée il aurait été impossible de déterminer exactement les points critiques.

On note aussi dans l'exemple précédent que certains points critiques ne sont ni des maximums ni des minimums. L'exemple type, illustré ci-dessous, est celui d'un **col** appelé aussi **point-selle** en référence à sa forme de selle de cheval.

### Exemple 7.3

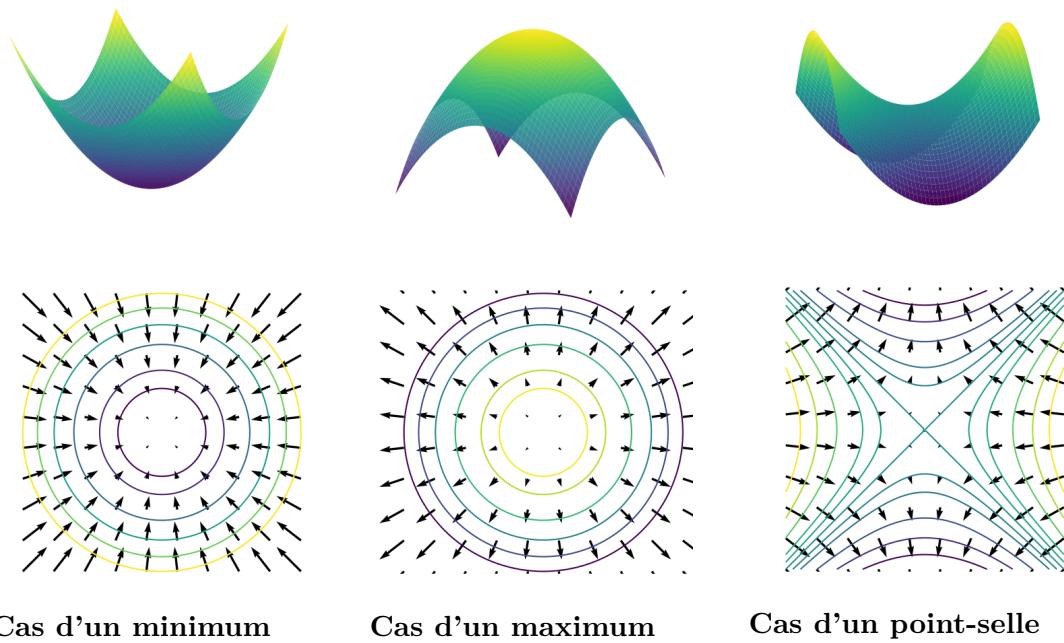
Soit  $f(x, y) = x^2 - y^2$ . Voici son graphe vu sous trois angles différents et ses lignes de niveau.





Comme il peut être difficile de calculer les points critiques de façon exacte, nous allons utiliser des méthodes numériques. L'idée qui sera détaillée dans le prochain chapitre est la suivante : comme le gradient indique la direction dans laquelle la fonction  $f$  croît le plus rapidement, nous allons suivre la direction opposée au gradient, pour laquelle  $f$  décroît le plus rapidement. Ainsi, partant d'un point  $(x_0, y_0)$  au hasard, on sait dans quelle direction se déplacer pour obtenir un nouveau point  $(x_1, y_1)$  en lequel  $f$  est plus petite. Et on recommence.

Sur les trois dessins ci-dessous, on a dessiné les lignes de niveau d'une fonction  $f$  ainsi que les vecteurs  $-\text{grad } f(x, y)$ . On voit que ces vecteurs pointent bien vers le minimum (figure de gauche), s'éloignent d'un maximum (figure centrale), le cas d'un point-selle est spécial (figure de droite). Dans tous les cas, la longueur des vecteurs gradients diminue à l'approche du point critique.

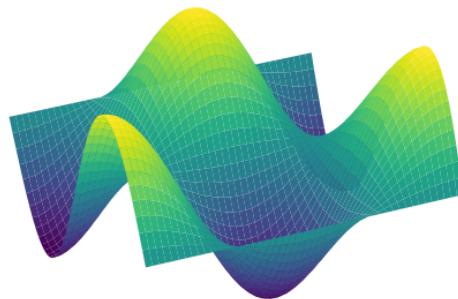


## VI-

**Descente de gradient classique**

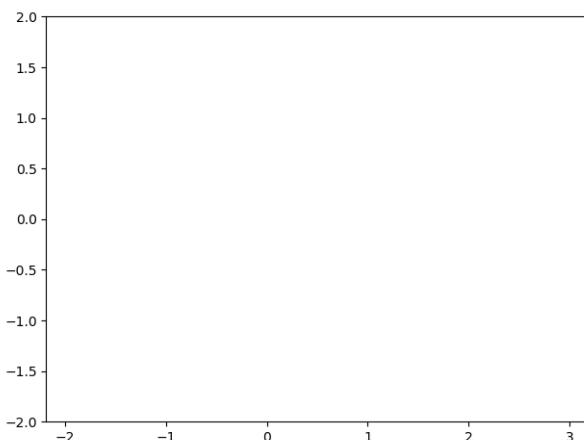
L'objectif de la méthode de descente de gradient est de trouver un minimum d'une fonction de plusieurs variables le plus rapidement possible. L'idée est très simple, on sait que le vecteur opposé au gradient indique une direction vers des plus petites valeurs de la fonction, il suffit donc de suivre d'un pas cette direction et de recommencer. Cependant, afin d'être encore plus rapide, il est possible d'ajouter plusieurs paramètres qui demandent pas mal d'ingénierie pour être bien choisis.

Imaginons une goutte d'eau en haut d'une colline. La goutte d'eau descend en suivant la ligne de plus grande pente et elle s'arrête lorsqu'elle atteint un point bas. C'est exactement ce que fait la descente de gradient : partant d'un point sur une surface, on cherche la pente la plus grande en calculant le gradient et on descend d'un petit pas, on recommence à partir du nouveau point jusqu'à atteindre un minimum local.

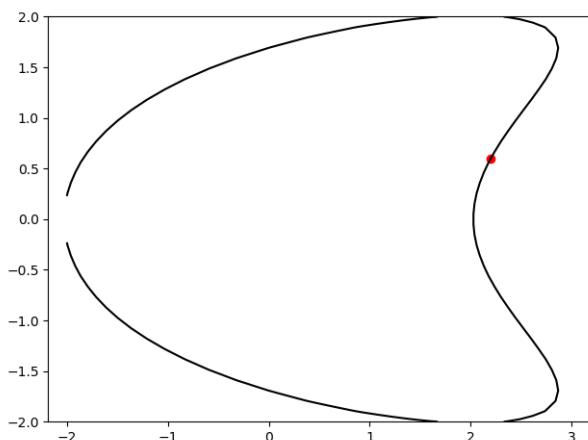
**1. Où est le minimum ?**

On nous donne une fonction  $f$  de deux variables  $(a, b)$  et nous cherchons un point  $(a_{\min}, b_{\min})$  en lequel  $f$  atteint un minimum. Voici la méthode expliquée par des dessins sur lesquels ont été tracées des lignes de niveau.

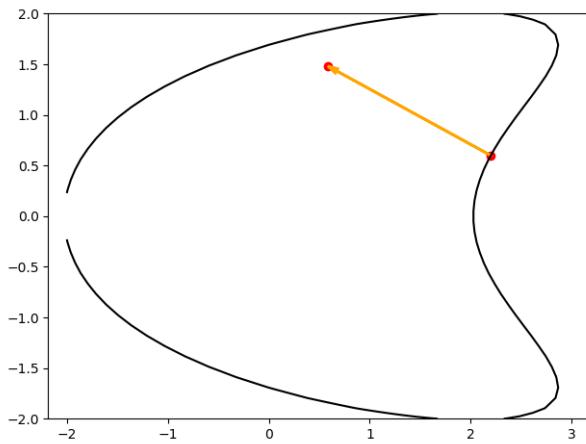
(a) Au départ : rien



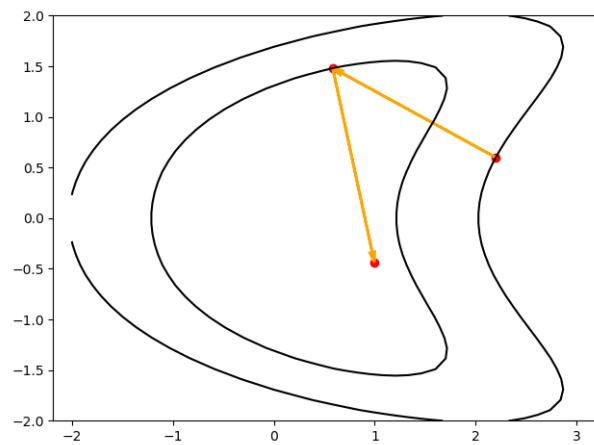
(b) On part d'un point au hasard



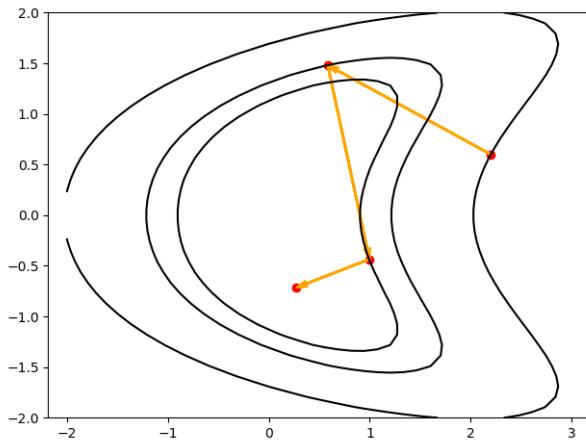
(c) On suit l'opposé du gradient



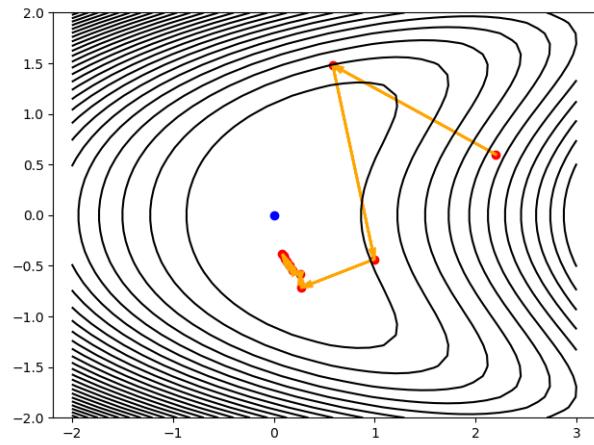
(d) Depuis le nouveau point, on suit l'opposé du gradient



(e) On répète le processus



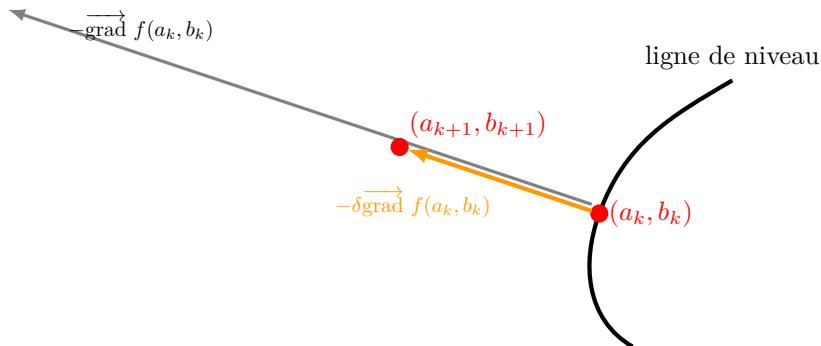
(f) La suite converge vers le minimum



**Figure (a).** Au départ nous n'avons aucune information globale sur  $f$ . La seule opération que l'on s'autorise c'est calculer  $\overrightarrow{\text{grad}} f(a, b)$  en certains points.

**Figure (b).** On choisit un point  $(a_0, b_0)$  au hasard. Si on note  $c_0 = f(a_0, b_0)$  la valeur de  $f$  en ce point, on sait que la ligne de niveau ( $f = c_0$ ) passe par  $(a_0, b_0)$ .

**Figure (c).** On calcule en ce point le gradient de  $f$ . On trace l'opposé du gradient :  $-\overrightarrow{\text{grad}} f(a_0, b_0)$ . On sait d'une part que la ligne de niveau est orthogonale à ce gradient et surtout que dans la direction de  $-\overrightarrow{\text{grad}} f(a_0, b_0)$ , les valeurs de  $f$  vont diminuer.



On se dirige alors dans la direction opposée au gradient d'un facteur  $\delta$  (par exemple  $\delta = 0.1$ ). On arrive à un point noté  $(a_1, b_1)$ . Par construction, si  $\delta$  est assez petit, la valeur  $c_1 = f(a_1, b_1)$  est plus petite que  $c_0$ .

**Figure (d).** On recommence depuis  $(a_1, b_1)$ . On calcule l'opposé du gradient en  $(a_1, b_1)$ , on se dirige dans cette nouvelle direction pour obtenir un point  $(a_2, b_2)$  où  $c_2 = f(a_2, b_2) < c_1$ .

**Figure (e).** On itère le processus pour obtenir une suite de points  $(a_k, b_k)$  pour lesquels  $f$  prend des valeurs de plus en plus petites.

**Figure (f).** On choisit de s'arrêter (selon une condition préalablement établie) et on obtient une valeur approchée  $(a_N, b_N)$  du point  $(a_{\min}, b_{\min})$  en lequel  $f$  atteint son minimum.

Évidemment avec la vision globale de la fonction, on se dit qu'on aurait pu choisir un point de départ plus près et que certaines directions choisies ne sont pas les meilleures. Mais souvenez-vous que l'algorithme est « aveugle », il ne calcule pas les valeurs de  $f$  en les  $(a_k, b_k)$  et n'a pas connaissance du comportement de  $f$  au voisinage de ces points.

## 2. Exemple en deux variables

Prenons l'exemple de  $f(a, b) = a^2 + 3b^2$  dont le minimum est bien évidemment atteint en  $(0, 0)$  et appliquons la méthode du gradient.

Nous aurons besoin de calculer la valeur du gradient en certains points par la formule :

$$\overrightarrow{\text{grad}} f(a, b) = \left( \frac{\partial f}{\partial a}(a, b), \frac{\partial f}{\partial b}(a, b) \right) = (2a, 6b).$$

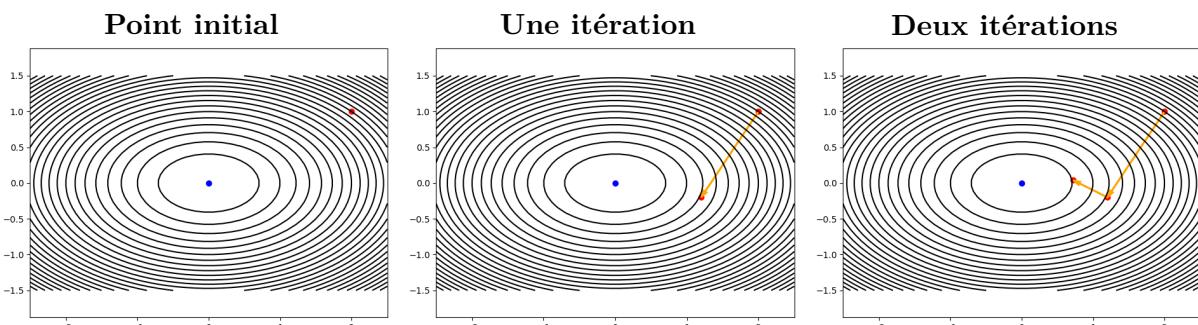
Tout d'abord, on part d'un point  $(a_0, b_0) = (2, 1)$  par exemple. Même si nous n'en avons pas besoin pour notre construction, on a  $f(a_0, b_0) = 7$ . On calcule  $\overrightarrow{\text{grad}} f(a_0, b_0) = (4, 6)$ . On fixe le facteur  $\delta = 0.2$ . On se déplace dans la direction opposée à ce gradient :

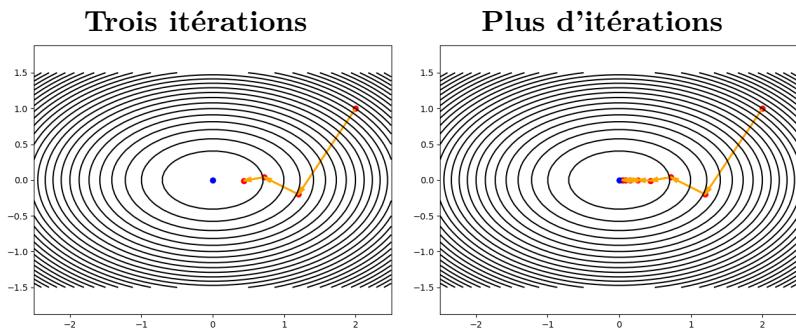
$$(a_1, b_1) = (a_0, b_0) - \delta \overrightarrow{\text{grad}} f(a_0, b_0) = (2, 1) - 0.2(4, 6) = (2, 1) - (0.8, 1.2) = (1.2, -0.2).$$

On note que  $f(a_1, b_1) = 1.56$  est bien plus petit que  $f(a_0, b_0)$ . On recommence ensuite depuis  $(a_1, b_1)$ . En quelques étapes les valeurs de  $f$  tendent vers la valeur minimale et, dans notre cas, la suite converge vers  $(0, 0)$  (les valeurs sont approchées).

$k$	$(a_k, b_k)$	$\overrightarrow{\text{grad}} f(a_k, b_k)$	$f(a_k, b_k)$
0	$(2, 1)$	$(4, 6)$	7
1	$(1.2, -0.2)$	$(2.4, -1.20)$	1.56
2	$(0.72, 0.04)$	$(1.44, 0.24)$	0.523
3	$(0.432, -0.008)$	$(0.864, -0.048)$	0.186
4	$(0.2592, 0.0016)$	$(0.5184, 0.0096)$	0.067
5	$(0.15552, -0.00032)$	$(0.31104, -0.00192)$	0.024
...			
10	$(0.012, 1.02 \cdot 10^{-7})$	$(0.024, 6.14 \cdot 10^{-7})$	0.00014
...			
20	$(7.31 \cdot 10^{-5}, 1.04 \cdot 10^{-14})$	$(1.46 \cdot 10^{-4}, 6.29 \cdot 10^{-14})$	$5.34 \cdot 10^{-9}$

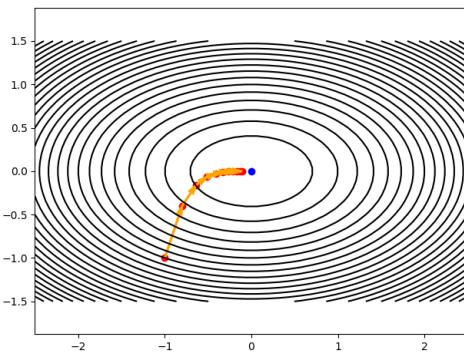
Voici les graphiques des premières itérations :





Que se passe-t-il si l'on part d'un autre point ? Partons cette fois de  $(a_0, b_0) = (-1, -1)$  et fixons le pas à  $\delta = 0.1$ . Alors  $(a_1, b_1) = (-0.8, -0.4)$ ,  $(a_2, b_2) = (-0.64, -0.16)\dots$  La suite converge également vers  $(0, 0)$ .

**Partant de  $(-1, -1)$  avec  $\delta = 0.1$**



### 3. Exemples en une variable

La descente de gradient fonctionne aussi très bien pour les fonctions d'une seule variable et sa visualisation est instructive.

#### Exemple 3.1 : C

nsidérons la fonction  $f : \mathbb{R} \rightarrow \mathbb{R}$  définie par

$$f(a) = a^2 + 1.$$

Il s'agit de trouver la valeur en laquelle  $f$  atteint son minimum, c'est clairement  $a_{\min} = 0$  pour lequel  $f(a_{\min}) = 1$ . Retrouvons ceci par la descente de gradient.

Partant d'une valeur  $a_0$  quelconque, la formule de récurrence est :

$$a_{k+1} = a_k - \delta \overrightarrow{\text{grad}} f(a)$$

où  $\delta$  est le pas, choisi assez petit, et  $\overrightarrow{\text{grad}} f(a) = f'(a) = 2a$ . Autrement dit :

$$a_{k+1} = a_k - 2\delta a_k.$$

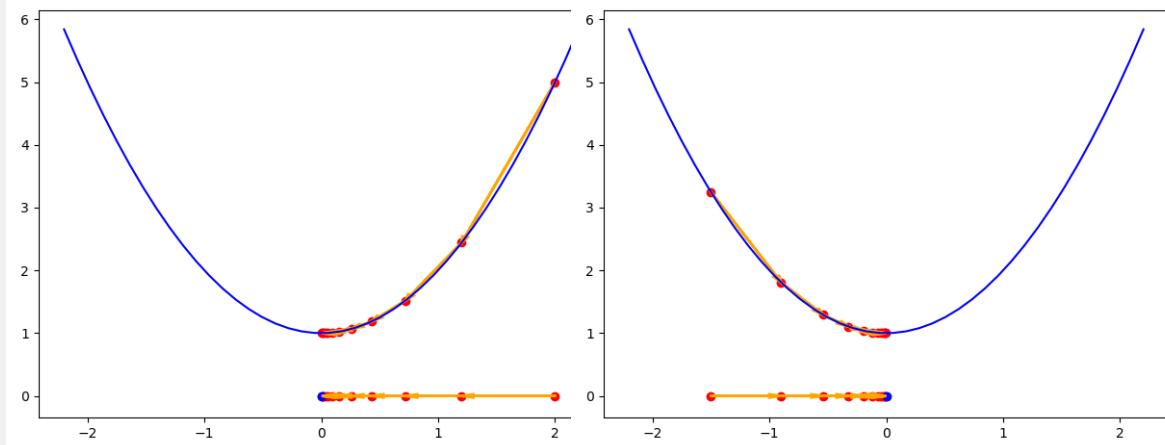
Voici le tableau des valeurs pour un pas  $\delta = 0.2$  et une valeur initiale  $a_0 = 2$ .

$k$	$a_k$	$f'(a_k) = \overrightarrow{\text{grad}} f(a_k)$	$f(a_k)$
0	2	4	5
1	1.2	2.4	2.44
2	0.72	1.44	1.5184
3	0.43	0.86	1.1866
4	0.25	0.5184	1.0671
5	0.15	0.31	1.0241
6	0.093	0.186	1.0087
7	0.055	0.111	1.0031
8	0.033	0.067	1.0011
9	0.020	0.040	1.0004
10	0.012	0.024	1.0001

Voici la version graphique de ces 10 premières itérations (figure de gauche). Si l'on change le point initial, ( $a_0 = -1.5$  sur la figure de droite) alors la suite  $(a_k)$  converge vers la même valeur  $a_{\min} = 0$ .

$$\delta = 0.2 \quad a_0 = 2$$

$$\delta = 0.2 \quad a_0 = -1.5$$



Il faut bien comprendre ce graphique : la suite des points  $(a_k)$  se lit sur l'axe des abscisses. Les vecteurs montrent les itérations. Il est plus facile de comprendre l'algorithme sur le graphe de  $f$ . Sur ce graphe, on reporte les points  $(a_k, f(a_k))$ , ce qui permet de bien comprendre que les valeurs  $f(a_k)$  décroissent rapidement. On note aussi que le gradient (ici  $f'(a_k)$ ) diminue à l'approche du minimum, ce qui se traduit par des vecteurs (c'est-à-dire l'écart entre deux points successifs) de plus en plus petits.

Justifions l'algorithme et l'intervention du gradient dans le cas d'une variable. Si la fonction est croissante sur un intervalle,  $f'(a) > 0$  pour tout  $a$  dans cet intervalle et la formule

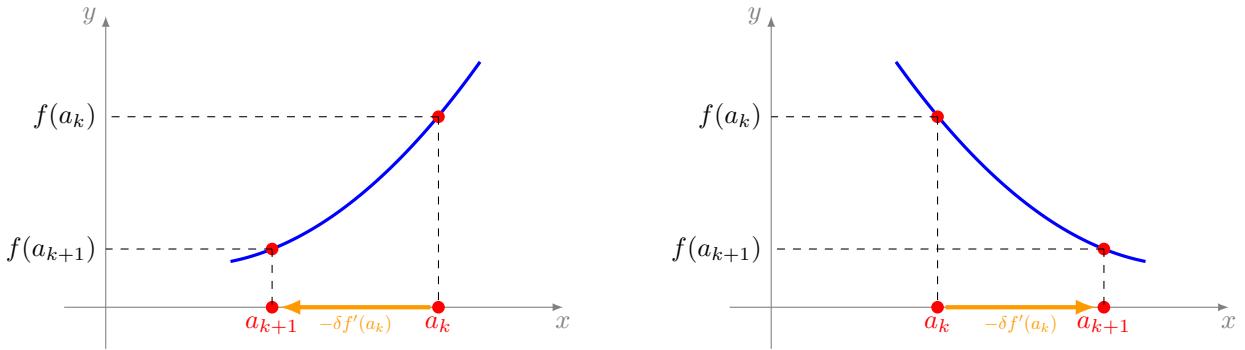
$$a_{k+1} = a_k - \delta f'(a_k) \quad \text{donne} \quad a_{k+1} < a_k.$$

Ainsi  $f(a_{k+1}) < f(a_k)$  et l'ordonnée du point  $(a_{k+1}, f(a_{k+1}))$  est donc inférieure à celle du point  $(a_k, f(a_k))$ . Par contre, si  $f$  est décroissante alors  $f'(a) < 0$  et

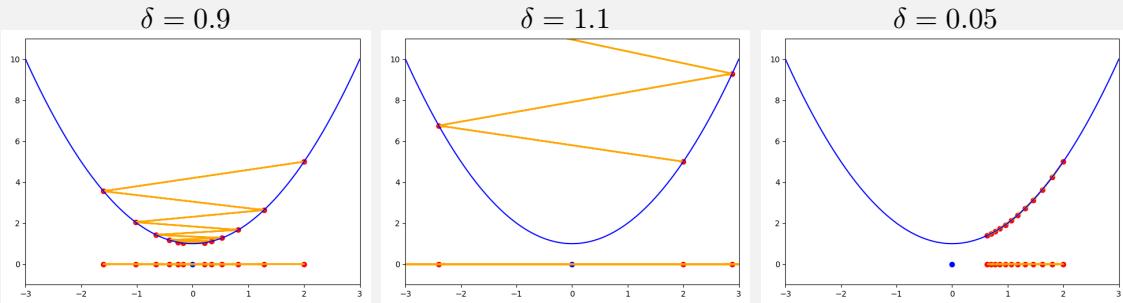
$$a_{k+1} = a_k - \delta f'(a_k) \quad \text{donne} \quad a_{k+1} > a_k,$$

ce qui implique de nouveau  $f(a_{k+1}) < f(a_k)$  (car  $f$  est décroissante).

Dans tous les cas, l'ordonnée du point  $(a_{k+1}, f(a_{k+1}))$  est inférieure à celle du point  $(a_k, f(a_k))$ .

**Exemple 3.2**

Le choix du paramètre  $\delta$  est important. Reprenons la fonction  $f$  définie par  $f(x) = x^2 + 1$  et testons différentes « mauvaises » valeurs du pas  $\delta$  (avec toujours  $a_0 = 2$ ).



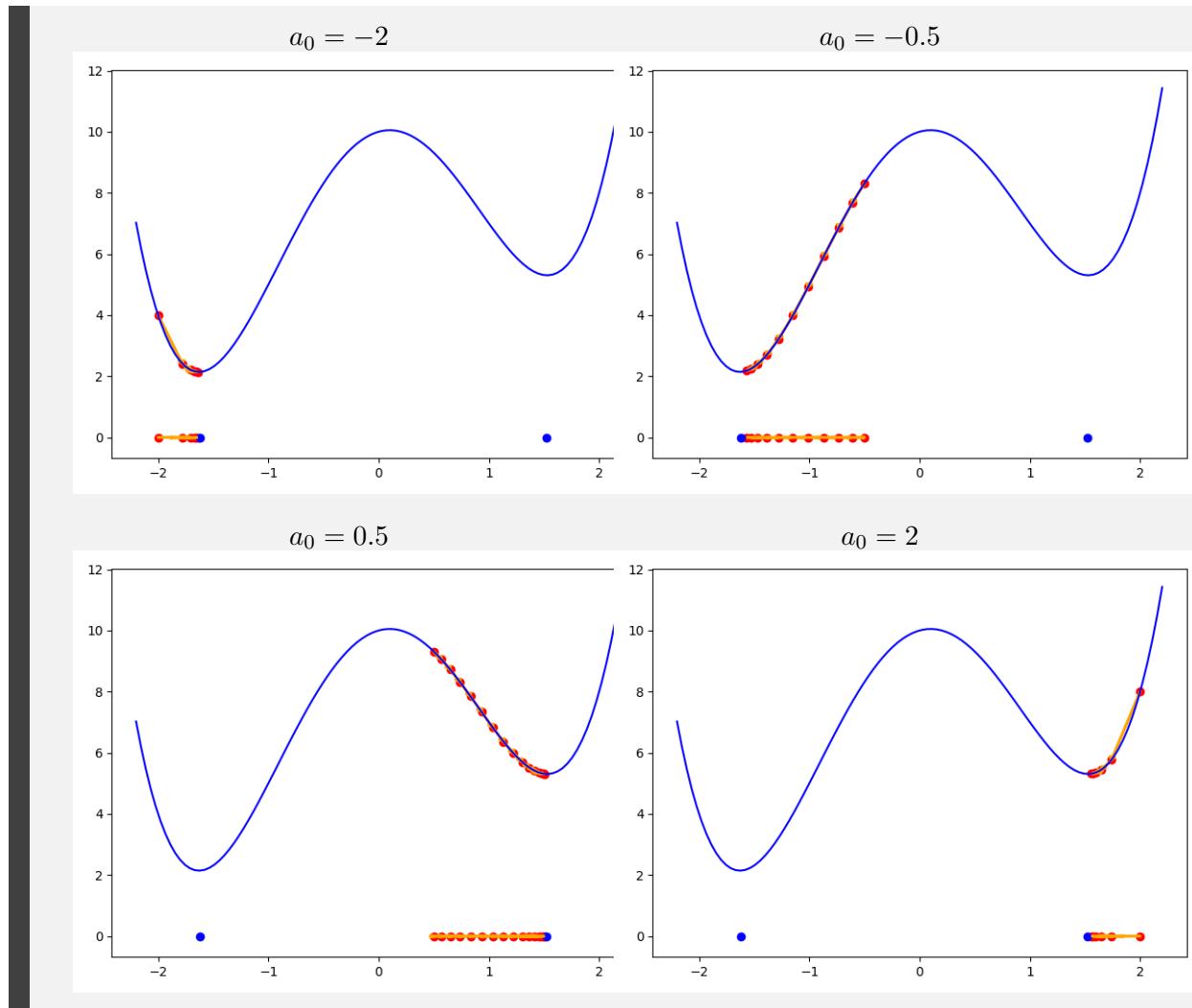
- Pour  $\delta = 0.9$ , la suite  $(a_k)$  tend bien vers  $a_{\min} = 0$ . Les ordonnées sont bien décroissantes mais comme  $\delta$  est trop grand, la suite des points oscille de part et d'autre du minimum.
- Pour  $\delta = 1.1$ , la suite  $(a_k)$  diverge. Les ordonnées augmentent, la suite des points oscille et s'échappe. Cette valeur de  $\delta$  ne donne pas de convergence vers un minimum.
- Pour  $\delta = 0.05$ , la suite  $(a_k)$  tend bien vers  $a_{\min}$  mais, comme  $\delta$  est trop petit, il faudrait beaucoup d'itérations pour arriver à une approximation raisonnable.

**Exemple 3.3**

Le choix du point de départ est également important surtout lorsqu'il existe plusieurs minima locaux. Soit la fonction  $f$  définie par :

$$f(a) = a^4 - 5a^2 + a + 10.$$

Cette fonction admet deux minima locaux. La suite  $(a_k)$  de la descente de gradient converge vers l'un de ces deux minima selon le choix du point initial  $a_0$  (ici  $\delta = 0.02$ ).



### Exemple 3.4

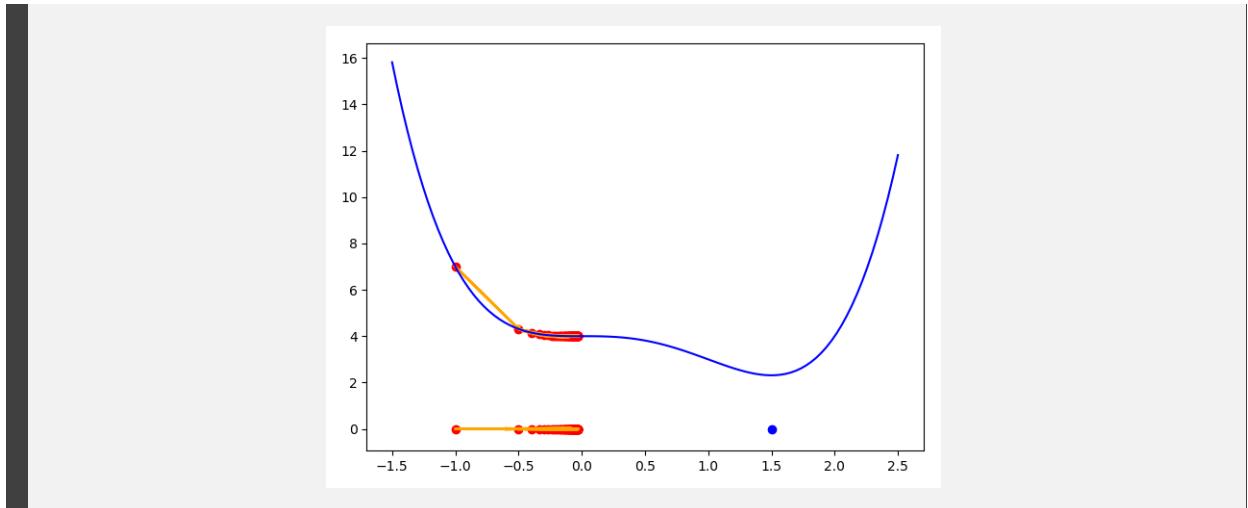
Les points-selles posent également problème.

La fonction  $f$  définie par

$$f(a) = a^4 - 2a^3 + 4$$

a pour dérivée  $f'(a) = 4a^3 - 6a^2$  qui s'annule en  $a = 0$  qui est l'abscisse d'un point-selle (ni un minimum ni un maximum, en fait la fonction est strictement décroissante autour de  $a = 0$ ). La dérivée s'annule aussi en  $a = \frac{3}{2}$  où est atteint le minimum global.

Voici les 100 premières itérations pour la descente de gradient en partant de  $a_0 = -1$  (avec  $\delta = 0.05$ ) : la suite  $a_k$  converge vers 0 qui n'est pas le minimum recherché.



#### 4. Algorithme du gradient

Formalisons un peu les choses pour mettre en évidence l'idée générale et les problèmes techniques qui surviennent.

##### Algorithme de la descente de gradient.

Soit une fonction  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $P \mapsto f(P)$  de plusieurs variables, avec  $P = (a_1, \dots, a_n)$ , dont on sait calculer le gradient  $\overrightarrow{\text{grad}} f(P)$ .

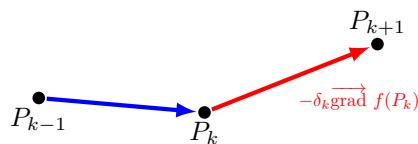
##### Données.

- Un point initial  $P_0 \in \mathbb{R}^n$ .
- Un niveau d'erreur  $\epsilon > 0$ .

**Itération.** On calcule une suite de points  $P_1, P_2, \dots \in \mathbb{R}^n$  par récurrence de la façon suivante. Supposons que l'on ait déjà obtenu le point  $P_k$  :

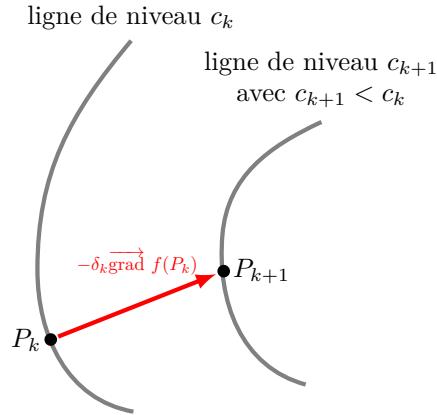
- on calcule  $\overrightarrow{\text{grad}} f(P_k)$ ,
- on choisit un pas  $\delta_k$  et on calcule

$$P_{k+1} = P_k - \delta_k \overrightarrow{\text{grad}} f(P_k).$$



**Arrêt.** On s'arrête lorsque  $\|\overrightarrow{\text{grad}} f(P_k)\| \leq \epsilon$ .

- Évidemment, plus on choisit le point initial  $P_0$  proche d'un minimum local, plus l'algorithme va aboutir rapidement. Mais comme on ne sait pas où est ce minimum local (c'est ce que l'on cherche), le plus simple est de choisir un  $P_0$  au hasard.
- Le choix du pas  $\delta_k$  est crucial. On sait que l'on peut choisir  $\delta_k$  assez petit de façon à avoir  $f(P_{k+1}) \leq f(P_k)$  car dans la direction de  $-\overrightarrow{\text{grad}} f(P_k)$  la fonction  $f$  décroît.



On peut fixer à l'avance un pas  $\delta$  commun à toutes les itérations, par exemple  $\delta = 0.01$ . On pourrait également tester à chaque itération plusieurs valeurs de  $\delta$  par balayage ( $\delta = 0.001$ , puis  $\delta = 0.002\dots$ ) et choisir pour  $\delta_k$  celui en lequel  $f$  prend la plus petite valeur.

- Le critère d'arrêt assure qu'en  $P_k$  le gradient est très petit. Cela ne garantit pas que ce point soit proche d'un minimum local (et encore moins d'un minimum global). Souvenez-vous : en un minimum local le gradient est nul, mais ce n'est pas parce que le gradient est nul que l'on a atteint un minimum local, cela pourrait être un point-selle voire un maximum local.
- Dans la pratique, on ne définira pas de seuil d'erreur  $\epsilon$ , mais un nombre d'itérations fixé à l'avance.
- Il est important de calculer  $\vec{\text{grad}} f(a_1, \dots, a_n)$  rapidement. On pourrait bien sûr calculer une approximation de chacune des dérivées partielles  $\frac{\partial f}{\partial a_i}(a_1, \dots, a_n)$  comme un limite. Mais pour gagner en temps et en précision, on préfère que ce calcul soit fait à l'aide de son expression exacte.

## VII- Optimisation

Nous allons d'une part résoudre des problèmes d'optimisation : quelle droite approche au mieux un nuage de points, et d'autre part étudier comment améliorer le choix du pas  $\delta$ .

### 1. Faire varier le pas

On se concentre d'abord sur le choix du **pas**  $\delta$  (*learning rate*).

Rappelons tout d'abord que lorsque l'on se rapproche d'un point minimum, le gradient tend vers 0. Le vecteur  $\vec{\text{grad}} f(P_k)$  tend donc vers 0 à l'approche du minimum, même si  $\delta$  reste constant.

Cependant, il faut choisir  $\delta$  ni trop grand, ni trop petit :  $\delta$  ne doit pas être trop grand car sinon les points  $P_k$  vont osciller autour du minimum, mais si  $\delta$  est trop petit alors les points  $P_k$  ne s'approcheront du minimum qu'au bout d'un temps très long. Une solution est de faire varier  $\delta$ . Pour les premières itérations, on choisit un  $\delta_k$  assez grand, puis de plus en plus petit au fil des itérations.

Voici différentes formules possibles, à chaque fois  $\delta_0$  est le pas initial (par exemple  $\delta_0 = 0.1$  ou  $\delta_0 = 0.01$ ).

**Décroissance linéaire.**

$$\delta_k = \frac{\delta_0}{k + 1}.$$

**Décroissance quadratique.**

$$\delta_k = \frac{\delta_0}{(k + 1)^2}.$$

**Décroissance exponentielle.**

$$\delta_k = \delta_0 e^{-\beta k}$$

où  $\beta$  est une constante positive.

### Décroissance linéaire utilisée par keras

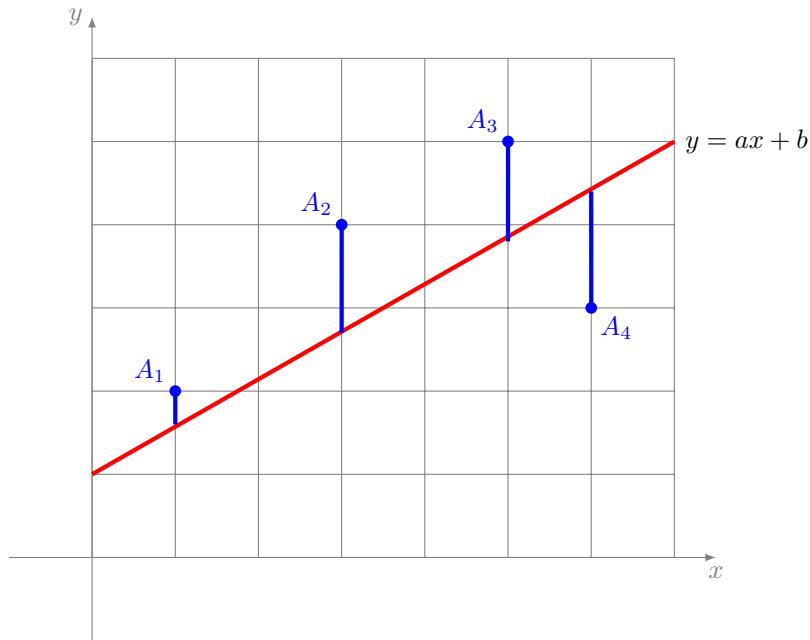
$$\delta_k = \frac{\delta_0}{\alpha k + 1}$$

où  $\alpha \geq 0$  est une constante (appelée *decay*). Si  $\alpha = 0$  alors  $\delta_k$  est constant (et vaut  $\delta_0$ ). L'usage courant est d'utiliser des valeurs de  $\alpha$  entre  $10^{-4}$  et  $10^{-6}$ .

Terminons par rappeler que le bon choix d'un  $\delta$  ou des  $\delta_k$  n'a rien d'évident, il s'obtient soit par test à la main, soit par des expérimentations automatiques, mais à chaque fois il doit être adapté à la situation.

## 2. Régression linéaire $y = ax + b$

On considère un ensemble de  $N$  points  $A_i = (x_i, y_i)$ ,  $i = 1, \dots, N$ . L'objectif est de trouver l'équation  $y = ax + b$  de la droite qui approche au mieux tous ces points. Précisons ce que veut dire « approcher au mieux » : il s'agit de minimiser la somme des carrés des distances verticales entre les points et la droite.



La formule qui donne l'erreur est :

$$E(a, b) = \sum_{i=1}^N (y_i - (ax_i + b))^2,$$

autrement dit

$$E(a, b) = (y_1 - (ax_1 + b))^2 + \dots + (y_N - (ax_N + b))^2.$$

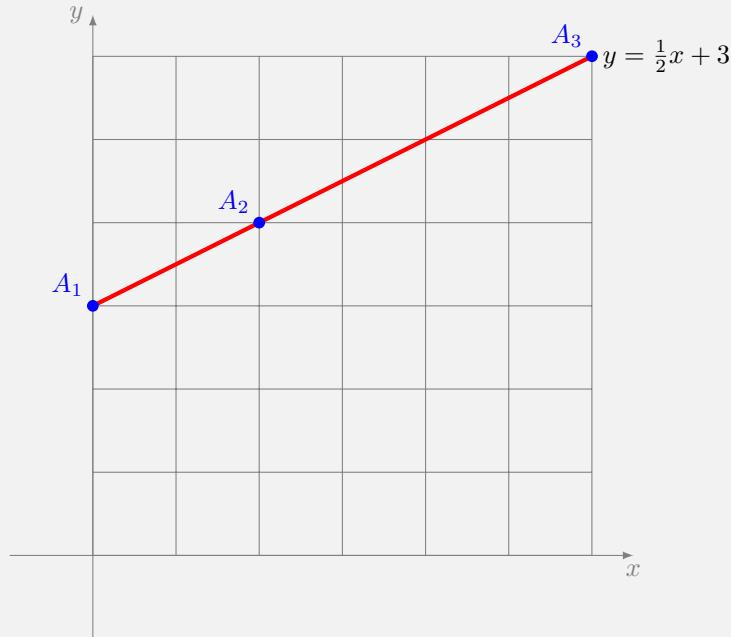
Remarquons que l'on a toujours  $E(a, b) \geq 0$ . Si par exemple tous les points sont alignés, alors on peut trouver  $a$  et  $b$  tels que  $E(a, b) = 0$ . Quand ce n'est pas le cas, on cherche  $a$  et  $b$  qui rendent  $E(a, b)$  le plus petit possible. Il s'agit donc bien ici de minimiser une fonction de deux variables (les variables sont  $a$  et  $b$ ).

Nous allons appliquer la méthode de la descente de gradient à la fonction  $E(a, b)$ . Pour cela nous aurons besoin de calculer son gradient :

$$\overrightarrow{\text{grad}} E(a, b) = \left( \frac{\partial E}{\partial a}(a, b), \frac{\partial E}{\partial b}(a, b) \right) = \left( \sum_{i=1}^N -2x_i(y_i - (ax_i + b)), \sum_{i=1}^N -2(y_i - (ax_i + b)) \right).$$

**Exemple 2.1**

Prenons d'abord l'exemple de trois points  $A_1 = (0, 3)$ ,  $A_2 = (2, 4)$  et  $A_3 = (6, 6)$  qui sont alignés.



La fonction  $E(a, b)$  s'écrit :

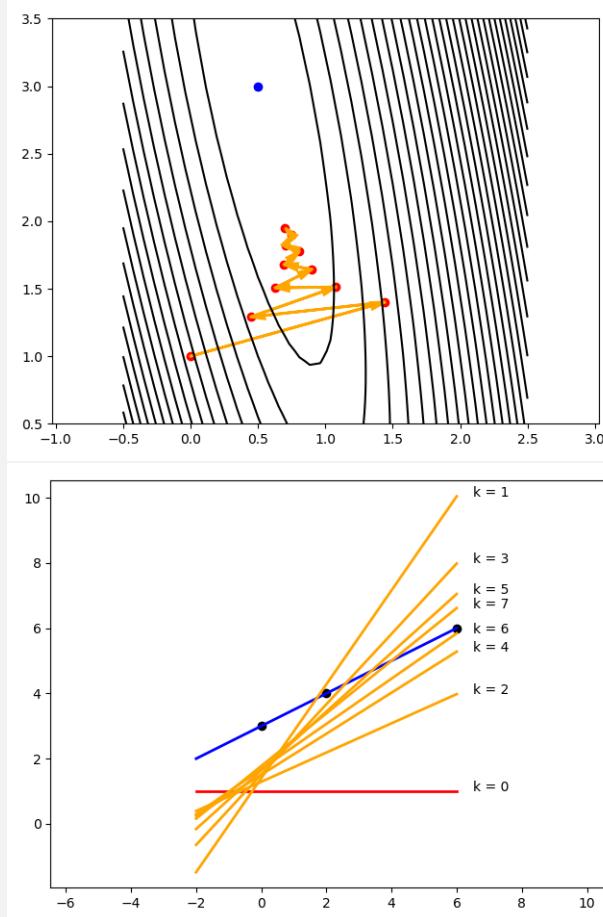
$$E(a, b) = (3 - b)^2 + (4 - (2a + b))^2 + (6 - (6a + b))^2.$$

Partons arbitrairement de  $(a_0, b_0) = (0, 1)$  (qui correspond à la droite horizontale d'équation  $y = 1$ ). Voici les valeurs successives  $(a_k, b_k)$  obtenues par la méthode de descente de gradient pour un pas  $\delta = 0.02$ .

$k$	$(a_k, b_k)$	$\overrightarrow{\text{grad}} E(a_k, b_k)$	$E(a_k, b_k)$
0	(0, 1)	(-72, -20)	38
1	(1.44, 1.4)	(49.60, 5.44)	18.96
2	(0.44, 1.29)	(-31.50, -11.08)	10.28
3	(1.07, 1.51)	(22.44, 0.32)	6.24
4	(0.62, 1.50)	(-13.57, -6.89)	4.27
5	(0.90, 1.64)	(10.35, -1.72)	3.24

Au bout de 100 itérations, on obtient  $a_{100} \simeq 0.501$  et  $b_{100} \simeq 2.99$  (avec un gradient et une erreur presque nuls). C'est bien la droite  $y = \frac{1}{2}x + 3$  qui passe par les trois points.

Sur la figure de gauche ci-dessous, sont dessinés, dans le plan de coordonnées  $(a, b)$ , les premiers points  $(a_k, b_k)$  qui convergent (lentement et en oscillant) vers  $(\frac{1}{2}, 3)$ . Sur la figure de droite sont tracées, dans le plan de coordonnées  $(x, y)$ , les droites d'équation  $y = a_k x + b_k$  pour les premières valeurs de  $k$ . Il est beaucoup plus difficile d'appréhender la convergence des droites (vers la droite d'équation  $y = \frac{1}{2}x + 3$ ) que celle des points de la figure de gauche.



### Exemple 2.2

À partir des données des 5 points suivants, quelle ordonnée peut-on extrapoler pour le point d'abscisse  $x = 6$  ?

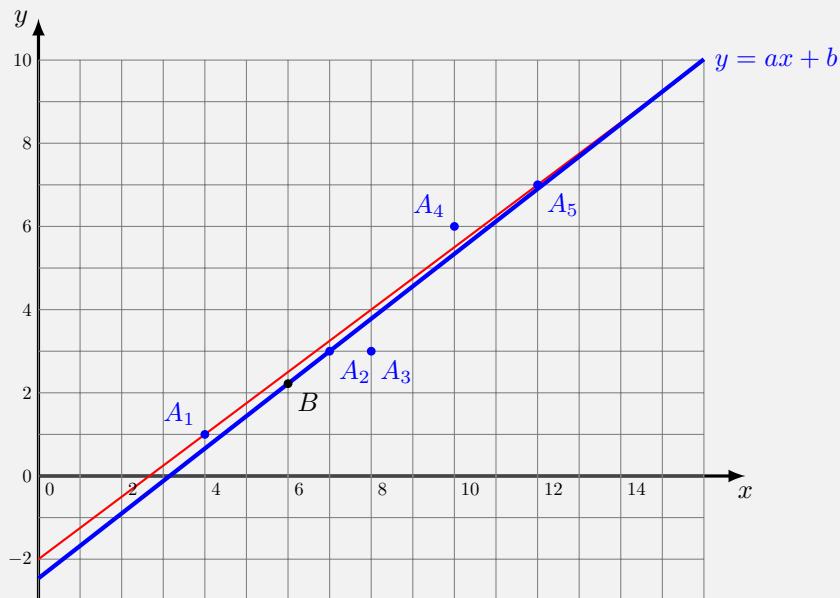
$$A_1 = (4, 1), \quad A_2 = (7, 3), \quad A_3 = (8, 3), \quad A_4 = (10, 6), \quad A_5 = (12, 7).$$



Ces 5 points sont à peu près alignés. On calcule la meilleure droite de régression linéaire par

la descente de gradient. Cela revient par exemple à minimiser la fonction  $E(a, b)$  présentée ci-dessus, mais actualisée avec nos données. On fixe un pas  $\delta = 0.001$ . On ne choisit pas le point initial  $(a_0, b_0)$  au hasard. Plus on part d'un point proche de la solution, plus la suite convergera rapidement. On trace la droite qui passe par le premier point  $A_1$  et le dernier point  $A_5$ . Cette droite a pour équation  $y = \frac{3}{4}x - 2$  et est déjà une droite qui approche assez bien les 5 points. Prenons cette droite comme point de départ, c'est-à-dire posons  $(a_0, b_0) = (\frac{3}{4}, -2)$ . La descente de gradient conduit au bout de 1000 itérations à  $a \simeq 0.78$  et  $b \simeq -2.46$ , pour l'équation de la droite de régression linéaire.

Sur le dessin ci-dessous sont tracées la droite initiale qui passe par les points  $A_1$  et  $A_5$  et la droite de régression linéaire.



N'oublions pas de répondre à la question initiale. Selon notre modèle linéaire, pour  $x = 6$ , on doit avoir  $y = ax + b \simeq 2.22$  (le point  $B$  de la figure ci-dessus).

Remarque : il existe une formule directe pour calculer exactement les coefficients  $a$  et  $b$  de la droite de régression linéaire, mais ce n'est pas l'esprit de ce cours.

## VIII-

## Descente de gradient stochastique

La descente de gradient stochastique (abrégée en *sgd*) est une façon d'optimiser les calculs de la descente de gradient pour une fonction d'erreur associée à une grande série de données. Au lieu de calculer un gradient (compliqué) et un nouveau point pour l'ensemble des données, on calcule un gradient (simple) et un nouveau point par donnée, il faut répéter ce processus pour chaque donnée.

## 1. Petits pas à petits pas

Revenons à l'objectif visé par la régression linéaire.

On considère des données  $(X_i, y_i)$ ,  $i = 1, \dots, N$  où  $X_i \in \mathbb{R}^\ell$  et  $y_i \in \mathbb{R}$ . Ces données proviennent d'observations ou d'expérimentations.

Il s'agit de trouver une fonction  $F : \mathbb{R}^\ell \rightarrow \mathbb{R}$  qui modélise au mieux ces données, c'est-à-dire telle que

$$F(X_i) \simeq y_i.$$

Pour l'**entrée**  $X_i$ , la valeur  $y_i$  est la **sortie attendue**, alors que  $F(X_i)$  est la **sortie produite** par notre modèle.

Pour mesurer la pertinence de la fonction  $F$ , on introduit la fonction d'**erreur totale** qui mesure l'écart entre la sortie attendue et la sortie produite :

$$E = \sum_{i=1}^N E_i = \sum_{i=1}^N (y_i - F(X_i))^2.$$

Cette erreur totale est une somme d'**erreurs locales** :

$$E_i = (y_i - F(X_i))^2.$$

Le but du problème est de déterminer la fonction  $F$  qui minimise l'erreur  $E$ . Par exemple, dans le cas de la régression linéaire, il fallait trouver les paramètres  $a$  et  $b$  pour définir  $F(x) = ax + b$ , ou bien, pour deux variables, les paramètres  $a$ ,  $b$ ,  $c$  pour définir  $F(x, y) = ax + by + c$ .

Considérons une fonction erreur  $E : \mathbb{R}^n \rightarrow \mathbb{R}$  qui dépend de  $n$  paramètres  $a_1, \dots, a_n$  (qui définissent l'expression de la fonction  $F$ ).

**Descente de gradient classique.** Pour minimiser l'erreur et déterminer les meilleurs paramètres, on peut appliquer la méthode du gradient classique.

On part d'un point  $P_0 = (a_1, \dots, a_n) \in \mathbb{R}^n$ , puis on applique la formule de récurrence :

$$\overrightarrow{P_{k+1}} = \overrightarrow{P_k} - \delta \overrightarrow{\text{grad}} E(P_k).$$

Pour appliquer cette formule, il faut calculer des gradients  $\overrightarrow{\text{grad}} E(P_k)$ , or

$$\overrightarrow{\text{grad}} E(P_k) = \sum_{i=1}^N \overrightarrow{\text{grad}} E_i(P_k).$$

Il faut donc calculer une somme de  $N$  termes à chaque itération, ce qui pose des problèmes d'efficacité pour de grandes valeurs de  $N$ .

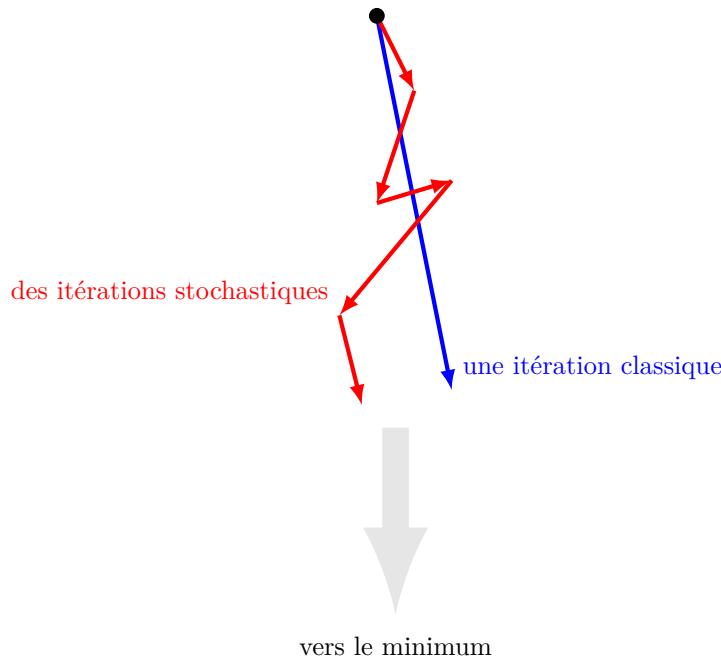
### Descente de gradient stochastique.

Pour diminuer la quantité de calculs, l'idée est de considérer à chaque itération un seul gradient  $E_i$  à la place de  $E$ . C'est-à-dire :

$$\overrightarrow{P_{k+1}} = \overrightarrow{P_k} - \delta \overrightarrow{\text{grad}} E_i(P_k)$$

pour une seule erreur  $E_i$  (correspondant à la donnée numéro  $i$ ). L'itération suivante se basera sur l'erreur  $E_{i+1}$ .

Quel est l'intérêt de cette méthode ? Dans la méthode de gradient classique, on calcule à chaque itération un « gros » gradient (associé à la totalité des  $N$  données) qui nous rapproche d'un grand pas vers le minimum. Ici on calcule  $N$  « petits » gradients qui nous rapprochent du minimum.



Voici les premières itérations de cet algorithme.

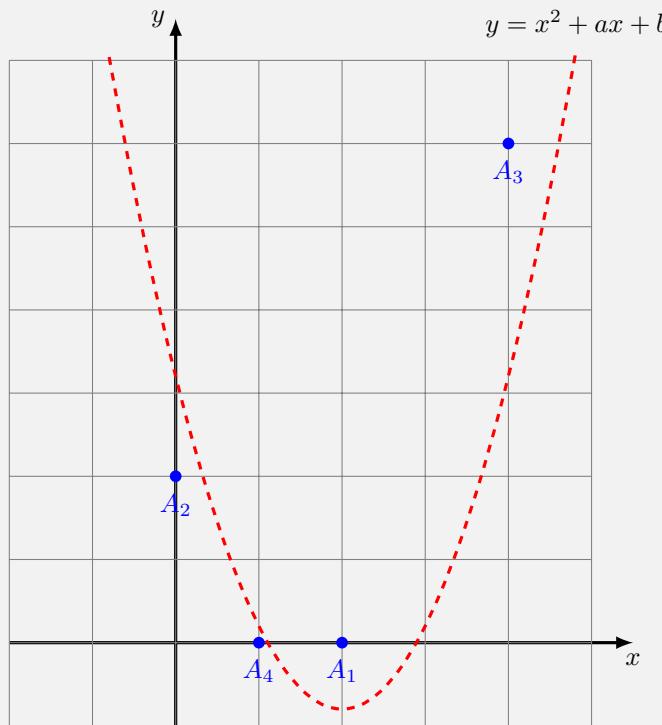
- On part d'un point  $P_0$ .
- On calcule  $P_1 = P_0 - \delta \overrightarrow{\text{grad}} E_1(P_0)$ . C'est la formule du gradient, mais seulement pour l'erreur locale  $E_1$  (juste à partir de la première donnée  $(X_1, y_1)$ ).
- On calcule  $P_2 = P_1 - \delta \overrightarrow{\text{grad}} E_2(P_1)$ . C'est la formule du gradient, mais seulement pour l'erreur locale  $E_2$ .
- On itère encore et encore.
- On calcule  $P_N = P_{N-1} - \delta \overrightarrow{\text{grad}} E_N(P_{N-1})$ . C'est la formule du gradient, mais seulement pour l'erreur locale  $E_N$ . À ce stade de l'algorithme, nous avons tenu compte de toutes les données.
- On calcule  $P_{N+1} = P_N - \delta \overrightarrow{\text{grad}} E_1(P_N)$ . On recommence pour  $P_N$  et l'erreur locale  $E_1$ .
- Etc. On s'arrête au bout d'un nombre d'étapes fixé à l'avance ou lorsque l'on est suffisamment proche du minimum.

### Exemple 1.1

On considère les quatre points :

$$A_1 = (2, 0), \quad A_2 = (0, 2), \quad A_3 = (4, 6), \quad A_4 = (1, 0).$$

Comme ces points ne sont clairement pas alignés, on cherche un modèle pour les placer au mieux sur une parabole.



On va ici chercher des coefficients  $a$  et  $b$  tels que les points soient proches de la parabole d'équation  $y = x^2 + ax + b$ . On note donc  $F(x) = x^2 + ax + b$  et on souhaite  $F(x_i) \simeq y_i$  pour les points  $A_i = (x_i, y_i)$ ,  $i = 1, \dots, 4$ .

Les fonctions d'erreurs locales sont :

$$E_i(a, b) = (y_i - (x_i^2 + ax_i + b))^2.$$

La fonction d'erreur globale est :

$$E(a, b) = E_1(a, b) + E_2(a, b) + E_3(a, b) + E_4(a, b).$$

Voici les premières itérations pour chacune des méthodes, la descente de gradient classique (à gauche), la descente de gradient stochastique (à droite) toutes les deux en partant du point  $(a_0, b_0) = (1, 1)$  et avec  $\delta = 0.01$ .

Descente de gradient		Descente de gradient stochastique		
$k$	$(a_k, b_k)$	$E(a_k, b_k)$	$(a'_k, b'_k)$	$E(a'_k, b'_k)$
0	(1, 1)	284	(1, 1)	284
1	(-0.54, 0.52)	84.87	(0.72, 0.86)	236.43
2	(-1.36, 0.29)	28.68	(0.72, 0.88)	237.41
			(-0.38, 0.60)	100.74
			(-0.40, 0.58)	97.917
			(-0.55, 0.50)	83.245
			(-0.55, 0.53)	83.913
			(-1.22, 0.37)	36.48
			(-1.22, 0.36)	36.29

$k$	$(a_k, b_k)$	$E(a_k, b_k)$
0	(1, 1)	284
1	(-0.54, 0.52)	84.87
2	(-1.36, 0.29)	28.68

$k$	$(a'_k, b'_k)$	$E(a'_k, b'_k)$
0	(1, 1)	284
1	(0.72, 0.86)	236.43
2	(0.72, 0.88)	237.41
3	(-0.38, 0.60)	100.74
4	(-0.40, 0.58)	97.917
5	(-0.55, 0.50)	83.245
6	(-0.55, 0.53)	83.913
7	(-1.22, 0.37)	36.48
8	(-1.22, 0.36)	36.29

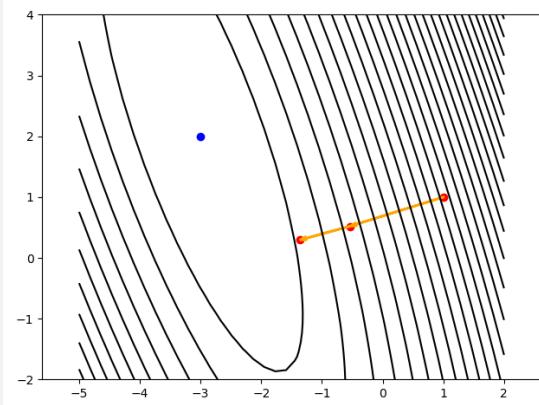
Au bout de 200 itérations la descente de gradient classique conduit à  $(a_{200}, b_{200}) \simeq (-2.9981, 1.9948)$  (chaque donnée a été utilisée 200 fois).

Cela correspond à 800 itérations de la descente de gradient stochastique (chacune des 4 données a été utilisée 200 fois) cela conduit à  $(a'_{800}, b'_{800}) \simeq (-2.9984, 1.9954)$ . La limite

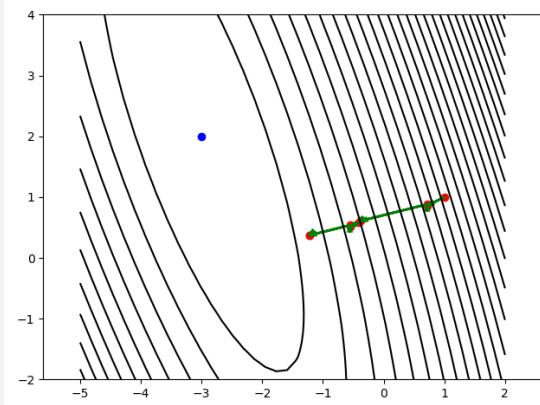
cherchée étant  $(a, b) = (-3, 2)$ , avec  $E(a, b) = 0$ , les deux méthodes convergent à la même vitesse. Chaque calcul de gradient de la méthode stochastique est très simple, mais il faut plus d'itérations.

Voici les points des premières itérations correspondant au tableau ci-dessus.

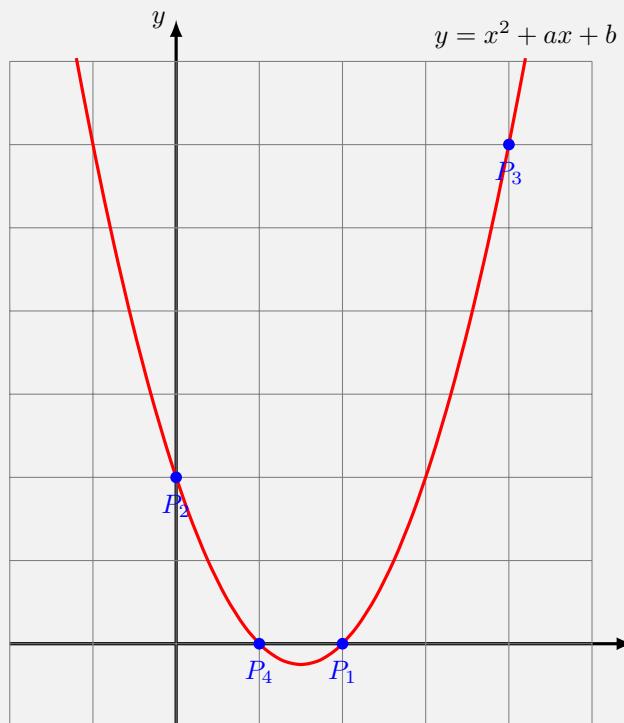
**Descente de gradient classique**



**Descente de gradient stochastique**



Conclusion : sur cet exemple les points sont exactement sur la parabole d'équation  $y = x^2 + ax + b$  avec  $a = -3$  et  $b = 2$ . Bien sûr cette méthode est encore plus intéressante lorsqu'il s'agit de trouver une parabole qui ne contient pas l'ensemble des points donnés.



Terminons par des remarques plus techniques. Tout d'abord, la formule précise de la descente de gradient stochastique est :

$$P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} E_{(k\%N)+1}(P_k)$$

où  $k\%N$  est «  $k$  modulo  $N$  ».

La descente de gradient stochastique est une méthode qui peut être plus efficace :

- Tout d'abord elle n'utilise qu'une donnée à la fois et évite ainsi les problèmes de mémoire de la descente classique pour laquelle il faut manipuler toutes les données à chaque itération.

- Toujours dans le cas où l'on a beaucoup de données, la descente de gradient stochastique peut converger en deux ou trois passages sur l'ensemble des données, alors que la descente classique nécessite toujours plusieurs itérations (voir la section 3 plus loin).
- Avec la méthode stochastique, on calcule des gradients en des points qui sont plus proches du minimum. Attention cependant, certains petits pas peuvent aller dans la mauvaise direction.
- Le caractère aléatoire de ces petits pas est parfois un avantage, par exemple pour s'échapper d'un point-selle.

## 2. Différentes fonctions d'erreurs

Il existe différentes formules pour calculer l'erreur entre la sortie attendue  $y_i$  et la sortie produite  $F(x_i)$ .

On considère une série de valeurs  $y_i$ ,  $i = 1, \dots, N$  (fournie par observations ou expérimentations) qui sont approchées par des valeurs  $F(x_i)$  produites par une formule issue d'un réseau de neurones par exemple. Le but est d'obtenir  $F(x_i)$  le plus proche possible de  $y_i$ , pour tout  $i = 1, \dots, N$ . Pour savoir si l'objectif est atteint, on mesure l'écart entre ces valeurs.

### Erreur quadratique moyenne.

$$E = \frac{1}{N} \sum_{i=1}^N (y_i - F(x_i))^2.$$

C'est la formule la plus classique (en anglais *minimal squared error* ou *mse*). Bien entendu  $E \geq 0$  quels que soient les  $F(x_i)$  et  $E = 0$  si et seulement si  $y_i = F(x_i)$  pour tous les  $i = 1, \dots, N$ . C'est presque la formule que l'on a utilisée pour la régression linéaire (il n'y avait pas le facteur  $\frac{1}{N}$ ).

### Erreur absolue moyenne.

$$E = \frac{1}{N} \sum_{i=1}^N |y_i - F(x_i)|.$$

C'est une formule plus naturelle, mais moins agréable à manipuler à cause de la valeur absolue.

Noter que pour ces deux formules, l'erreur globale  $E$  est la moyenne d'erreurs locales  $E_i = (y_i - F(x_i))^2$  (ou bien  $E_i = |y_i - F(x_i)|$ ). Les erreurs locales sont indépendantes les unes des autres, ce qui est la base de la descente de gradient stochastique. (Une formule d'erreur du type  $E = y_1 y_2 - F(x_1)F(x_2)$  ne permettrait pas la descente stochastique.)

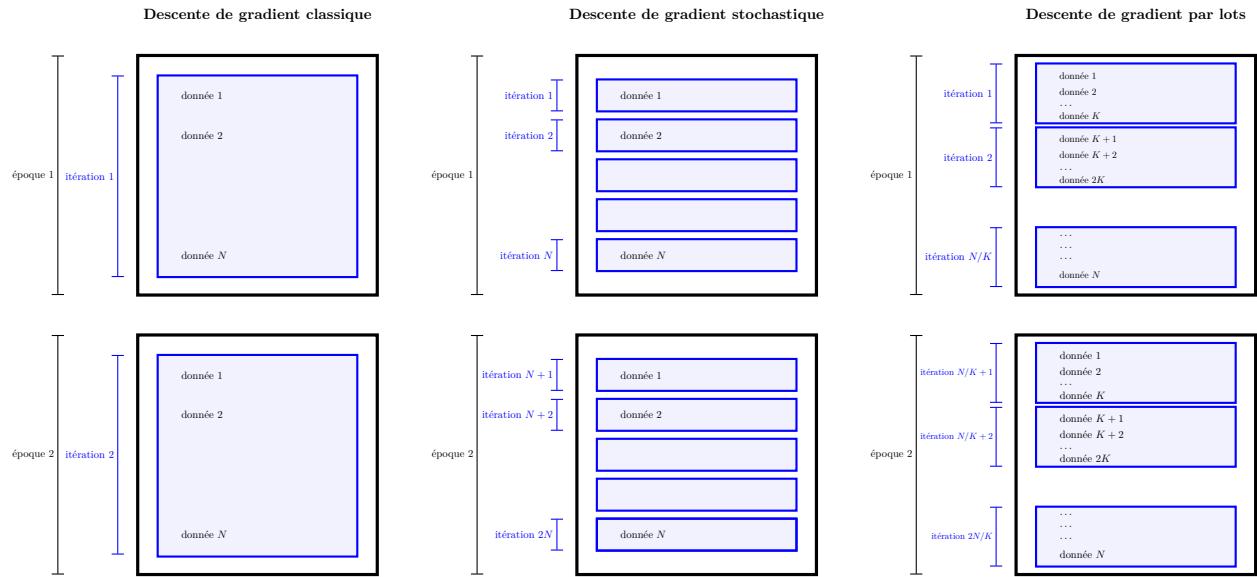
Il existe d'autres formules d'erreur, en particulier si la sortie attendue est du type 0 ou 1 ou bien si la sortie produite est une probabilité  $0 \leq p \leq 1$ .

## 3. Descente par lots

Il existe une méthode intermédiaire entre la descente de gradient classique (qui tient compte de toutes les données à chaque itération) et la descente de gradient stochastique (qui n'utilise qu'une seule donnée à chaque itération).

La descente de gradient par **lots** (ou **mini-lots**, *mini-batch*) est une méthode intermédiaire : on divise les données par paquets de taille  $K$ . Pour chaque paquet (appelé « lot »), on calcule un gradient et on effectue une itération.

Au bout de  $N/K$  itérations, on a parcouru tout le jeu de données : cela s'appelle une **époque**.



La formule est donc

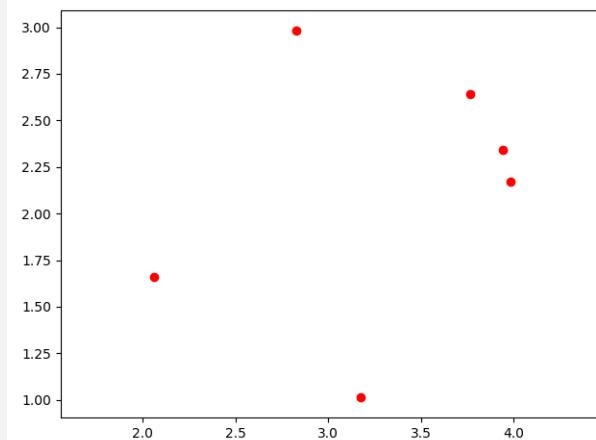
$$P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} (E_{j_0+1} + E_{j_0+2} + \dots + E_{j_0+K})(P_k).$$

Pour  $P_{k+2}$ , on repart de  $P_{k+1}$  et on utilise le gradient de la fonction  $E_{j_0+K+1} + E_{j_0+K+2} + \dots + E_{j_0+2K}$ .

- Pour  $K = 1$ , c'est exactement la descente de gradient stochastique. Pour  $K = N$ , c'est la descente de gradient classique.
- Cette méthode combine le meilleur des deux mondes : la taille des données utilisées à chaque itération peut être adaptée à la mémoire et le fait de travailler par lots évite les pas erratiques de la descente stochastique pure.
- On peut par exemple choisir  $2 \leq K \leq 32$  et profiter du calcul parallèle en calculant  $\overrightarrow{\text{grad}} (E_1 + \dots + E_K)$ , par le calcul de chacun des  $\overrightarrow{\text{grad}} E_i$  sur  $K$  processeurs, puis en additionnant les résultats.
- Il est d'usage de mélanger au hasard les données  $(X_i, y_i)$  avant chaque époque.

### Exemple 3.1

Voyons un exemple d'interpolation circulaire. Les 6 points ci-dessous sont sur un cercle. Comment déterminer son centre et son rayon ?



Pour des points  $(x_i, y_i)$ ,  $i = 1, \dots, N$ , on mesure la distance globale par rapport au cercle  $\mathcal{C}$

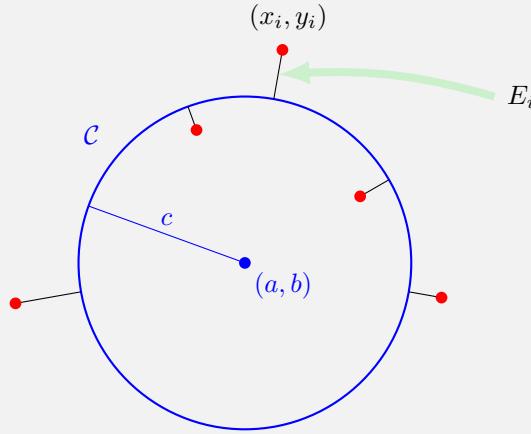
de centre  $(a, b)$  et de rayon  $c$  par la formule d'erreur :

$$E(a, b, c) = \sum_{i=1}^N E_i(a, b, c)$$

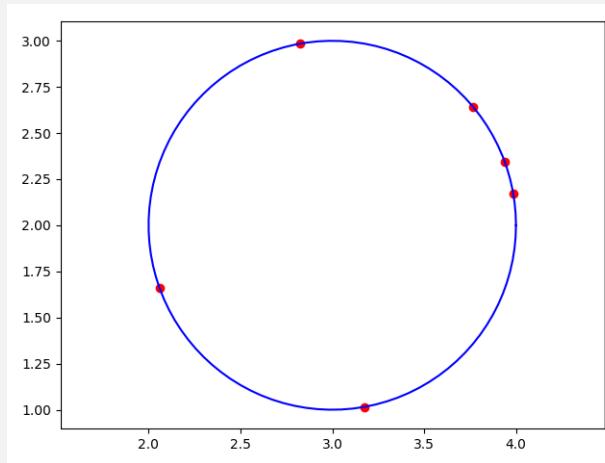
où

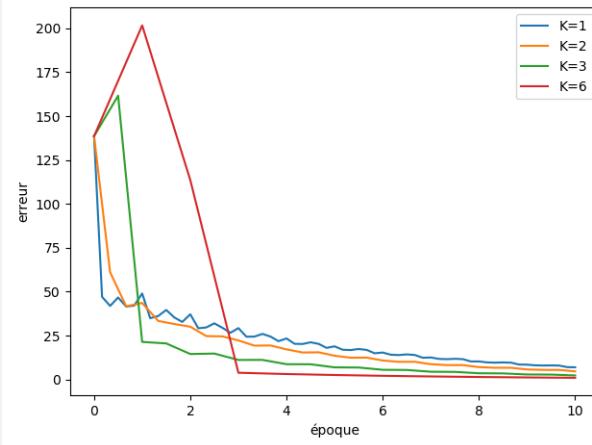
$$E_i(a, b, c) = ((x_i - a)^2 + (y_i - b)^2 - c^2)^2.$$

En effet,  $E_i$  mesure en quelque sorte la distance entre le point  $(x_i, y_i)$  et le cercle  $\mathcal{C}$  de centre  $(a, b)$  et de rayon  $c$ . Donc  $E_i = 0$  si et seulement si  $(x_i, y_i) \in \mathcal{C}$ , sinon  $E_i > 0$ .



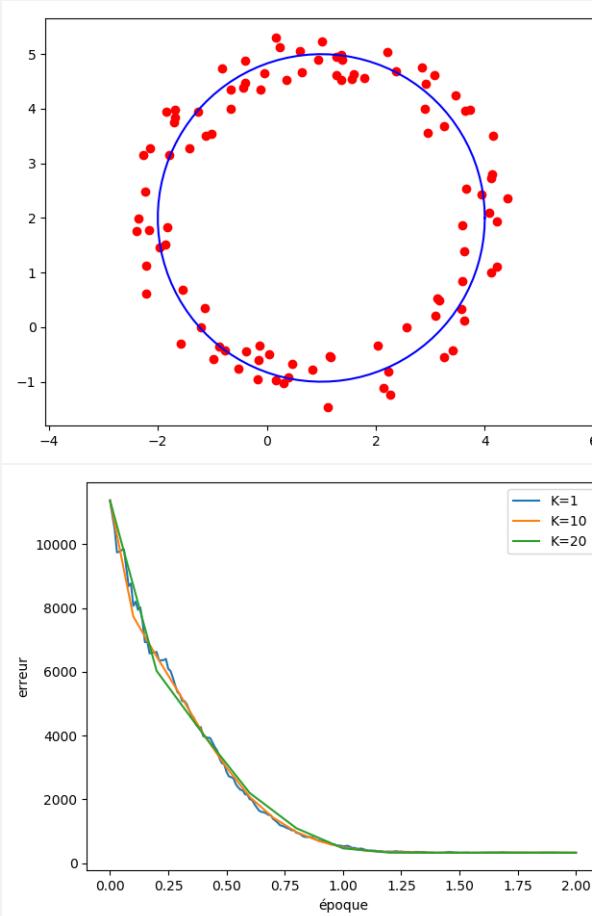
Dans notre exemple, les  $N = 6$  points sont exactement situés sur le cercle de centre  $(a, b)$  et de rayon  $c$ . En appliquant la descente de gradient par lots avec  $\delta = 0.01$  et  $(a_0, b_0, c_0) = (1, 1, 2)$ , on trouve  $(a, b) = (3, 2)$  et  $c = 1$ . Ci-dessous, à droite, nous avons représenté les valeurs de l'erreur totale  $E$  (qui tend vers 0) en fonction du nombre d'époques et ceci pour différentes tailles du lot :  $K = 1$  (descente stochastique),  $K = 2$ ,  $K = 3$  et  $K = 6$  (descente classique).





On remarque qu'au bout de 10 époques la valeur de l'erreur est à peu près la même quelle que soit la taille  $K$  de l'échantillon. Par contre, l'évolution au départ est différente. Par exemple pour  $K = 1$ , l'erreur fluctue à la hausse ou à la baisse à chaque itération.

Cette méthode présente bien sûr davantage d'intérêt quand les points ne sont pas exactement sur un cercle. Il s'agit alors de trouver le meilleur cercle qui convient, c'est-à-dire de trouver le minimum (cette fois non nul) de  $E$ . Voici un exemple de  $N = 100$  points tirés au hasard autour du cercle de centre  $(a, b) = (1, 2)$  et de rayon  $c = 3$ . La descente de gradient est appliquée avec  $\delta = 0.001$  et  $(a_0, b_0, c_0) = (1, 1, 1)$  pour des lots de différentes tailles  $K = 1$ ,  $K = 10$  et  $K = 20$ . On remarque que deux époques suffisent pour avoir convergence et que plus la taille  $K$  de l'échantillon est grande plus la convergence est régulière vers le minimum.

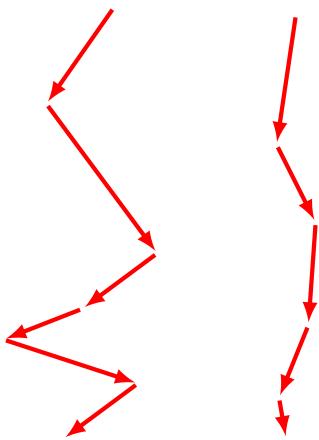


## IX- Accélérations

Le choix du pas  $\delta$  n'est pas la seule amélioration possible de la méthode du gradient, nous allons voir comment la modifier à l'aide du « moment ». Commençons par revenir à l'analogie de la descente du gradient classique qui correspond à une goutte d'eau qui descend une montagne : la goutte emprunte le chemin qui suit la courbe de plus grande pente, quitte à serpenter et osciller lors de la descente. Imaginons que l'on lance maintenant une balle assez lourde du haut de la même montagne. Cette balle va suivre, comme la goutte d'eau, le chemin de la plus forte pente, mais une fois lancée elle va acquérir de l'inertie, appelée **moment**, qui va atténuer ses changements de direction. Ainsi la balle ne s'embarrasse pas des petits aléas du terrain et dévale la pente plus rapidement que la goutte d'eau.

Nos petits aléas de terrain à nous viennent du fait que l'on ne calcule pas exactement le gradient de la fonction d'erreur en utilisant tout le jeu de données à chaque fois, mais seulement un échantillon. Cela peut conduire à certains gradients mal orientés. L'inertie de la balle est en quelque sorte la mémoire de la trajectoire passée qui corrige les mouvements erratiques.

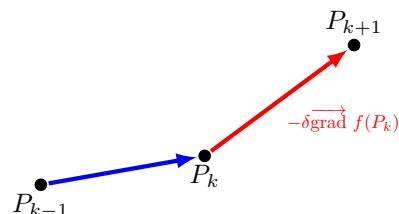
Sur la figure de gauche, la descente classique (la goutte d'eau), sur la figure de droite, la descente de gradient avec le moment (la balle).



### 1. Moment

Rappelons la formule de la descente de gradient classique :

$$P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} f(P_k).$$



Considérons nos points comme une particule qui voyage au cours du temps. Alors le vecteur  $\overrightarrow{P_{k-1}P_k}$  correspond à la vitesse de cette particule et est appelé le **moment** au point  $P_k$ .

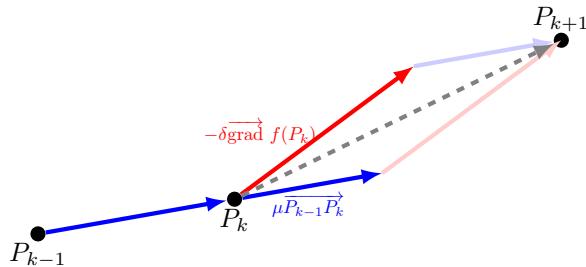
La formule de la descente de gradient avec moment est :

$$P_{k+1} = P_k + \mu \overrightarrow{P_{k-1}P_k} - \delta \overrightarrow{\text{grad}} f(P_k).$$

où  $\mu, \delta \in \mathbb{R}$ . Cette formule peut être définie pour  $k = 0$  si on suppose que la particule est immobile au départ, c'est-à-dire en posant  $\overrightarrow{P_{-1}P_0} = \vec{0}$ .

On peut prendre par exemple  $\mu \in [0.5, 0.9]$  et  $\delta = 0.01$ .

Schématiquement, au point  $P_k$  nous avons deux vecteurs : un qui provient du moment  $\mu \overrightarrow{P_{k-1}P_k}$  (la mémoire du passé) et un qui provient du gradient  $-\delta \overrightarrow{\text{grad}} f(P_k)$  (qui projette vers l'avenir). La somme permet de calculer le point suivant  $P_{k+1}$ .



## 2. Nesterov

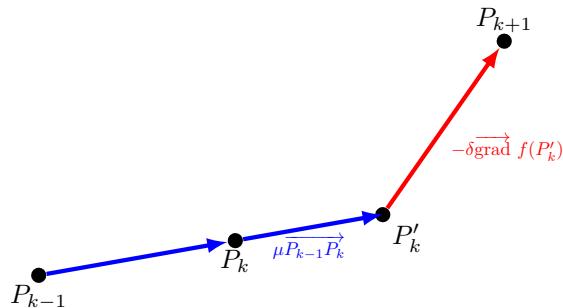
Dans la méthode précédente, le moment et le gradient sont calculés au même point  $P_k$ . La méthode de Nesterov est une variante de cette méthode. Elle consiste à appliquer d'abord le moment, pour obtenir un point  $P'_k$ , puis de calculer le gradient en ce point (et non en  $P_k$ ).

La formule est donc

$$P_{k+1} = P_k + \mu \overrightarrow{P_{k-1}P_k} - \delta \overrightarrow{\text{grad}} f(P_k + \mu \overrightarrow{P_{k-1}P_k}).$$

Autrement dit, si on note  $P'_k$  le point  $P_k + \mu \overrightarrow{P_{k-1}P_k}$  alors

$$P_{k+1} = P'_k - \delta \overrightarrow{\text{grad}} f(P'_k).$$



C'est un petit avantage par rapport à la méthode du moment puisqu'on calcule le gradient au point  $P'_k$  qui est censé être plus près de la solution  $P_{\min}$  que  $P_k$ .

## 3. Vocabulaire

Terminons par un petit résumé du vocabulaire avec sa traduction en anglais :

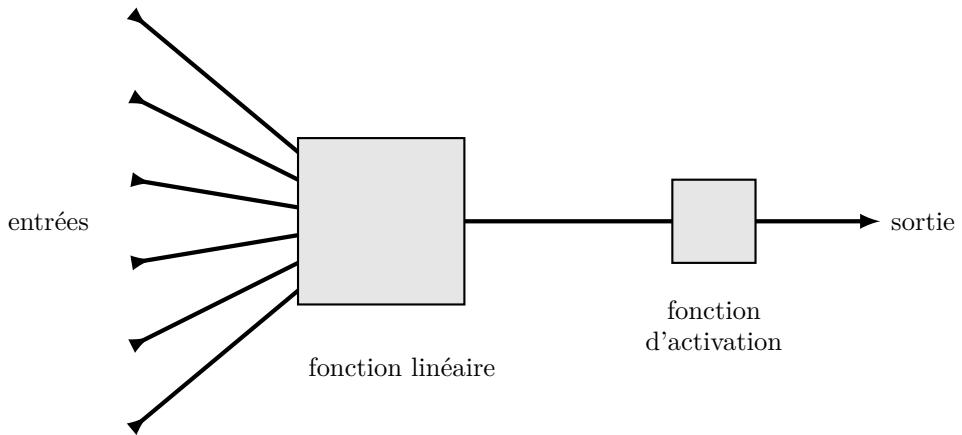
- descente de gradient (classique), *(batch) gradient descent*,
- descente de gradient stochastique, *sgd* pour *stochastic gradient descent*,
- descente de gradient par lots, *mini-batch gradient descent*,
- pas  $\delta$ , *learning rate*,
- erreur quadratique moyenne, *mse* pour *minimal squared error*,
- moment, *momentum*,
- époque, *epoch*.

## Réseaux de neurones

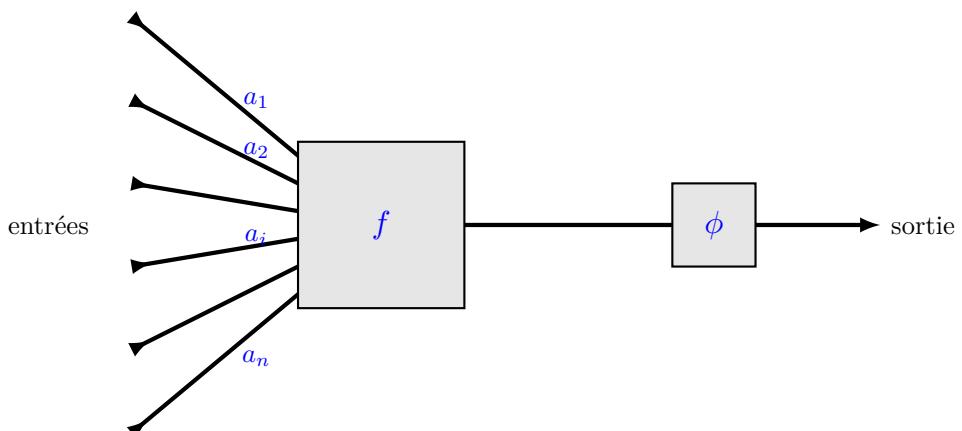
### I- Perceptron

#### 1. Perceptron linéaire

Le principe du perceptron linéaire est de prendre des valeurs en entrées, de faire un calcul simple et de renvoyer une valeur en sortie. Les calculs dépendent de paramètres propres à chaque perceptron.



Le calcul effectué par un perceptron se décompose en deux phases : un calcul par une fonction linéaire  $f$ , suivi d'une fonction d'activation  $\phi$ .



Détaillons chaque phase.

- **Partie linéaire.** Le perceptron est d'abord muni de **poids**  $a_1, \dots, a_n$  qui déterminent une fonction linéaire

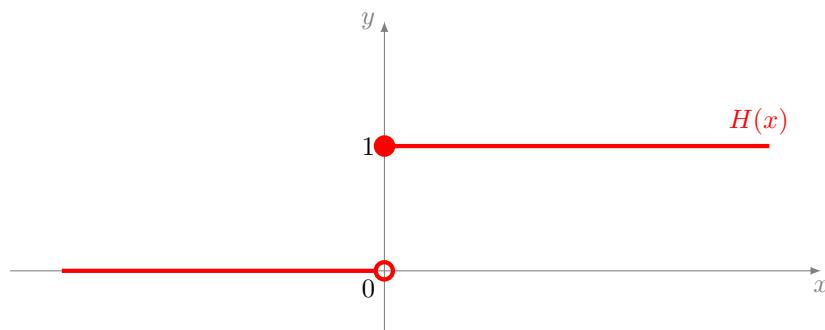
$$f(x_1, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n.$$

- **Fonction d'activation.** La valeur renvoyée par la fonction linéaire  $f$  est ensuite composée par une fonction d'activation  $\phi$ .

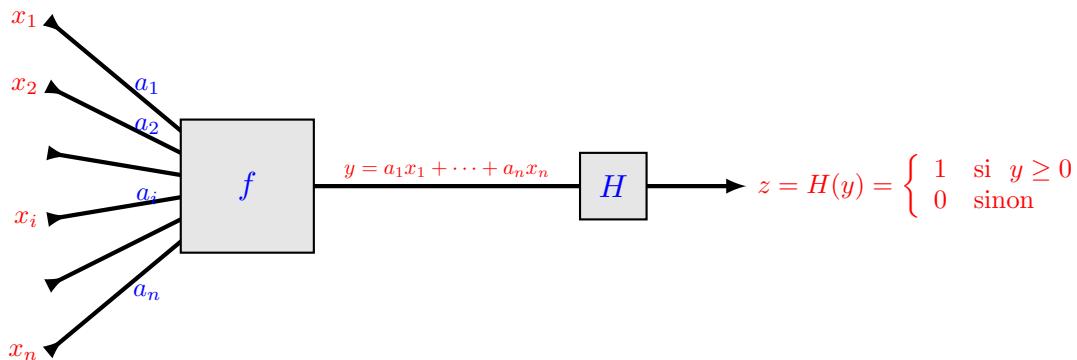
- **Sortie.** La valeur de sortie est donc  $\phi(a_1x_1 + a_2x_2 + \dots + a_nx_n)$ .

Dans ce chapitre, la fonction d'activation sera (presque) toujours la fonction marche de Heaviside :

$$\begin{cases} H(x) = 1 & \text{si } x \geq 0, \\ H(x) = 0 & \text{si } x < 0. \end{cases}$$



Voici ce que fait un perceptron linéaire de poids  $a_1, \dots, a_n$  et de fonction d'activation la fonction marche de Heaviside :

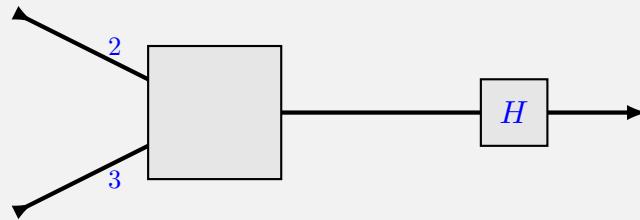


On peut donc définir ce qu'est un perceptron. Un **perceptron linéaire** à  $n$  variables et de fonction d'activation la fonction marche de Heaviside est la donnée de  $n$  coefficients réels  $a_1, \dots, a_n$  auxquels est associée la fonction  $F : \mathbb{R} \rightarrow \mathbb{R}$  définie par  $F = H \circ f$ , c'est-à-dire :

$$\begin{cases} F(x_1, \dots, x_n) = 1 & \text{si } a_1x_1 + a_2x_2 + \dots + a_nx_n \geq 0, \\ F(x_1, \dots, x_n) = 0 & \text{sinon.} \end{cases}$$

### Exemple 1.1

Voici un perceptron à deux entrées. Il est défini par les poids  $a = 2$  et  $b = 3$ .



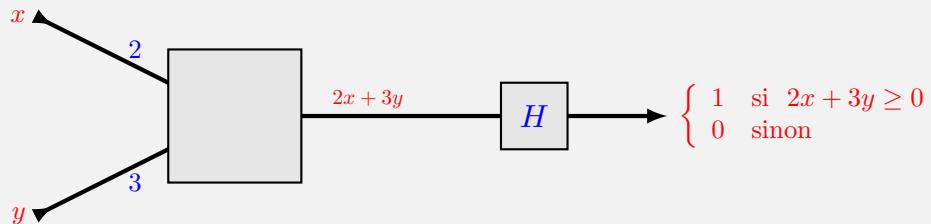
— **Formule.**

Notons  $x$  et  $y$  les deux réels en entrée. La fonction linéaire  $f$  est donc

$$f(x, y) = 2x + 3y.$$

La valeur en sortie est donc :

$$\begin{cases} F(x, y) = 1 & \text{si } 2x + 3y \geq 0 \\ F(x, y) = 0 & \text{sinon.} \end{cases}$$



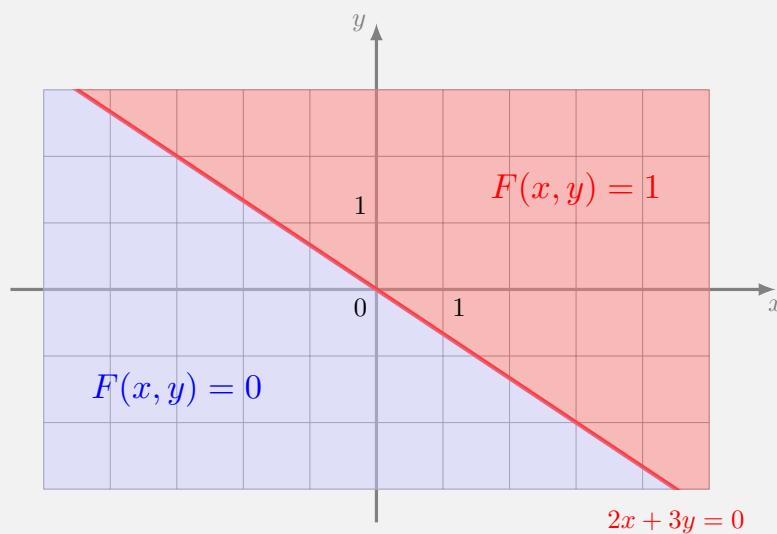
— **Évaluation.** Utilisons ce perceptron comme une fonction. Que renvoie le perceptron pour la valeur d'entrée  $(x, y) = (4, -1)$ ? On calcule  $f(x, y) = 2x + 3y = 5$ . Comme  $f(x, y) \geq 0$ , alors la valeur de sortie est donc  $F(x, y) = 1$ .

Recommençons avec  $(x, y) = (-3, 1)$ . Cette fois  $f(x, y) = -3 < 0$  donc  $F(x, y) = 0$ .

L'entrée  $(x, y) = (6, -4)$  est « à la limite » car  $f(x, y) = 0$  ( $0$  est l'abscisse critique pour la fonction marche de Heaviside). On a  $F(x, y) = 1$ .

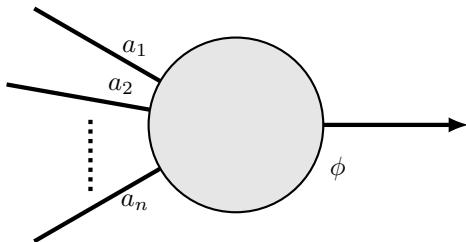
— **Valeurs de la fonction.**

La fonction  $F$  prend seulement deux valeurs :  $0$  ou  $1$ . La frontière correspond aux points  $(x, y)$  tels que  $f(x, y) = 0$ , c'est-à-dire à la droite  $2x + 3y = 0$ . Pour les points au-dessus de la droite (ou sur la droite) la fonction  $F$  prend la valeur  $1$ ; pour les points en-dessous de la droite, la fonction  $F$  vaut  $0$ .

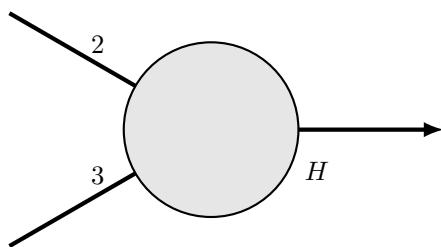


**Notation.**

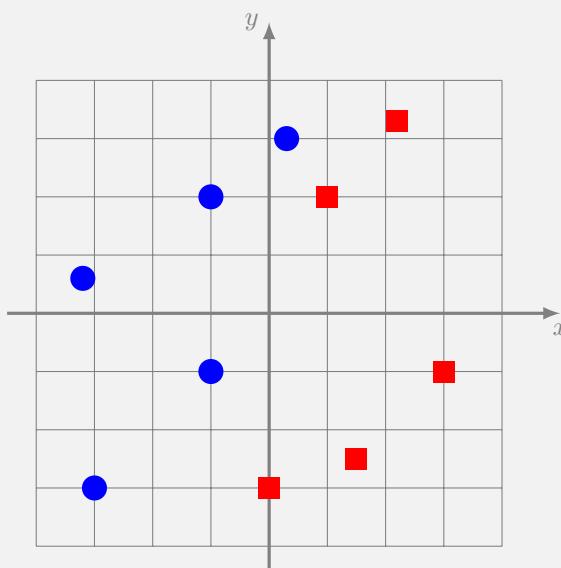
Nous représentons un perceptron par une forme plus condensée : sous la forme d'un **neurone**, avec des poids sur les arêtes d'entrées. Nous précisons en indice la fonction d'activation utilisée  $\phi$ . Si le contexte est clair cette mention est omise.



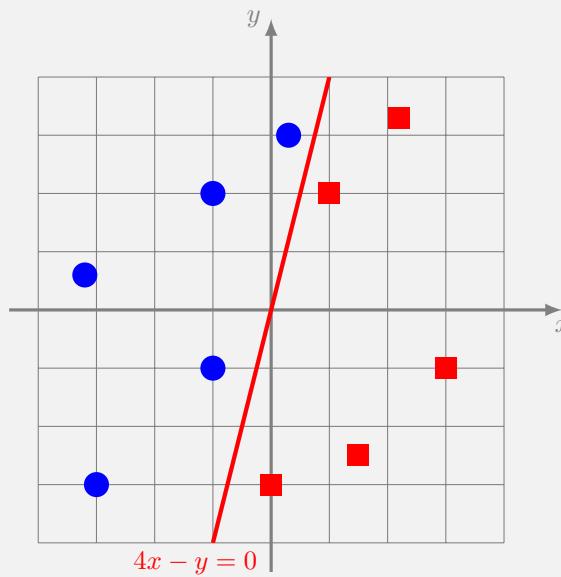
Voici le neurone à deux variables de l'exemple précédent.

**Exemple 1.2**

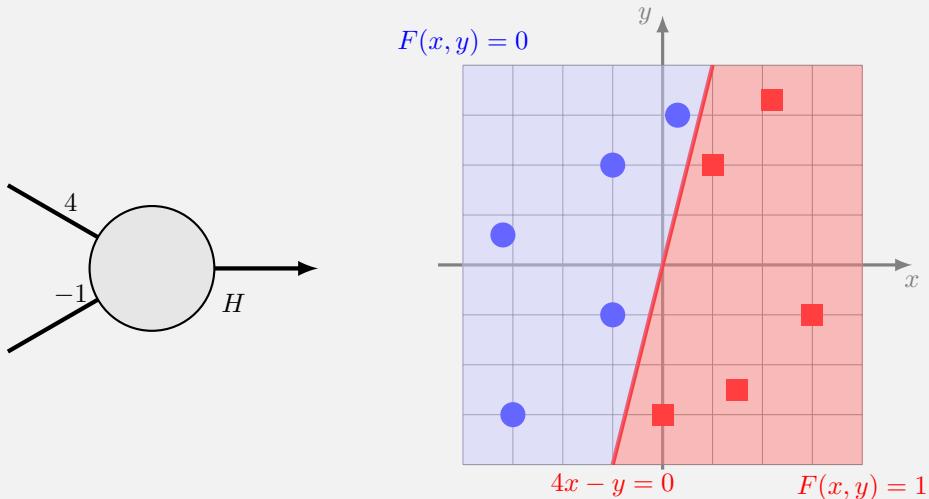
Voici deux catégories de points : des ronds bleus et des carrés rouges. Comment trouver un perceptron qui les sépare ?



Il s'agit donc de trouver les deux poids  $a$  et  $b$  d'un perceptron, dont la fonction associée  $F$  vérifie  $F(x, y) = 1$  pour les coordonnées des carrés et  $F(x, y) = 0$  pour les ronds.



Trouvons une droite qui les sépare. Par exemple, la droite d'équation  $4x - y = 0$  sépare les ronds des carrés. On définit donc le neurone avec les poids  $a = 4$  et  $b = -1$ . Si  $(x, y)$  sont les coordonnées d'un carré alors on a bien  $F(x, y) = 1$  et pour un rond  $F(x, y) = 0$ .



## 2. Biais – Perceptron affine

Pour l'instant notre perceptron à deux entrées sépare le plan en deux parties selon une droite passant par l'origine. Nous allons plus loin avec le **perceptron affine**.

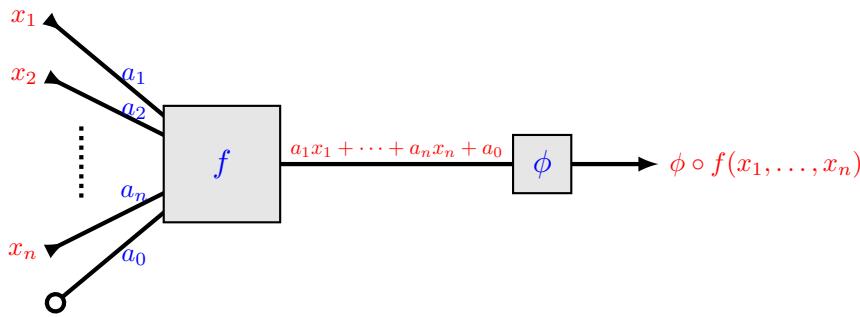
On modifie la définition du perceptron, sans changer le nombre d'entrées, mais en ajoutant un poids supplémentaire. Dans le cas de  $n$  entrées il y a donc  $n + 1$  **poids** :

- les **coefficients**  $a_1, \dots, a_n \in \mathbb{R}$ ,
- et le **biais**  $a_0 \in \mathbb{R}$ ,

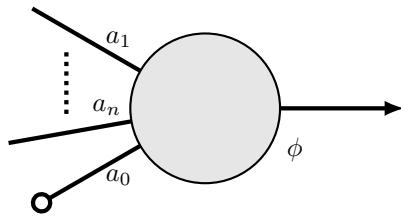
qui définissent la fonction affine :

$$f(x_1, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n + a_0.$$

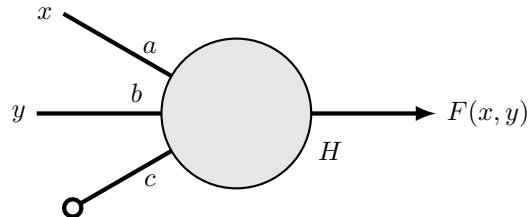
Comme auparavant, ce calcul est suivi de la fonction d'activation.



On représente le neurone avec une nouvelle arête, pondérée par le biais. L'arête est terminée par un rond pour signifier que cela ne correspond pas à une entrée.



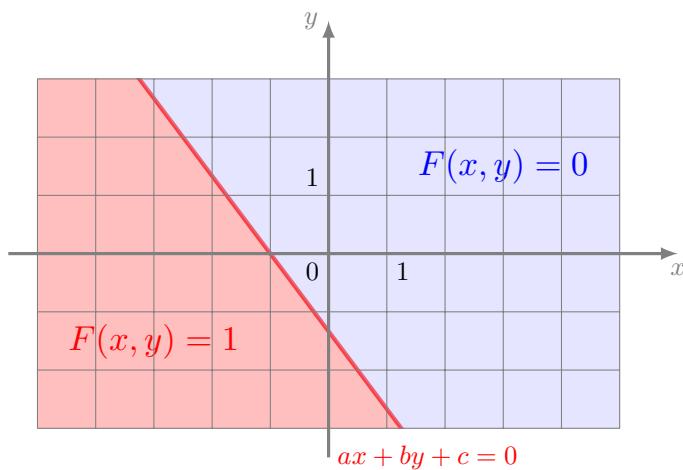
Dans le cas de deux entrées, les poids sont trois réels  $a, b$  (les coefficients) et  $c$  (le biais).

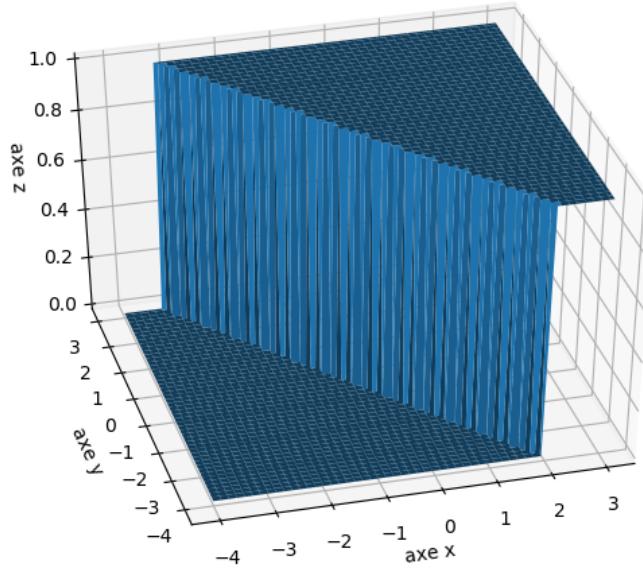


La fonction  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  est la fonction affine  $f(x, y) = ax + by + c$ . Si la fonction d'activation est la fonction marche de Heaviside, alors le perceptron affine définit une fonction  $F : \mathbb{R}^2 \rightarrow \mathbb{R}$  par la formule :

$$\begin{cases} F(x, y) = 1 & \text{si } ax + by + c \geq 0 \\ F(x, y) = 0 & \text{sinon.} \end{cases}$$

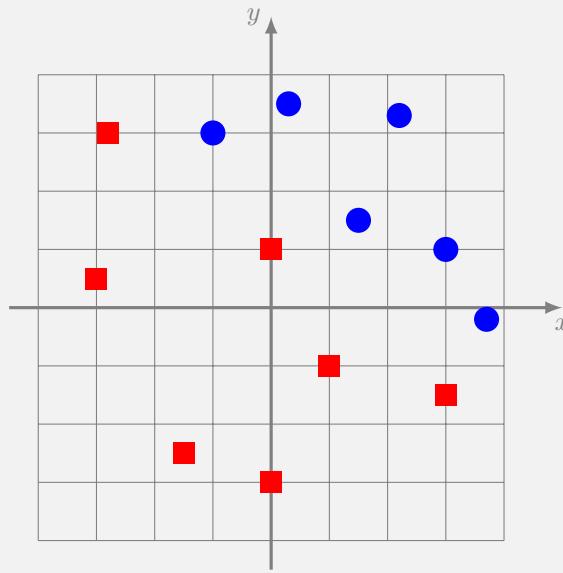
Un perceptron affine à deux entrées sépare donc le plan en deux parties, la frontière étant la droite d'équation  $ax + by + c = 0$ . D'un côté de cette droite la fonction  $F$  vaut 1, de l'autre elle vaut 0.





### Exemple 2.1

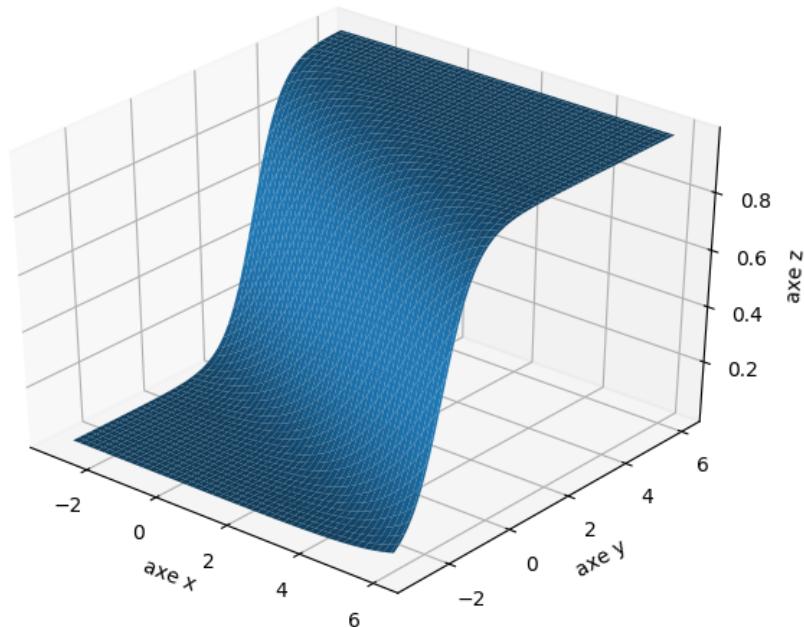
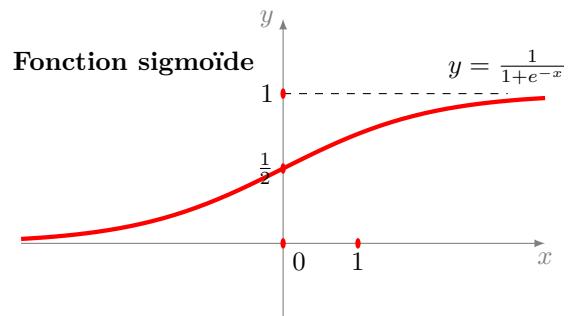
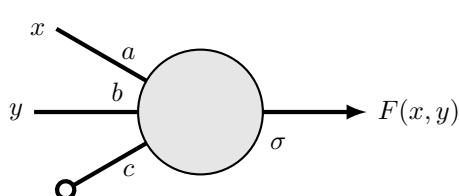
Trouver un perceptron qui distingue les carrés des ronds.



Il n'est pas toujours possible de répondre à une question seulement par « oui » ou « non ». Pour y remédier, on peut changer de fonction d'activation en utilisant par exemple la fonction sigmoïde  $\sigma$ . Dans ce cas, à chaque point du plan, on associe, non plus 0 ou 1, mais une valeur entre 0 et 1.

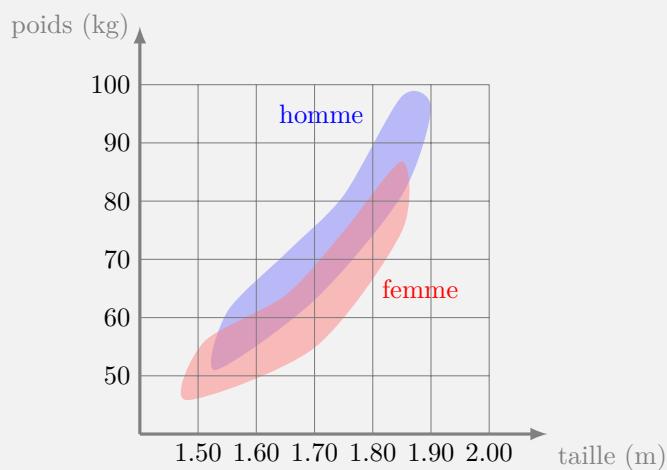
Ce nombre peut correspondre à un degré de certitude. Par exemple avec une question « Est-ce que cette photo est un chat ? », si la sortie vaut 0.8, cela signifie « c'est bien un chat avec 80% de certitude ». Si la sortie vaut 0.1 alors ce n'est probablement pas un chat.

Voici un neurone à deux entrées muni de la fonction d'activation sigmoïde définie par  $\sigma(x) = \frac{1}{1+e^{-x}}$ .



### Exemple 2.2

Voici les données (fictives) de la répartition des hommes et des femmes selon leur taille et leur poids.

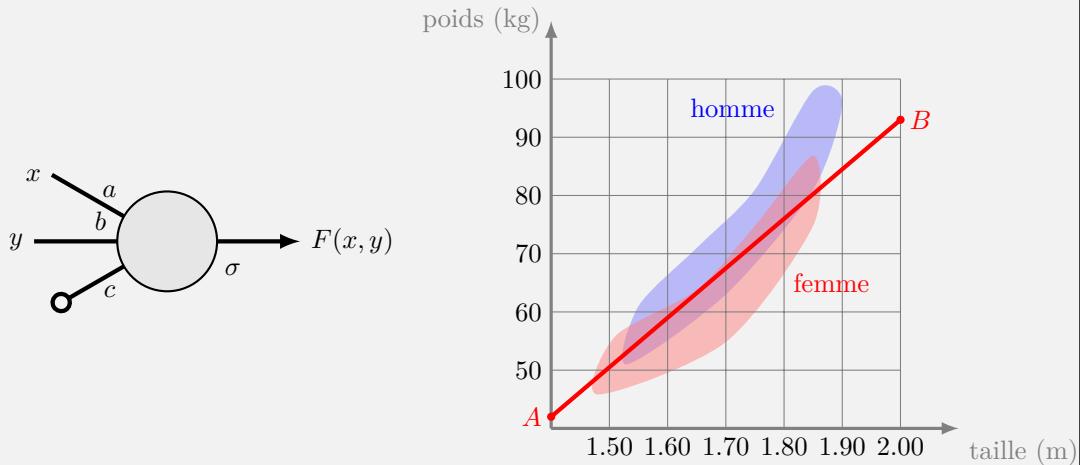


Problème : la taille et le poids d'une personne étant donnés, trouver un perceptron qui réponde à la question « Cette personne est-elle un homme ou une femme ? ».

Au vu de la superposition des zones, il n'est pas possible de répondre avec certitude. On construit donc un perceptron selon les idées suivantes :

- on trace une droite qui sépare au mieux les hommes des femmes, par exemple ici la droite qui passe par les points  $A(1.40, 42)$  et  $B(2.00, 93)$  d'équation approchée  $y = 85x - 77$  où  $(x, y) = (t, p)$  représente la taille et le poids ;
- on choisit la fonction d'activation sigmoïde.

Ce qui nous permet de définir le perceptron suivant avec  $a = 85$ ,  $b = -1$  et  $c = -77$ .



Maintenant pour un couple donné  $(t, p)$ , le perceptron associe une valeur  $F(t, p) \in [0, 1]$ . Si  $F(t, p)$  est proche de 0 alors la personne est probablement un homme, si  $F(t, p)$  est proche de 1 c'est probablement une femme.

Exemple : une personne mesurant 1.77 m et pesant 75 kg est-elle plutôt un homme ou une femme ? On calcule  $f(1.77, 75) = -1.55$  où  $f(x, y) = ax+by+c$ . Puis on calcule  $F(1.77, 75) = \sigma(-1.55) \simeq 0.17$ . Selon notre modèle cette personne est probablement un homme (car la valeur de  $F$  est proche de 0).

Autre exemple :  $(t, p) = (1.67, 64)$ . On calcule  $F(1.67, 64) \simeq 0.72$ . Une personne mesurant 1.67m et pesant 64kg est probablement une femme (car la valeur de  $F$  est plus proche de 1 que de 0).

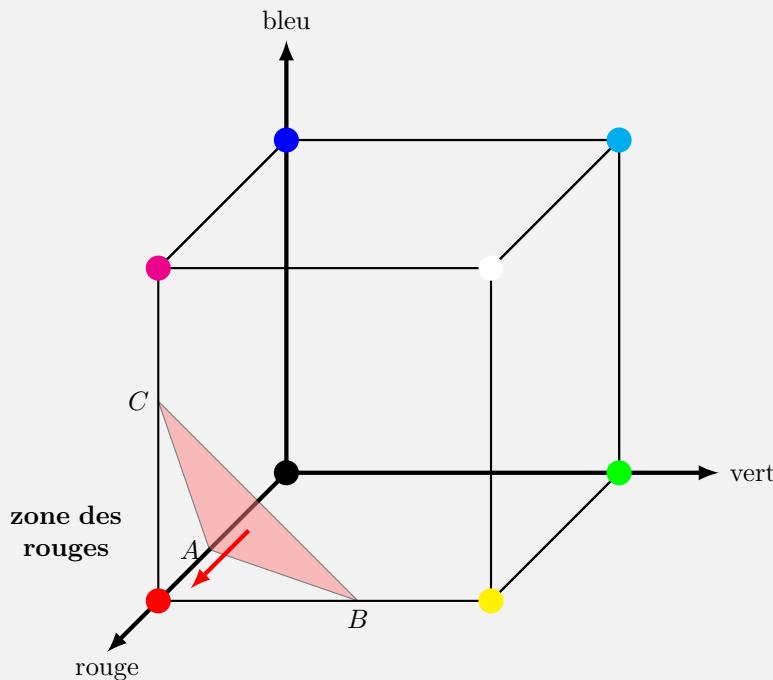
Malgré les données fictives, cet exemple met en évidence le problème de la superposition et de l'utilité d'avoir une sortie plus riche que « oui » ou « non ». On peut aussi discuter de la pertinence de la frontière, car la séparation par une droite ne semble pas la mieux adaptée. En fait, le poids varie en fonction du carré de la taille (les zones ont une forme de parabole).

Plus généralement un **perceptron affine** à  $n$  entrées est la donnée de  $n+1$  poids  $a_0, a_1, \dots, a_n \in \mathbb{R}$  et d'une fonction d'activation  $\phi$  qui définissent une fonction  $F : \mathbb{R}^n \rightarrow \mathbb{R}$  par la formule :

$$F(x_1, \dots, x_n) = \phi(a_1x_1 + \dots + a_nx_n + a_0).$$

### Exemple 2.3

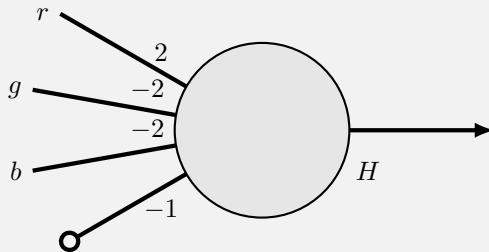
Dans le système de couleurs RGB (*red, green, blue*), une couleur est déterminée par trois réels  $r, g, b \in [0, 1]$ . Le « vrai » rouge est codé par  $(1, 0, 0)$ , mais bien sûr des couleurs proches sont aussi des nuances de rouge. On représente toutes les couleurs par un « cube de couleurs », chaque point du cube représente le code  $(r, g, b)$  d'une couleur.



On décide (un peu arbitrairement) que toute couleur située dans la zone du coin  $(1, 0, 0)$  de la figure ci-dessus est une nuance de rouge.

Problème : trouver un perceptron qui répond à la question « Cette couleur est-elle une nuance de rouge ? ».

Solution. On considère que la zone est délimitée par le plan passant par les points  $A(\frac{1}{2}, 0, 0)$ ,  $B(1, \frac{1}{2}, 0)$  et  $C(1, 0, \frac{1}{2})$  et les plans des faces du cube. Une équation de ce plan est  $2x - 2y - 2z - 1 = 0$  où en fait  $(x, y, z) = (r, g, b)$ . Le perceptron qui répond à la question est donc :



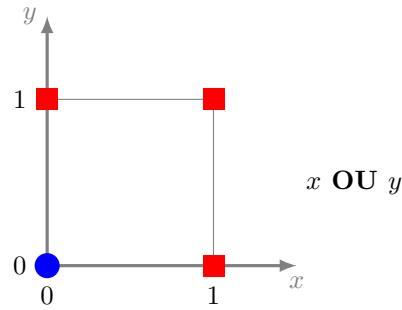
La fonction  $F$  associée, vérifie que  $F(r, g, b) = 1$  pour les points de la zone des rouges et  $F(r, g, b) = 0$  sinon.

## II- Théorie du perceptron

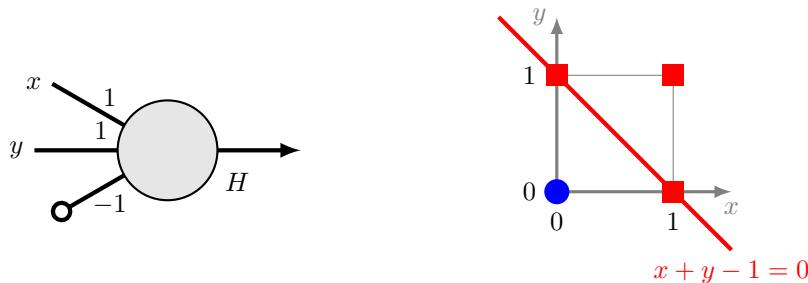
### 1. OU, ET, OU exclusif

**OU.** Un **booléen** est une variable qui peut prendre soit la valeur « vrai », soit la valeur « faux ». Dans la pratique, on associe 1 à « vrai » et 0 à « faux ». À partir de deux booléens  $x$  et  $y$ , on peut associer un nouveau booléen «  $x$  OU  $y$  » qui est vrai lorsque  $x$  est vrai ou  $y$  est vrai.

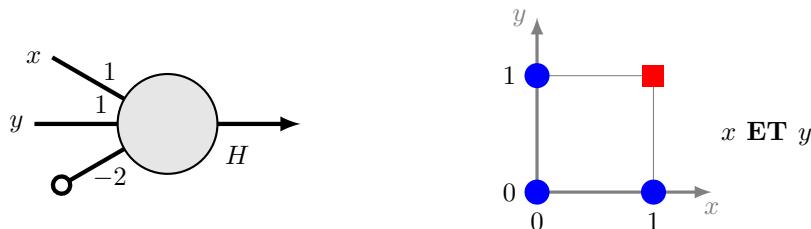
Graphiquement, on représente toutes les configurations associées à «  $x$  OU  $y$  » par un diagramme. Un rond bleu en position  $(x, y)$  signifie que «  $x$  OU  $y$  » est faux (le résultat vaut 0), un carré rouge que «  $x$  OU  $y$  » est vrai (le résultat vaut 1).



Peut-on réaliser l'opération «  $x$  OU  $y$  » par un perceptron ? Oui ! Cela revient à séparer les ronds bleus des carrés rouges. C'est possible avec le perceptron de poids  $a = 1$ ,  $b = 1$ ,  $c = -1$ . Par exemple, si  $x = 0$  et  $y = 1$  alors, on calcule  $1 \cdot 0 + 1 \cdot 1 - 1 = 0 \geq 0$ . Composée avec la fonction marche de Heaviside, la fonction  $F(x, y)$  définie par le perceptron renvoie dans ce cas 1 (« vrai »). Si  $x = 0$  et  $y = 0$ , alors  $1 \cdot 0 + 1 \cdot 0 - 1 = -1 < 0$  et  $F(x, y) = 0$  (« faux »).

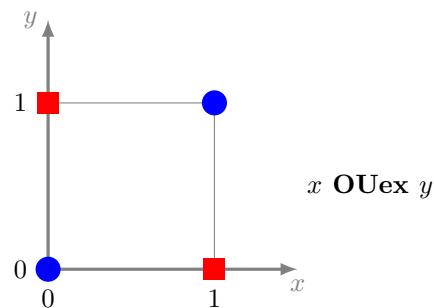


**ET.** On peut également réaliser l'opération «  $x$  ET  $y$  » (qui renvoie « vrai » uniquement si  $x$  et  $y$  sont vrais) en choisissant les poids  $a = 1$ ,  $b = 1$ ,  $c = -2$ .



D'un point de vue numérique, si on considère des valeurs réelles pour  $x$  et  $y$ , notre perceptron « ET » n'est pas numériquement très stable. Par exemple avec  $x = 0.9$  et  $y = 0.9$ , on calcule  $x + y - 2 = -0.2$  et la sortie est 0, mais comme  $x$  et  $y$  sont proches de 1, on aimerait que la sortie soit 1. Il suffit de changer un peu les paramètres : prenons  $a = 1$ ,  $b = 1$  et  $c = -1.5$ , alors  $x + y - 1.5 = 0.3$  et cette fois la sortie est 1.

**OU exclusif.** Le ou exclusif, noté «  $x$  OUex  $y$  » est vrai lorsque  $x$  ou  $y$  est vrai, mais pas les deux en même temps. Le ou exclusif est-il réalisable par un perceptron ? Cette fois la réponse est non !

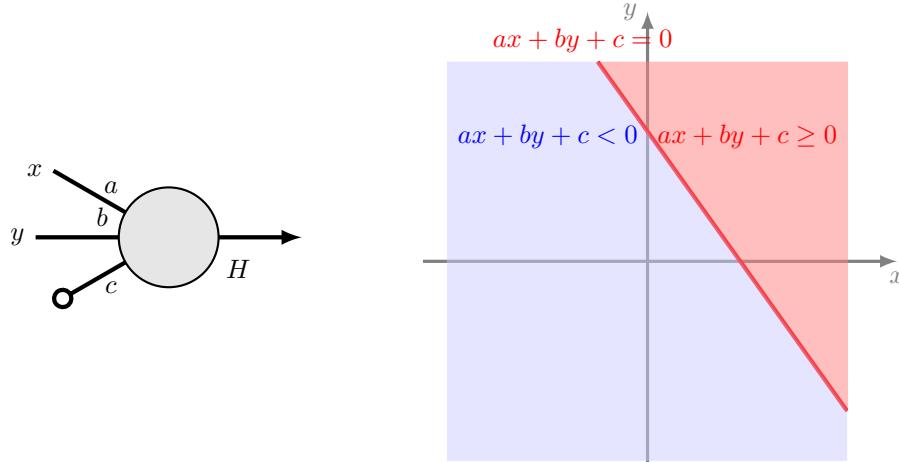


### Proposition 2.1

Il n'existe pas de perceptron (affine, à deux entrées et de fonction d'activation la fonction marche de Heaviside) qui réalise le « ou exclusif ».

Nous sommes convaincus géométriquement qu'il n'existe pas de droite qui sépare les ronds bleus des carrés rouges. Nous allons le prouver par un petit calcul.

*Démonstration.* Nous raisonnons par l'absurde en supposant qu'un tel perceptron existe. Nous cherchons une contradiction. Soit  $a_1 = a, a_2 = b, a_0 = c$  les coefficients.



- Pour  $(x, y) = (1, 0)$ , on doit avoir  $ax + by + c \geq 0$  (car après avoir composé avec la fonction d'activation le résultat doit être 1), donc

$$a + c \geq 0. \quad (2.1)$$

- De même pour  $(x, y) = (0, 1)$ , on doit avoir  $ax + by + c \geq 0$ , ce qui donne :

$$b + c \geq 0. \quad (2.2)$$

- Pour  $(0, 0)$  et  $(1, 1)$ , on a  $ax + by + c < 0$ , ce qui implique :

$$c < 0, \quad (2.3)$$

$$a + b + c < 0. \quad (2.4)$$

Si on additionne les inégalités (2.1) et (2.2), on obtient  $a + b + 2c \geq 0$ . Par l'inégalité (2.3) on a  $-c > 0$ . Donc en ajoutant  $-c$  à l'inégalité  $a + b + 2c \geq 0$ , on obtient  $a + b + c > 0$ . Ce qui contredit l'inégalité (2.4).

Conclusion : il ne peut exister trois réels  $a, b, c$  pour définir un perceptron réalisant le « ou exclusif ».

□

## 2. Séparation linéaire

Nous formalisons un peu les idées de la section précédente. Une **droite** du plan  $\mathbb{R}^2$  est l'ensemble des points  $(x, y)$  vérifiant une équation du type  $ax+by+c = 0$ . Un **plan** de l'espace  $\mathbb{R}^3$  est l'ensemble des points  $(x, y, z)$  vérifiant une équation du type  $ax + by + cz + d = 0$ . Nous généralisons cette notion en dimension plus grande.

### Définition II.1

Un **hyperplan (affine)** de  $\mathbb{R}^n$  est l'ensemble des points  $(x_1, \dots, x_n)$  qui vérifient une équation du type :

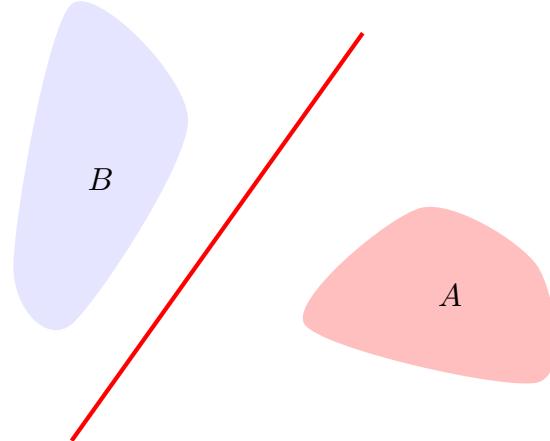
$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_0 = 0$$

où  $a_1, \dots, a_n$  sont des réels (non tous nuls) et  $a_0$  est aussi un réel.

Le but d'un perceptron affine est de séparer deux ensembles. Par exemple deux ensembles  $A$  et  $B$  du plan seront séparés par une droite, deux ensembles de l'espace sont séparés par un plan.

### Définition II.2

Deux ensembles  $A$  et  $B$  sont **linéairement séparables** s'il existe un hyperplan qui sépare  $A$  de  $B$ . Plus précisément, s'il existe des réels  $a_0, a_1, \dots, a_n$  tels que  $a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_0 \geq 0$  pour tout  $(x_1, \dots, x_n) \in A$  et  $a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_0 < 0$  pour tout  $(x_1, \dots, x_n) \in B$ .



Nous résumons ce qui précède dans le résultat suivant.

### Proposition 2.2

Deux ensembles  $A$  et  $B$  de  $\mathbb{R}^n$  sont linéairement séparables si, et seulement si, il existe un perceptron affine dont la fonction  $F$  vaut 1 sur  $A$  et 0 sur  $B$ .

La preuve est simple : la fonction  $f$  définie par un perceptron (avant activation) est une fonction affine :

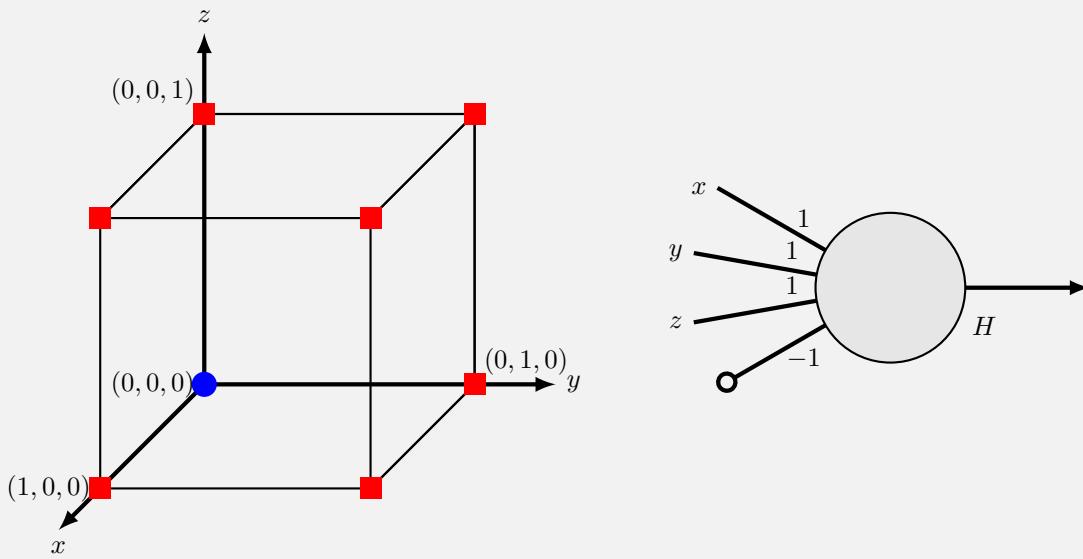
$$f(x_1, \dots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_0.$$

Si  $f(x_1, \dots, x_n) \geq 0$ , alors après activation  $F(x_1, \dots, x_n) = 1$ , sinon  $F(x_1, \dots, x_n) = 0$ .

### Exemple 2.1

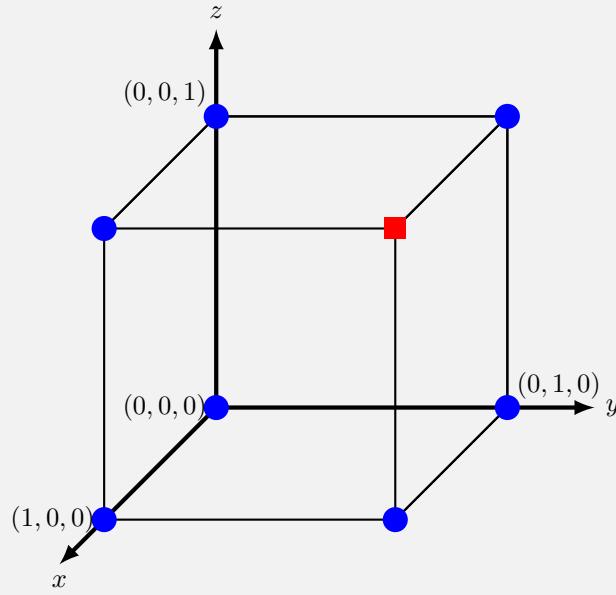
Réaliser «  $x$  OU  $y$  OU  $z$  » revient à séparer par un plan le rond bleu des carrés rouges du cube suivant. Trouver l'équation d'un de ces plans donne donc les poids du perceptron qui

conviennent.



### Exemple 2.2

Trouver le perceptron qui réalise «  $x$  ET  $y$  ET  $z$  ».

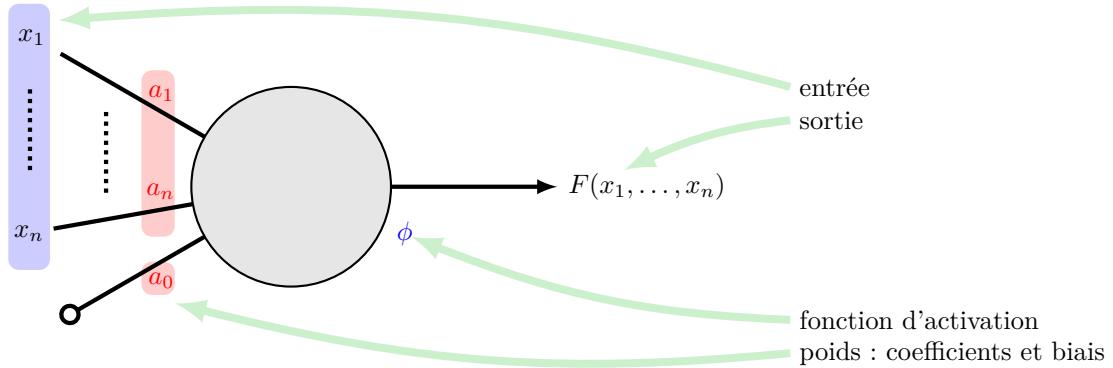


### 3. Vocabulaire

Résumons le vocabulaire utilisé ci-dessus ainsi que les termes anglais correspondant :

- le **perceptron linéaire** ou **neurone artificiel**, a un biais qui vaut 0, à la différence du **perceptron affine** pour lequel le biais est un réel quelconque ;
- les **poids** (*weights*) ou **paramètres** sont les **coefficients**  $a_1, \dots, a_n$  auxquels s'ajoute le **biais** (*bias*) (dans ce livre le biais est l'un des poids, ce qui n'est pas toujours le cas dans la littérature) ;
- un perceptron est la donnée des poids et d'une fonction d'activation ;
- chaque perceptron définit une fonction  $F$  qui est la composée d'une **fonction affine**  $f$  et d'une **fonction d'activation**  $\phi$  ; la fonction d'activation la plus utilisée dans ce chapitre est la fonction marche de Heaviside (*step function*) ;

- on utilise un perceptron lors d'une **évaluation** : pour une **entrée** (*input*)  $(x_1, \dots, x_n) \in \mathbb{R}^n$ , on calcule la sortie (*output*)  $F(x_1, \dots, x_n) \in \mathbb{R}$  (qui vaut 0 ou 1 dans le cas de la fonction marche de Heaviside).

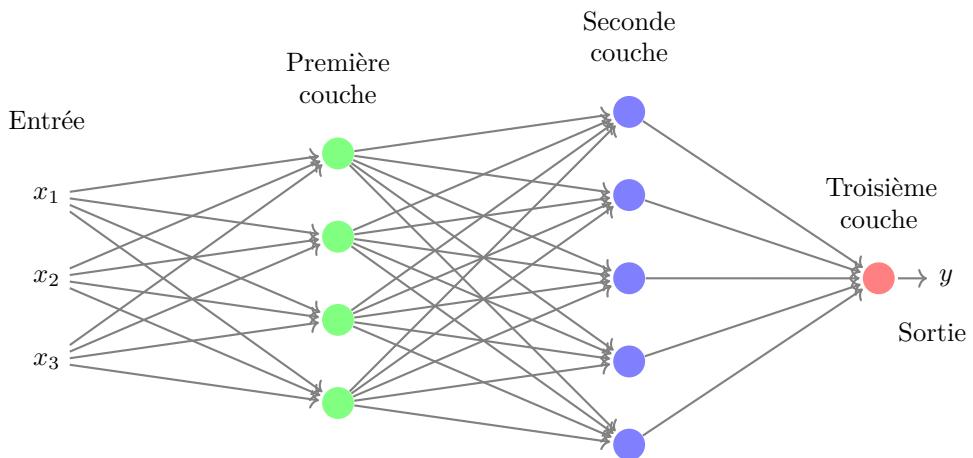


### III- Réseau de neurones

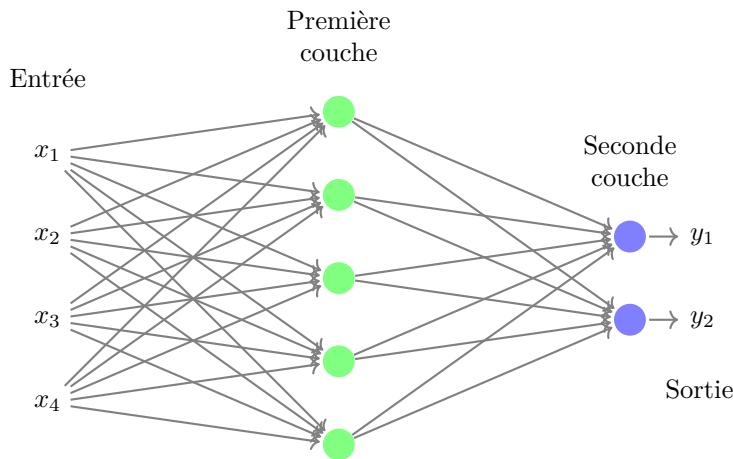
Un neurone permet de séparer l'espace en deux parties, la frontière étant linéaire (avec deux entrées c'est une droite, avec trois entrées c'est un plan...). Un neurone est trop élémentaire pour résoudre nos problèmes, par exemple il ne peut réaliser le « ou exclusif ». Comment faire mieux ? En connectant plusieurs neurones !

#### 1. Couches de neurones

Un **réseau de neurones** est la juxtaposition de plusieurs neurones, regroupés par **couches**.

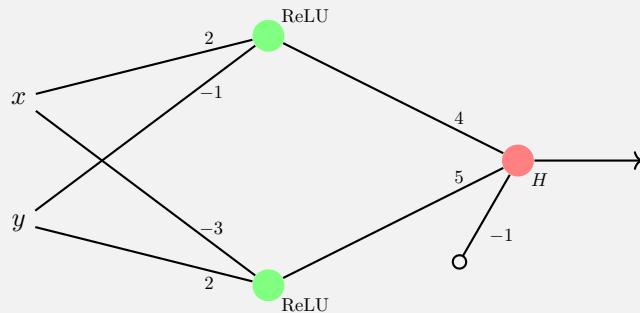


À un réseau de neurones on associe une fonction. Si à la dernière couche la fonction ne contient qu'un seul neurone (voir ci-dessus), cette fonction est  $F : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $y = F(x_1, \dots, x_n)$  où  $(x_1, \dots, x_n)$  est l'**entrée** et  $y$  est la **sortie**. Sinon (voir ci-dessous), la fonction est  $F : \mathbb{R}^n \rightarrow \mathbb{R}^p$ ,  $(y_1, \dots, y_p) = F(x_1, \dots, x_n)$  où  $(x_1, \dots, x_n) \in \mathbb{R}^n$  est l'entrée et  $(y_1, \dots, y_p)$  est la sortie.

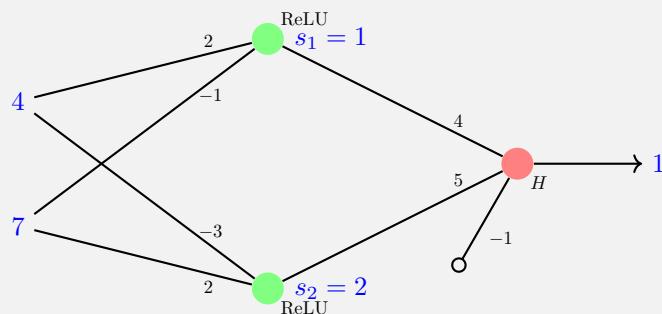


### Exemple 1.1

Voici un réseau de neurones à deux couches : 2 neurones (perceptrons linéaires) sur la première couche (ayant pour fonction d'activation la fonction ReLU), 1 neurone (perceptron affine) sur la seconde couche (de fonction d'activation  $H$ ).



- Si  $x = 4$  et  $y = 7$  alors on calcule la sortie de chaque neurone de la première couche. Ces sorties sont les entrées du neurone de la seconde couche. Pour le premier neurone, on effectue le calcul  $2 \cdot 4 + (-1) \cdot 7 = 1 \geq 0$ , le réel étant positif la fonction ReLU le garde inchangé : le premier neurone renvoie la valeur  $s_1 = 1$ . Le second neurone effectue le calcul  $(-3) \cdot 4 + 2 \cdot 7 = 2 \geq 0$  et renvoie  $s_2 = 2$ . Le neurone de la couche de sortie reçoit en entrées  $s_1$  et  $s_2$  et effectue le calcul  $4 \cdot 1 + 5 \cdot 2 - 1 = 13 \geq 0$ . La fonction d'activation étant  $H$ , ce neurone renvoie 1. Ainsi  $F(4, 7) = 1$ .

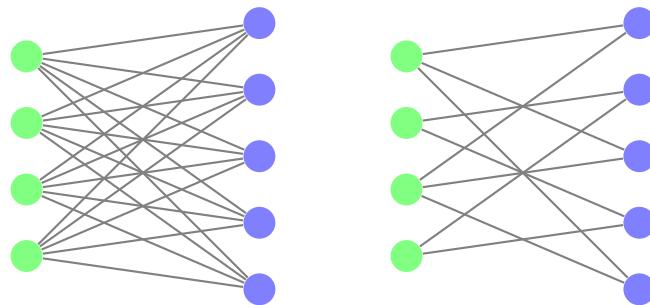


- Si  $(x, y) = (3, 2)$  alors  $s_1 = 4$ , par contre  $s_2 = 0$  car  $(-3) \cdot 3 + 2 \cdot 2 = -5 < 0$  et la fonction ReLU renvoie 0 (on dit que le neurone ne s'active pas). Les entrées du dernier neurone sont donc  $s_1 = 4$  et  $s_2 = 0$ , ce neurone calcule  $4 \cdot 4 + 5 \cdot 0 - 1 = 15 \geq 0$  et, après la fonction d'activation, renvoie 1. Donc  $F(3, 2) = 1$ .
- Vérifier que pour  $(x, y) = (\frac{1}{10}, \frac{1}{10})$  le premier neurone s'active, le second ne s'active pas (il renvoie 0) et le dernier neurone renvoie 0. Ainsi  $F(\frac{1}{10}, \frac{1}{10}) = 0$ .

- Chaque neurone est défini par ses poids et une fonction d'activation. Pour une couche donnée, on choisit toujours la même fonction d'activation. En général on choisit la même fonction d'activation pour tout le réseau, sauf peut-être pour la couche de sortie.
- Un neurone prend plusieurs valeurs en entrée mais ne renvoie qu'une seule valeur en sortie ! Si ce neurone est connecté à plusieurs autres neurones, il envoie la même valeur à tous.



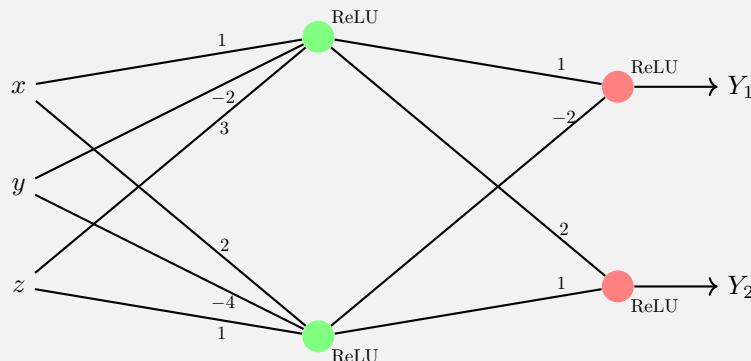
- On peut relier un neurone à tous les neurones de la couche suivante (voir ci-dessous figure de gauche). On dit que la seconde couche est **dense** ou **complètement connectée**. Mais on peut aussi choisir de ne relier que certains neurones entre eux (voir ci-dessous figure de droite). S'il n'y a pas d'arêtes du neurone  $A$  vers le neurone  $B$ , c'est que la sortie de  $A$  n'intervient pas comme entrée de  $B$  (cela revient à imposer un poids nul entre ces deux neurones).



### Exemple 1.2

Pour le réseau de neurones représenté ci-dessous, calculer les valeurs de sortie ( $Y_1, Y_2$ ) pour chacune des entrées  $(x, y, z)$  suivantes :

$$(0, 0, 0) \quad (1, 0, 0) \quad (1, -1, 1) \quad (3, 2, 1)$$



Trouver un  $(x, y, z)$  tel que  $Y_1 = 1$  et  $Y_2 = 7$  (commencer par déterminer les valeurs de sorties  $s_1$  et  $s_2$  de la première couche).

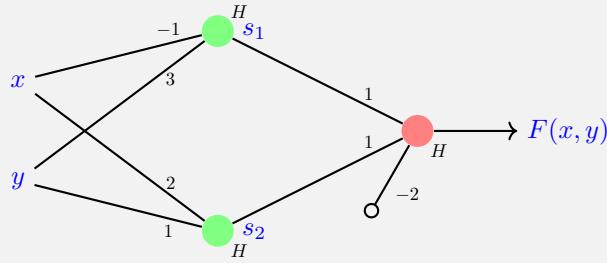
Existe-t-il  $(x, y, z)$  tel que  $Y_2$  soit nul, mais pas  $Y_1$  ?

## 2. Exemples

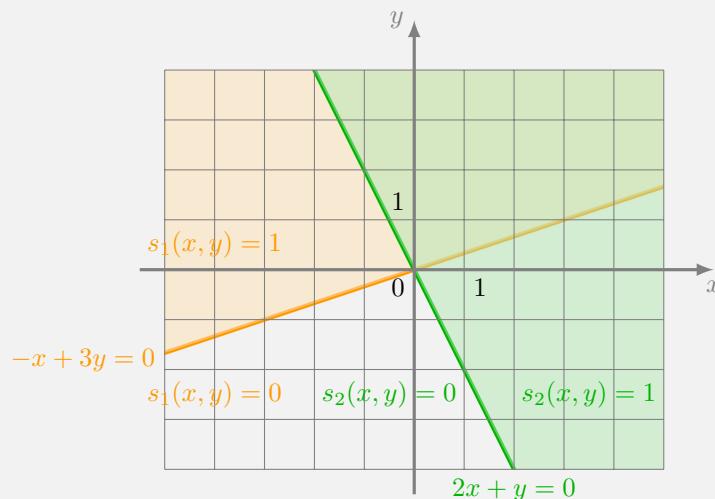
Étudions quelques exemples plus en détails. Au lieu de calculer la sortie  $F(x, y)$  pour des valeurs données, nous allons calculer  $F(x, y)$  quelque soit  $(x, y)$ .

### Exemple 2.1

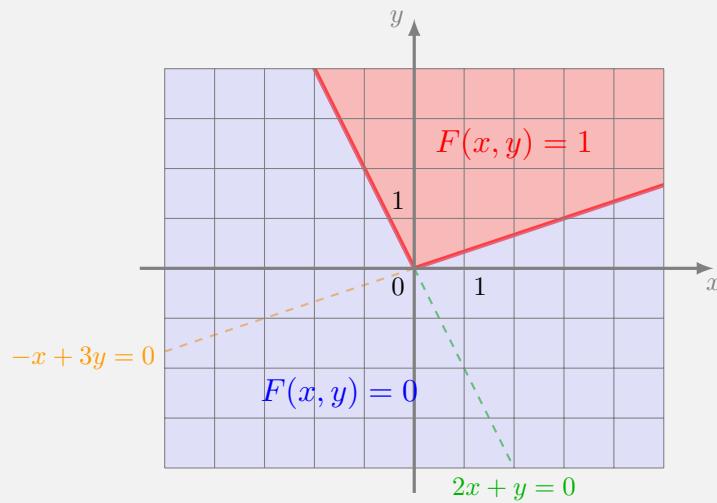
Voici un réseau de 3 neurones : deux sur la première couche et un sur la seconde. La fonction d'activation est partout la fonction marche de Heaviside. Combien vaut la fonction associée  $F$  selon l'entrée  $(x, y)$  ?



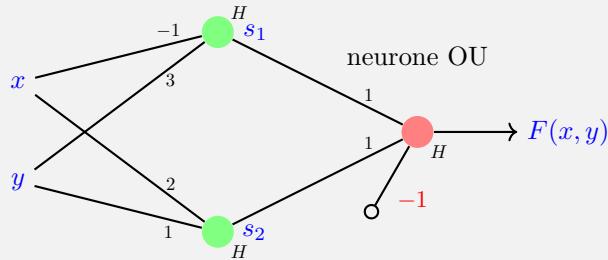
On commence par calculer les sorties des neurones de la première couche. Pour le premier neurone la sortie  $s_1$  dépend du signe de  $-x + 3y$ . Si  $-x + 3y \geq 0$  alors  $s_1(x, y) = 1$ , sinon  $s_1(x, y) = 0$ . Donc pour les points  $(x, y)$  situés au-dessus de la droite d'équation  $-x + 3y = 0$ , on a  $s_1(x, y) = 1$ . De même pour le second neurone, on a  $s_2(x, y) = 1$  pour les points situés au dessus de la droite  $2x + y = 0$ . Voir la figure ci-dessous.



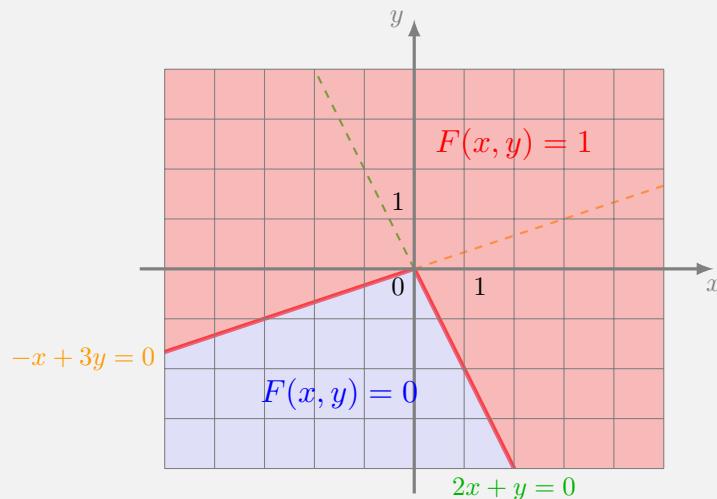
On reconnaît dans le neurone de sortie un neurone qui réalise le « ET ». C'est pourquoi l'ensemble des points pour lesquels  $F$  vaut 1 est l'intersection des deux demi-plans en lesquels  $s_1$  et  $s_2$  valent 1. Ainsi  $F(x, y) = 1$  dans un secteur angulaire et  $F(x, y) = 0$  ailleurs. Voir la figure ci-dessous.

**Exemple 2.2**

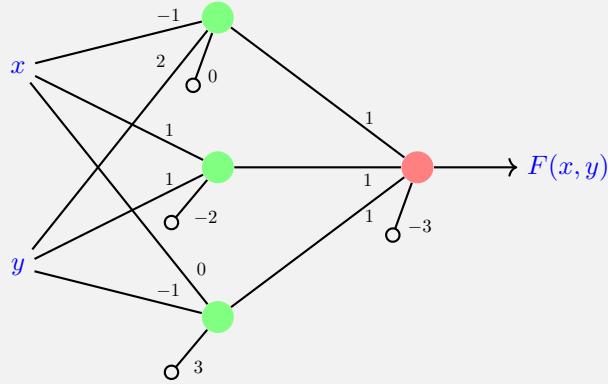
On reprend la même architecture que l'exemple précédent, mais en changeant les poids du neurone de sortie qui réalise cette fois l'opération « OU ».



L'ensemble des points pour lesquels  $F$  vaut 1 est maintenant l'union des deux demi-plans en lesquels  $s_1$  et  $s_2$  valent 1.

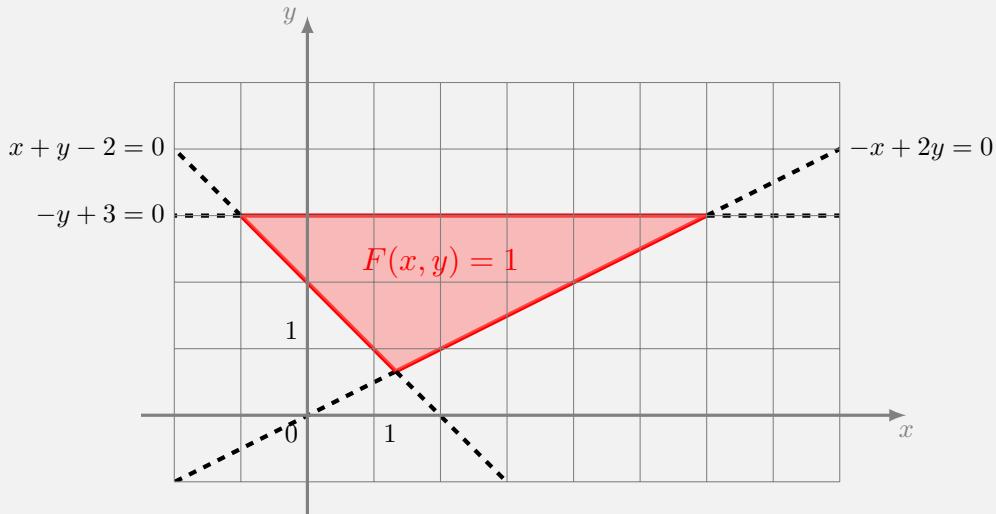
**Exemple 2.3**

Voici un réseau de neurones un peu plus compliqué (la fonction d'activation est  $H$  partout).



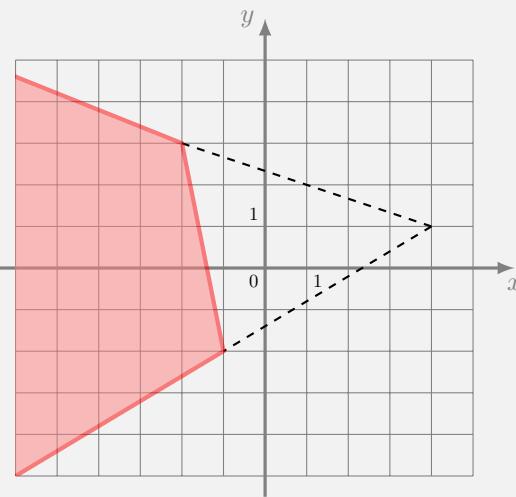
Chaque neurone de la première couche délimite un demi-plan. Ce sont les demi-plans  $-x + 2y \geq 0$ ,  $x + y - 2 \geq 0$  et  $-y + 3 \geq 0$ .

Le neurone de sortie est un neurone qui réalise l'opération « ET » : il s'active uniquement si les trois précédents neurones sont activés. Ainsi la sortie finale  $F(x, y)$  vaut 1 si et seulement si  $(x, y)$  appartient simultanément aux trois demi-plans.



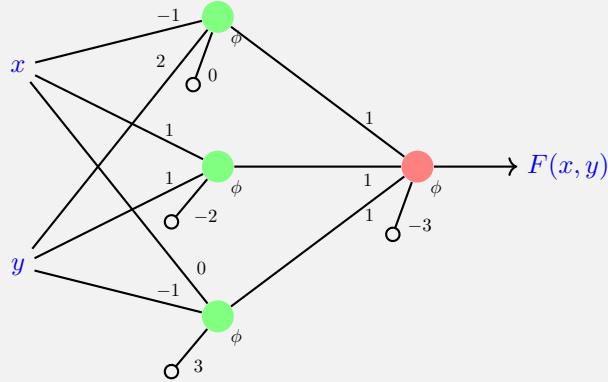
#### Exemple 2.4

En utilisant les idées de l'exemple précédent et pour le dessin ci-dessous, trouver un réseau de neurones dont la fonction  $F$  vaut 1 pour la zone colorée et 0 ailleurs.

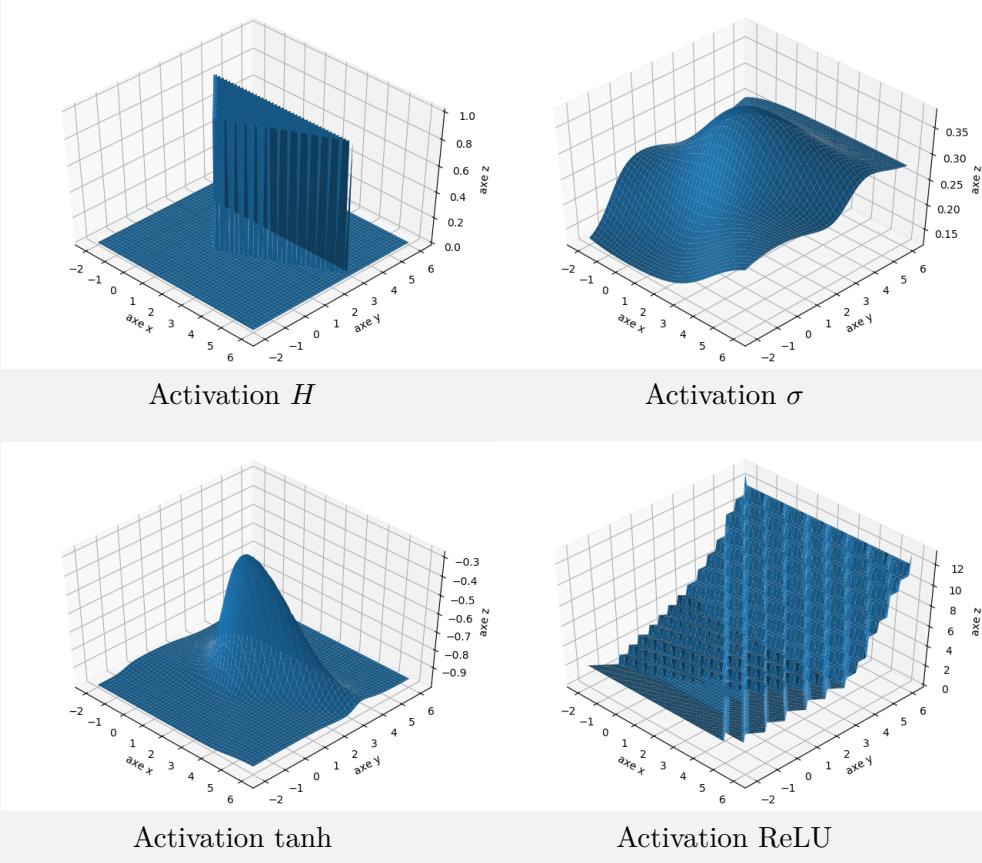


**Exemple 2.5**

On termine en comparant les fonctions produites par des réseaux ayant la même architecture, les mêmes poids mais de fonction d'activation  $\phi$  différente :  $H$ ,  $\sigma$ , tanh et ReLU.



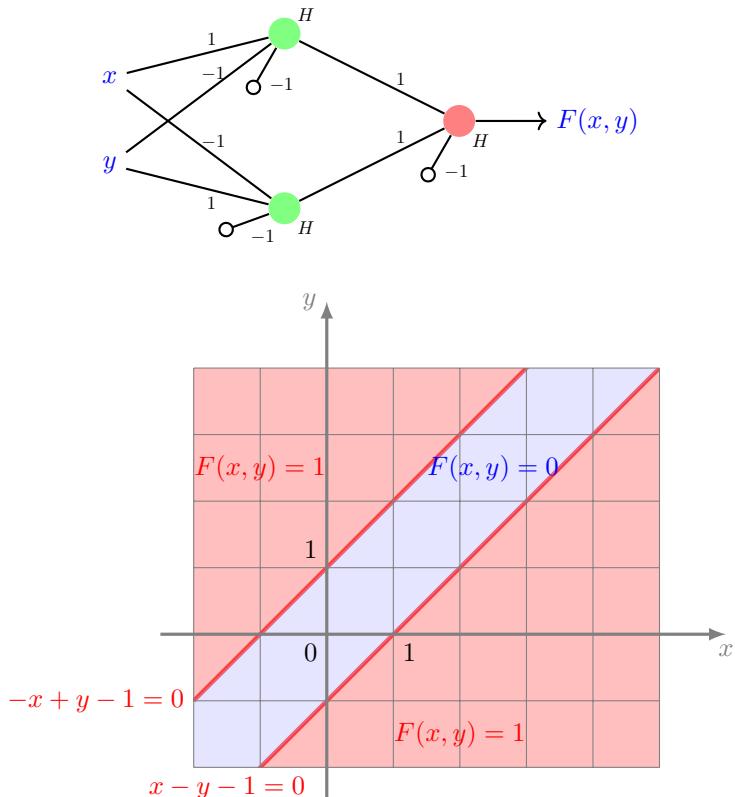
Voici les graphiques 3D obtenus pour les fonctions d'activation  $H$  (comme dans l'exemple vu auparavant),  $\sigma$ , tanh et ReLU. Les fonctions  $F$  obtenues dépendent fortement du choix de la fonction d'activation.



## IV- Théorie des réseaux de neurones

### 1. OU exclusif

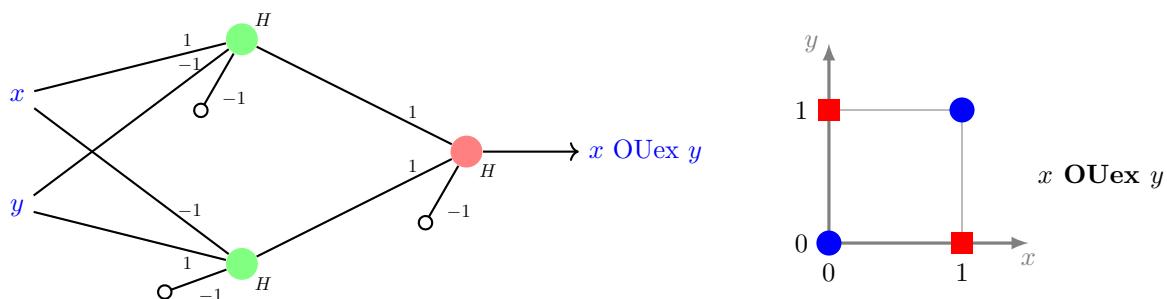
Nous avons vu qu'un seul neurone ne permet pas de réaliser la fonction associée au « OU exclusif ». Avec plusieurs neurones c'est facile ! Voici un réseau de neurones qui sépare le plan en trois parties, le secteur central en lequel la fonction associée au réseau vaut 0, alors que la fonction vaut 1 partout ailleurs (y compris sur la frontière).



Ce réseau permet d'obtenir une valeur  $F(x, y) = 1$  en  $(1, 0)$  et  $(0, 1)$  et une valeur  $F(x, y) = 0$  en  $(0, 0)$  et  $(1, 1)$ .

L'idée de ce réseau vient du fait que l'opération « OU exclusif » est une combinaison de « OU » et de « ET » :

$$x \text{ OUex } y = (x \text{ ET non}y) \text{ OU } (\text{non}x \text{ ET } y).$$

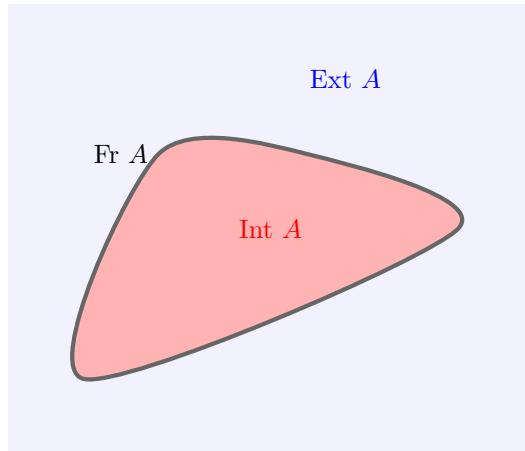


Le neurone du haut réalise «  $x$  ET non  $y$  », le neurone du bas « non  $x$  ET  $y$  » et celui de sortie l'opération « OU ».

## 2. Ensemble réalisable

Nous aimerais savoir quels ensembles peuvent être décrits par un réseau de neurones. On rappelle qu'un ensemble  $A \subset \mathbb{R}^2$  découpe le plan en trois parties disjointes : intérieur, frontière, extérieur :

$$\text{Int}(A) \quad \text{Fr}(A) \quad \text{Ext}(A).$$



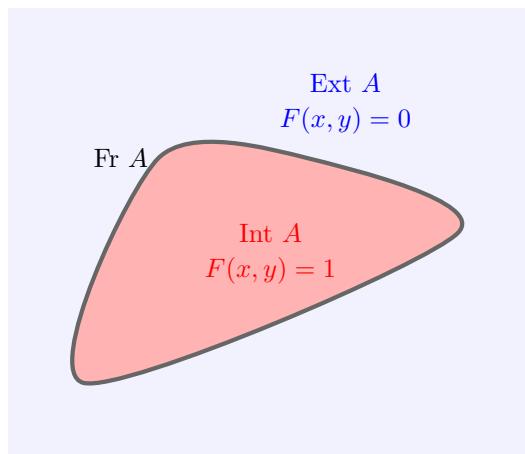
### Définition IV.1

Un ensemble  $A$  est dit **réalisable par un réseau de neurones** s'il existe un réseau de neurones  $\mathcal{R}$  (d'unique fonction d'activation la fonction marche de Heaviside) tel que la fonction  $F : \mathbb{R}^2 \rightarrow \mathbb{R}$  associée à  $\mathcal{R}$  vérifie :

$$F(x, y) = 1 \quad \text{pour tout } (x, y) \in \text{Int}(A)$$

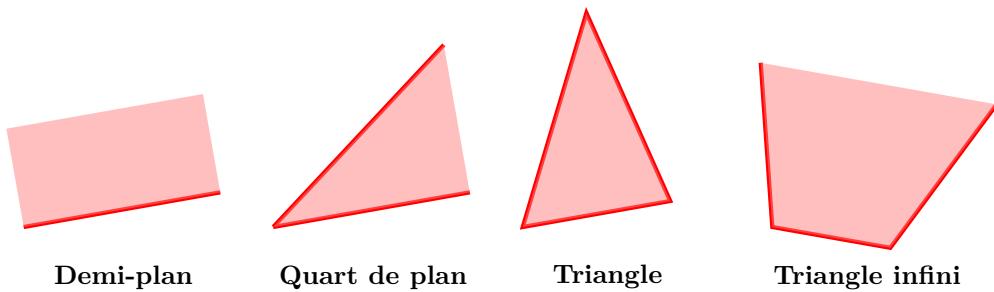
et

$$F(x, y) = 0 \quad \text{pour tout } (x, y) \in \text{Ext}(A).$$



Remarque : on n'exige rien sur la frontière  $\text{Fr}(A)$ ,  $F$  peut y prendre la valeur 0 ou la valeur 1.  
Voici les types d'ensembles que nous avons déjà réalisés :

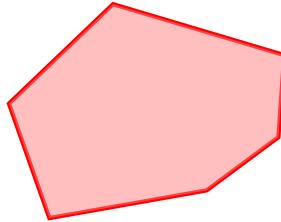
- les demi-plans,
- les « quart de plans »,
- les zones triangulaires,
- les triangles avec un sommet « à l'infini ».



En augmentant le nombre de neurones, on peut réaliser les quadrilatères convexes et plus généralement n'importe quelle zone polygonale convexe.

### Proposition 2.3

Tout zone polygonale convexe à  $n$  côtés est réalisable par un réseau de  $n + 1$  neurones.



Polygone convexe

*Démonstration.* Un polygone convexe à  $n$  côtés est l'intersection de  $n$  demi-plans. Chaque demi-plan, bordé par une droite d'équation du type  $ax + by + c = 0$ , correspond à un neurone de la première couche dont les coefficients sont  $(a, b)$  et le biais est  $c$  (ou alors  $(-a, -b)$  et  $-c$ ). Sur la seconde couche, on place un neurone qui réalise l'opération « ET » sur les  $n$  entrées : tous ses coefficients sont 1 et le biais est  $-n$ .  $\square$

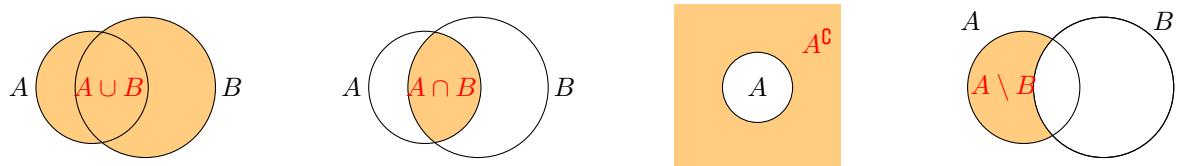
Continuons avec des opérations élémentaires sur les ensembles réalisables.

### Proposition 2.4

Si  $A$  et  $B$  sont deux ensembles du plan réalisables par des réseaux de neurones alors :

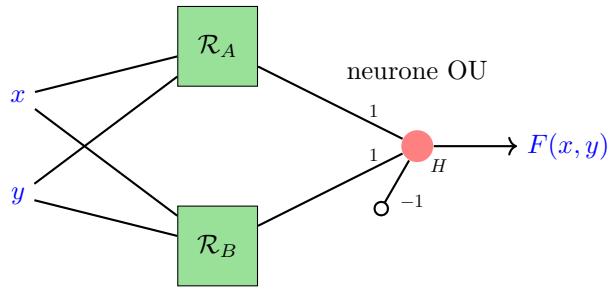
$$A \cup B \quad A \cap B \quad A^c \quad A \setminus B$$

sont aussi des ensembles réalisables.

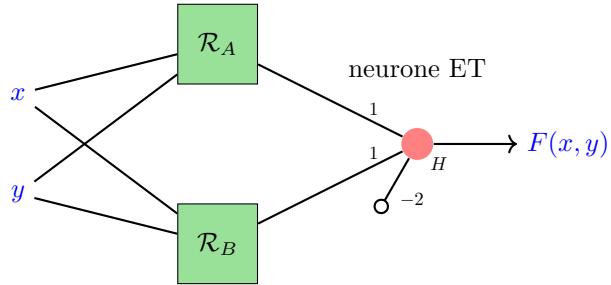


*Démonstration.*

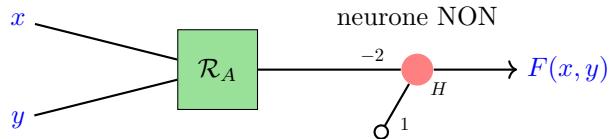
- Si  $A$  est réalisé par un réseau  $\mathcal{R}_A$  et  $B$  par un réseau  $\mathcal{R}_B$  alors on crée un nouveau réseau  $\mathcal{R}$  en superposant  $\mathcal{R}_A$  et  $\mathcal{R}_B$  et en ajoutant un neurone « OU » à partir des sorties de  $\mathcal{R}_A$  et  $\mathcal{R}_B$ . Ainsi si  $(x, y)$  est dans  $A \cup B$ , il est dans  $A$  ou dans  $B$ , une des sorties  $\mathcal{R}_A$  ou  $\mathcal{R}_B$  vaut alors 1 et le neurone sortant de  $\mathcal{R}$  s'active.



- Pour réaliser  $A \cap B$ , on remplace le neurone « OU » par un neurone « ET ».



- Pour réaliser le complément d'un ensemble  $A$ , on utilise l'opération « non » : 0 s'envoie sur 1 et 1 sur 0. Ceci se fait par l'application  $s \mapsto H(1 - 2s)$ . Il suffit juste de rajouter un neurone « NON » à  $\mathcal{R}_A$ .



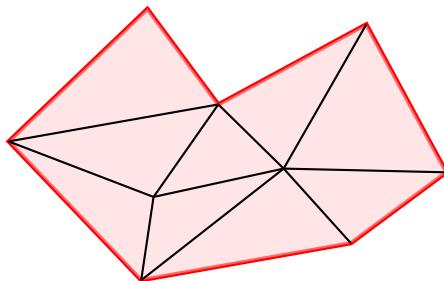
- Comme  $A \setminus B = A \cap ((A \cap B)^c)$  alors il est possible de réaliser  $A \setminus B$  comme succession d'opérations élémentaires  $\cap$ ,  $c$  et  $\cup$ .

□

### Proposition 2.5

Tout polygone (convexe ou non) est réalisable par un réseau de neurones.

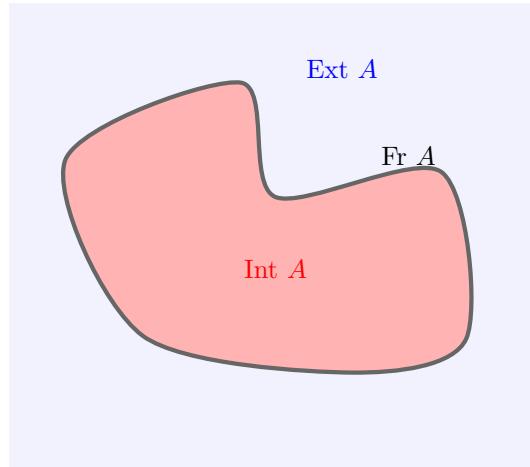
*Démonstration.* Un polygone peut être découpé en triangles. Chaque triangle est réalisable, donc l'union des triangles l'est aussi.



□

### 3. Approximation d'ensembles

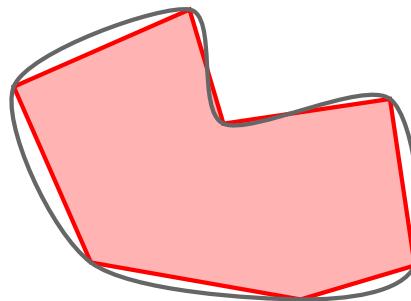
On rappelle qu'une **courbe de Jordan**  $C$  est une courbe fermée simple (c'est l'image d'un cercle par une application continue injective). Le théorème suivant est une sorte de théorème d'approximation universelle géométrique.



#### Théorème 2.1

Un ensemble  $A$  délimité par une courbe de Jordan peut être approché d'aussi près que l'on veut par un ensemble  $A'$  réalisable par un réseau de neurones.

Il s'agit juste du fait qu'une courbe de Jordan peut être approchée par un polygone. C'est un résultat théorique qui ne dit en rien comment choisir la structure du réseau ou les poids.



## V- Théorème d'approximation universelle

Nous allons maintenant prouver qu'un réseau de neurones bien construit peut approcher n'importe quelle fonction.

Dans cette section nous partirons d'une seule entrée  $x \in \mathbb{R}$ , avec une seule sortie  $y = F(x) \in \mathbb{R}$ . Les réseaux de neurones de cette section produisent donc des fonctions  $F : \mathbb{R} \rightarrow \mathbb{R}$ .

L'objectif est le suivant : on nous donne une fonction  $f : [a, b] \rightarrow \mathbb{R}$  et nous devons trouver un réseau, tel que la fonction  $F$  associée à ce réseau soit proche de  $f$  :

$$F(x) \simeq f(x) \quad \text{pour tout } x \in [a, b].$$

Pour paramétriser le réseau nous allons bien sûr fixer des poids, mais avant cela choisissons les fonctions d'activation :

- pour le neurone de sortie, on choisit la fonction identité  $\phi(x) = x$ ,
- pour tous les autres neurones, on choisit la fonction marche de Heaviside.

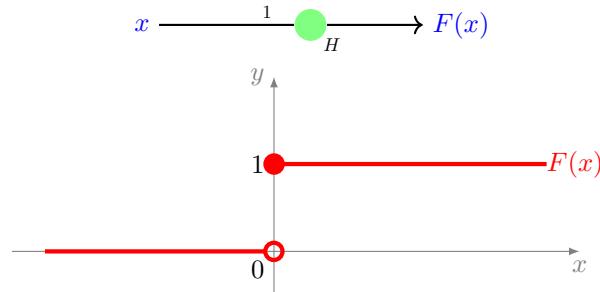
- Si on choisissait aussi la fonction d'activation marche de Heaviside pour le neurone de sortie, alors  $F$  ne pourrait prendre que deux valeurs, 0 ou 1, ce qui empêcherait de réaliser la plupart des fonctions.
- Par contre, si on choisissait la fonction identité pour tous les neurones alors on ne réaliseraient que des fonctions affines  $F(x) = ax + b$  (en effet la composition de plusieurs fonctions affines reste une fonction affine).

## 1. Fonctions marches

Nous allons réaliser des fonctions de plus en plus compliquées à partir d'éléments très simples. Commençons par étudier le comportement d'un seul neurone avec la fonction d'activation marche de Heaviside (on rajoutera le neurone de sortie plus tard).

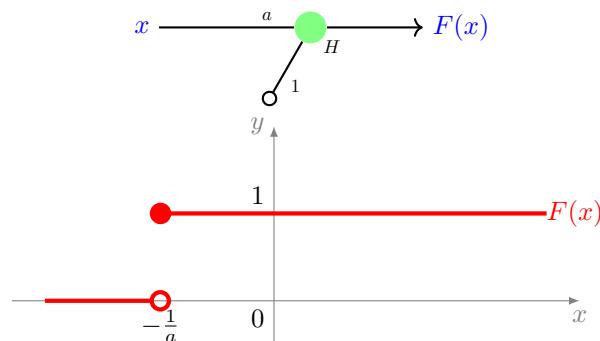
Voici différents neurones et les fonctions qu'ils réalisent :

- La fonction marche de Heaviside.

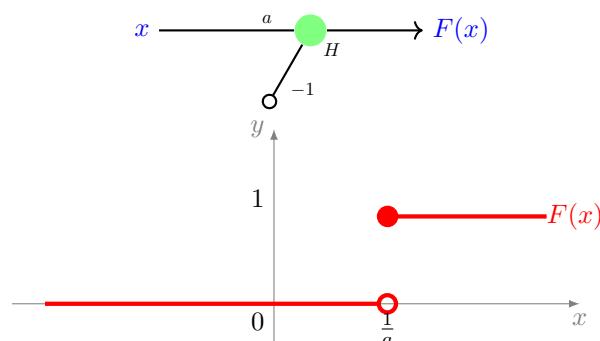


- La fonction marche décalée vers la gauche, avec  $a > 0$ .

Preuve :  $F(x) = 1 \iff ax + 1 \geq 0 \iff x \geq -\frac{1}{a}$ .

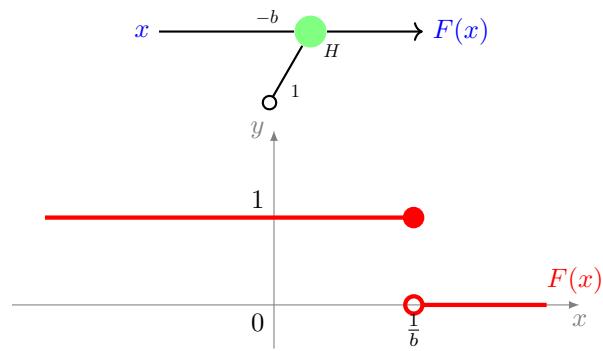


- La fonction marche décalée vers la droite, avec  $a > 0$ .

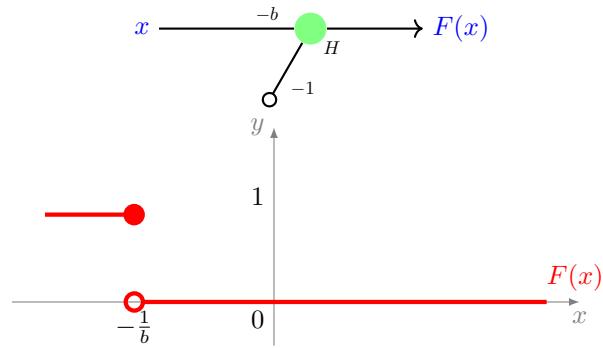


- La fonction marche à l'envers décalée vers la droite, avec  $b > 0$ .

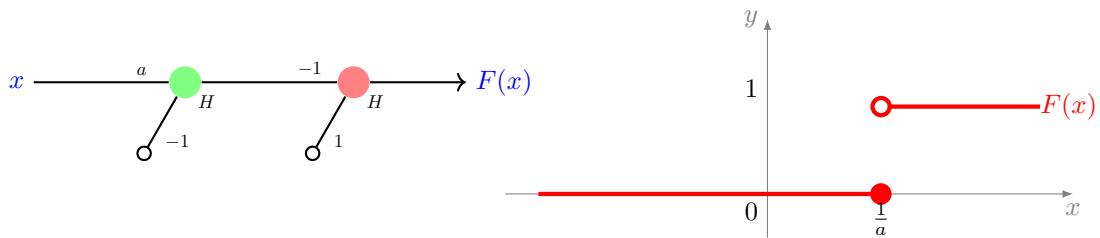
Preuve :  $F(x) = 1 \iff -bx + 1 \geq 0 \iff bx \leq 1 \iff x \leq \frac{1}{b}$ .



— La fonction marche à l'envers décalée vers la gauche, avec  $b > 0$ .

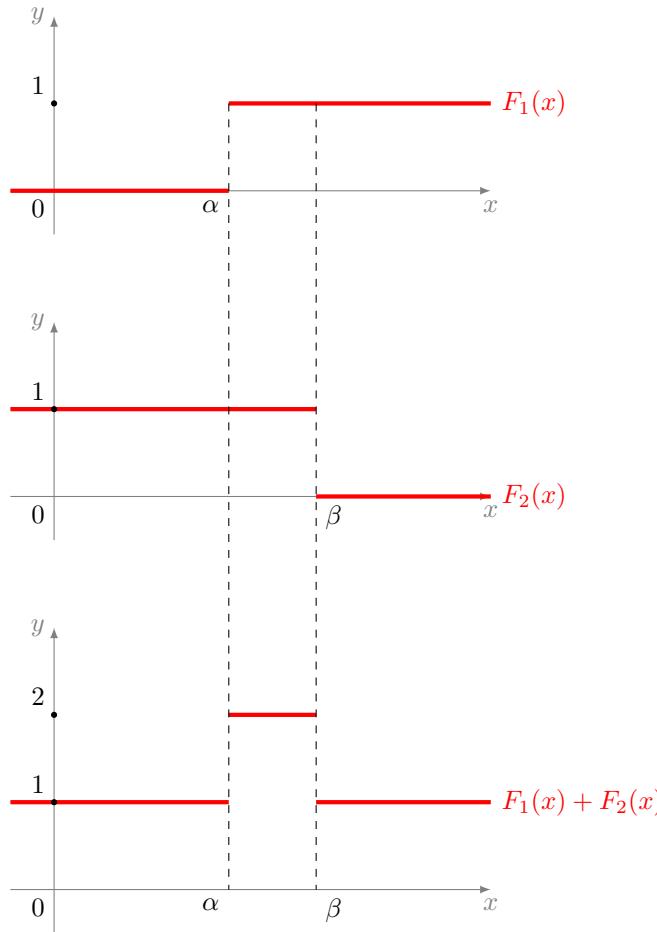


Selon les cas, la valeur au niveau de la marche est 0 ou 1. On peut obtenir l'autre situation en rajoutant un neurone de type « NON ».

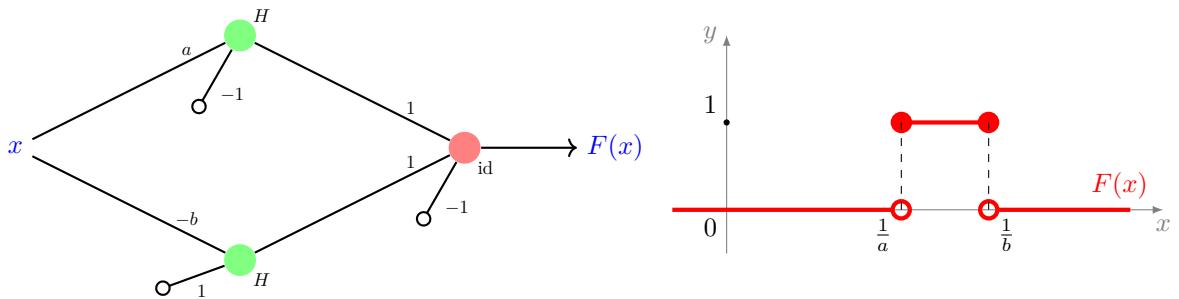


## 2. Fonctions créneaux

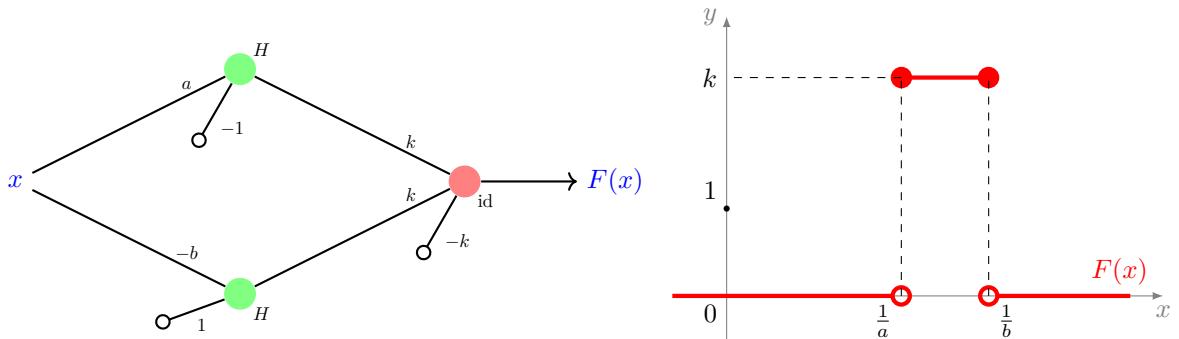
Pour réaliser un « créneau », l'idée est d'additionner une marche à l'endroit et une marche à l'envers.



Pour effectuer cette opération, nous allons construire un réseau avec deux neurones sur la première couche (de fonction d'activation  $H$ ) et un neurone sur la seconde couche (de fonction d'activation identité) qui additionne les deux sorties précédentes et soustrait 1 (afin de ramener la ligne de base à 0).

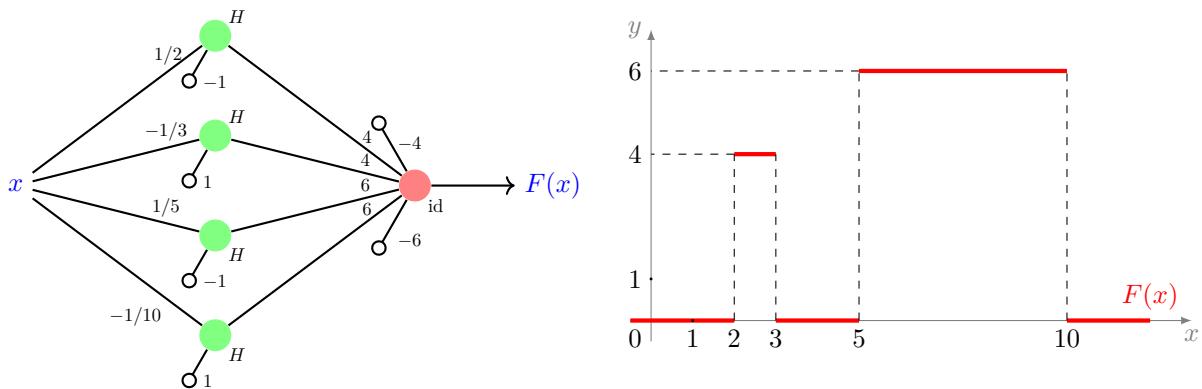


Si on veut une marche plus haute il suffit de changer les poids du neurone de sortie d'un facteur  $k$ .



### 3. Fonctions en escalier

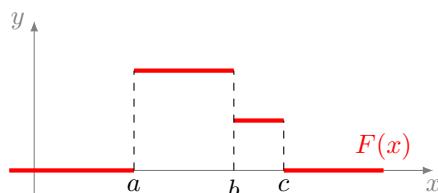
On réalise des doubles créneaux en superposant les premières couches de chaque créneau et en réunissant les deux neurones de sortie. Voici un exemple avec un créneau de hauteur 4 sur  $[2, 3]$  et un créneau de hauteur 6 sur  $[5, 10]$ .



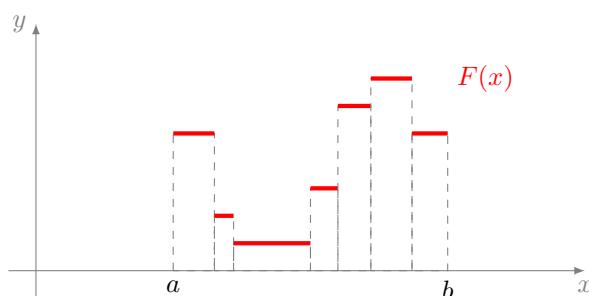
On peut aussi calculer la valeur de la fonction  $F$  de la façon suivante ( $s_i$  représente la sortie du neurone numéro  $i$  de la première couche) :

$$F(x) = \underbrace{4s_1 + 4s_2 - 4}_{\text{vaut 4 ou 0}} + \underbrace{6s_3 + 6s_4 - 6}_{\text{vaut 6 ou 0}} = \begin{cases} 4 & \text{si } x \in [2, 3[ \\ 6 & \text{si } x \in [5, 10[ \\ 0 & \text{sinon} \end{cases}$$

- Noter l'écriture avec deux biais  $-4$  et  $-6$  pour le neurone de sortie. C'est juste une écriture pour décomposer et expliquer le « vrai » biais qui est  $-4 - 6 = -10$ .
- Il peut y avoir un problème pour réaliser deux créneaux contigus. Si on n'y prend pas garde, la valeur est mauvaise à la jonction (c'est la somme des deux valeurs). Pour corriger le problème, il faut utiliser une marche où on a changé la valeur au bord, voir la fin de la section 1.



Une **fonction en escalier** est une fonction qui est constante sur un nombre fini d'intervalles bornés.



#### Proposition 2.6

Toute fonction en escalier est réalisable par un réseau de neurones.

*Démonstration.* Soit  $n$  le nombre de marches de l'escalier. On construit un réseau de  $2n + 1$  neurones. La première couche est constituée de  $n$  paires de neurones, chaque paire réalise une marche de l'escalier. La seconde couche contient uniquement le neurone de sortie, les coefficients sont choisis pour ajuster la hauteur de la marche et le biais assure que la fonction vaut 0 en dehors de l'escalier.

Si on veut les valeurs exactes aux bornes des marches, il faut éventuellement ajouter des neurones de type « NON » entre la première couche et le neurone de sortie.  $\square$

#### 4. Théorème d'approximation universelle (une variable)

Nous pouvons maintenant énoncer le résultat théorique le plus important de ce chapitre.

##### Théorème 2.2 : Théorème d'approximation universelle

Toute fonction continue  $f : [a, b] \rightarrow \mathbb{R}$  peut être approchée d'aussi près que l'on veut par une fonction  $F : [a, b] \rightarrow \mathbb{R}$  réalisée par un réseau de neurones.

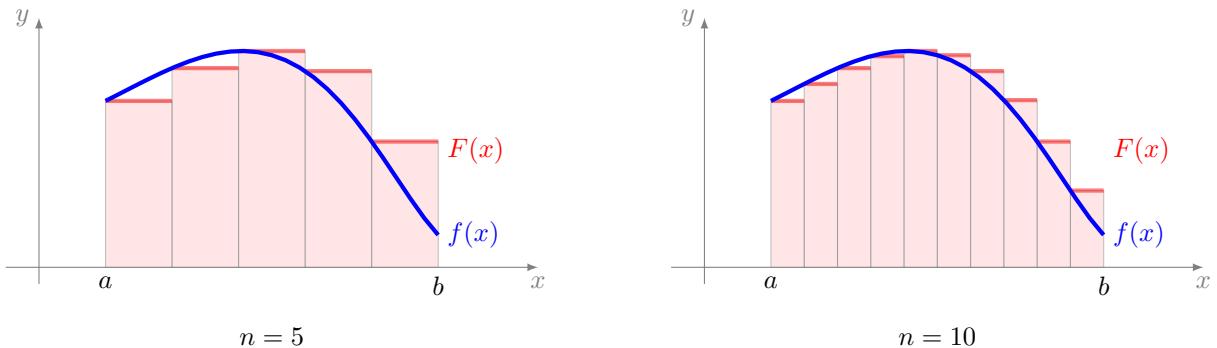
Plusieurs commentaires importants.

- Tout d'abord rappelons que nous réalisons nos neurones avec des fonctions d'activation  $H$  (marche de Heaviside) sauf pour le neurone de sortie qui a pour fonction d'activation l'identité.
- Les hypothèses sont importantes :  $f$  est continue et définie sur un intervalle fermé et borné. Si une des hypothèses venait à manquer l'énoncé serait faux.
- Bien que l'énoncé ne le dise pas, on peut concrètement réaliser à la main un réseau qui approche la fonction  $f$  (voir le chapitre suivant). Cependant, ce n'est pas l'esprit des réseaux de neurones.
- Que signifie « approcher d'aussi près que l'on veut la fonction  $f$  » ? C'est dire que pour chaque  $\epsilon > 0$ , il existe une fonction  $F$  (ici issue d'un réseau de neurones), telle que :

$$\text{pour tout } x \in [a, b] \quad |f(x) - F(x)| < \epsilon.$$

C'est l' **approximation uniforme** des fonctions.

*Démonstration.* La preuve est simple : toute fonction continue sur un intervalle  $[a, b]$  peut être approchée d'aussi près que l'on veut par une fonction en escalier. Par exemple, on subdivise l'intervalle  $[a, b]$  en  $n$  sous-intervalles  $[x_i, x_{i+1}]$  et on prend une marche de hauteur  $f(x_i)$  sur cet intervalle. Comme nous avons prouvé que l'on sait réaliser toutes les fonctions en escalier, la preuve est terminée.



Remarque : la preuve se rapproche de la construction de l'intégrale. Pour calculer l'intégrale, on calcule en fait l'aire de rectangles. Ces rectangles correspondent à nos fonctions en escalier.  $\square$

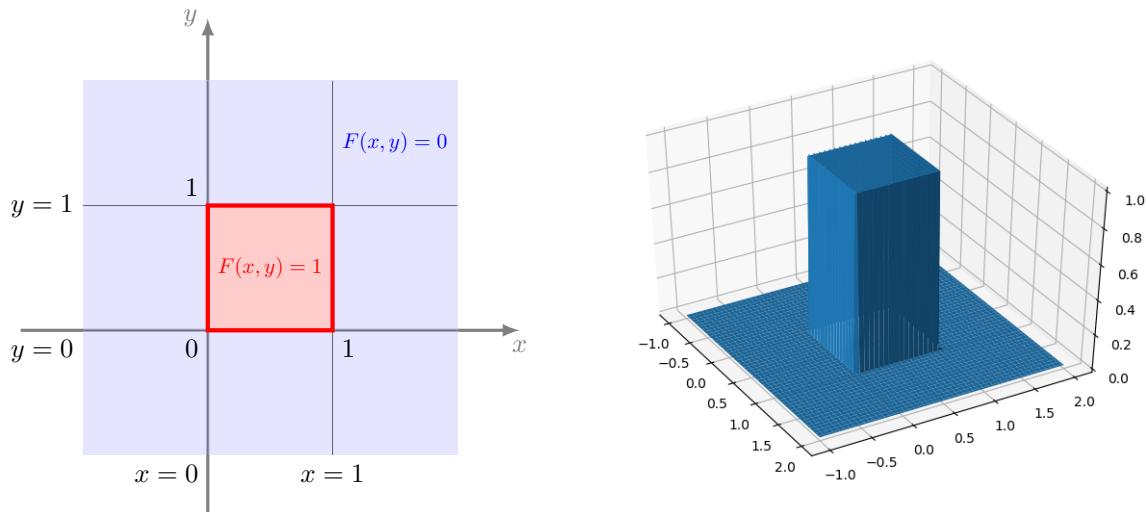
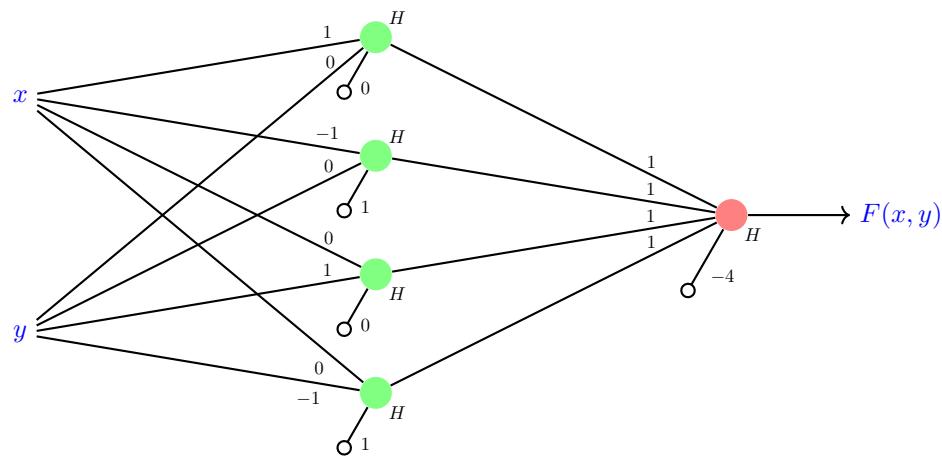
## 5. Théorème d'approximation universelle (deux variables et plus)

Ce que nous avons fait pour une variable, nous pouvons le faire pour deux variables (ou plus).

### Théorème 2.3 : Théorème d'approximation universelle

Toute fonction continue  $f : [a, b] \times [a, b] \rightarrow \mathbb{R}$  peut être approchée d'autant près que l'on veut par une fonction  $F : [a, b] \times [a, b] \rightarrow \mathbb{R}$  réalisée par un réseau de neurones.

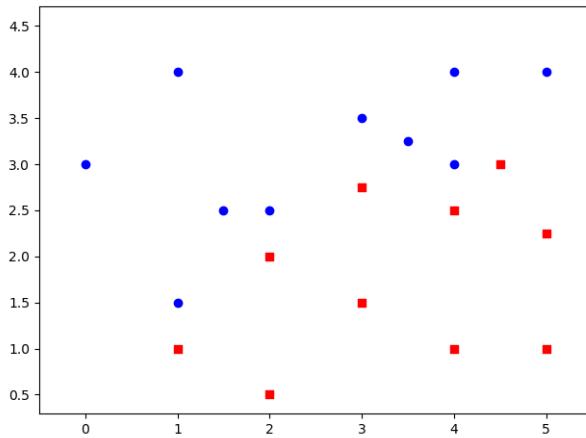
Il suffit là encore de réaliser des fonctions marches élémentaires. Nous ne donnons pas de détails mais seulement l'exemple d'un réseau qui réalise la fonction  $F : \mathbb{R}^2 \rightarrow \mathbb{R}$  qui vaut 1 sur  $[0, 1] \times [0, 1]$  et 0 partout ailleurs.



## VI- Principe de la rétropropagation

La rétropropagation, c'est la descente de gradient appliquée aux réseaux de neurones. Nous allons étudier des problèmes variés et analyser les solutions produites par des réseaux de neurones.

Voici un jeu de données : des cercles bleus et des carrés rouges. Nous souhaitons trouver un modèle simple qui caractérise ces données.

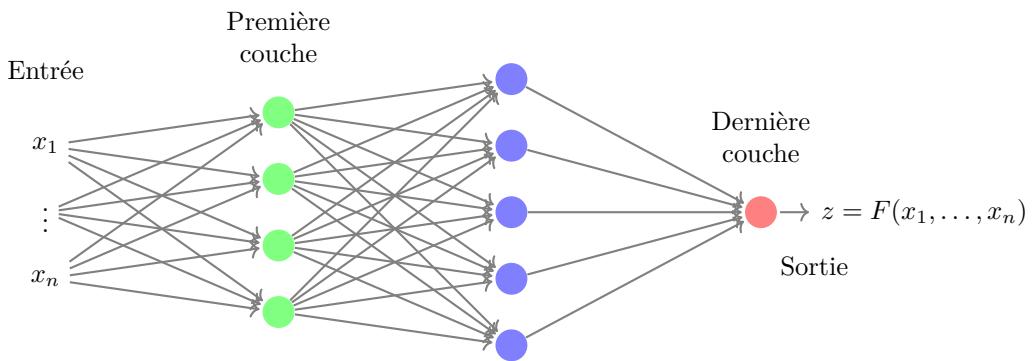


Plus exactement, nous souhaiterions pouvoir dire pour chaque point du plan s'il devrait être colorié en rouge ou bien en bleu et ceci pour des points qui ne sont pas dans les données de départ. De plus, nous voulons ne rien faire à la main, mais que la réponse soit calculée par la machine !

Nous allons revoir pas à pas l'utilisation d'un réseau de neurones pour résoudre un problème et traiterons en particulier l'exemple ci-dessus.

## 1. Objectif du réseau

- Soit  $\mathcal{R}$  un réseau de neurones. Celui-ci est défini par son architecture (le nombre de couches, le nombre de neurones par couche), les fonctions d'activation et l'ensemble  $P = (a_1, a_2, \dots)$  des poids de tous les neurones.
- À ce réseau  $\mathcal{R}$  on associe une fonction  $F : \mathbb{R}^n \rightarrow \mathbb{R}^p$  où  $n$  est la dimension des entrées (de la première couche) et  $p$  le nombre de neurones de la couche de sortie. Dans ce chapitre, nous supposerons qu'il n'y a qu'une seule sortie, c'est-à-dire  $p = 1$  et  $F : \mathbb{R}^n \rightarrow \mathbb{R}$ .



- On dispose de données  $(X_i, z_i)$  (pour  $i = 1, \dots, N$ ) où  $X_i \in \mathbb{R}^n$  est une **entrée** (de la forme  $X = (x_1, \dots, x_n)$ ) et  $z_i \in \mathbb{R}$  est la **sortie attendue** pour cette entrée.
- Le but est de trouver les poids du réseau afin que la fonction  $F$  qui lui est associée vérifie :

$$F(X_i) \simeq z_i \quad \text{pour tout } i = 1, \dots, N.$$

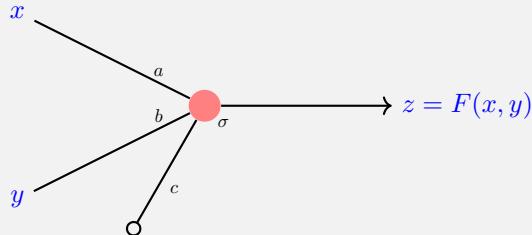
- Pour mesurer précisément la performance de l'approximation, on définit une **fonction erreur** :

$$E = \frac{1}{N} \sum_{i=1}^N E_i \quad \text{avec} \quad E_i = (F(X_i) - z_i)^2.$$

### Exemple 1.1

Nous traitons l'exemple donné en introduction.

- On décide de construire le réseau le plus simple possible : avec un seul neurone. Cela correspond à séparer les points du plan selon une droite. On choisit la fonction d'activation  $\sigma$ . La dimension de l'entrée est 2 et celle la sortie est 1. Il y a 3 poids ( $a, b, c$ ) à calculer pour terminer le paramétrage du réseau.



- Pour chaque triplet de poids  $(a, b, c)$  notre réseau définit une fonction  $F : \mathbb{R}^2 \rightarrow \mathbb{R}$  qui est en fait ici :

$$F(x, y) = \sigma(ax + by + c)$$

$$\text{avec } \sigma(t) = \frac{1}{1+e^{-t}}.$$

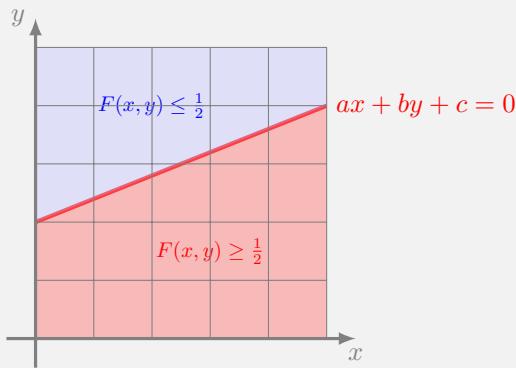
- Nos données sont les points rouges et bleus. Une entrée  $X_i$  est donc constituée des coordonnées  $(x_i, y_i)$  d'un point et la sortie attendue pour ce point est  $z_i = 0$  (pour les points bleus) et  $z_i = 1$  (pour les points rouges). Voici les coordonnées des ronds bleus (sortie attendue  $z_i = 0$ ) :

$$(0, 3), (1, 1.5), (1, 4), (1.5, 2.5), (2, 2.5), (3, 3.5), (3.5, 3.25), (4, 3), (4, 4), (5, 4)$$

et des carrés rouges (sortie attendue  $z_i = 1$ ) :

$$(1, 1), (2, 0.5), (2, 2), (3, 1.5), (3, 2.75), (4, 1), (4, 2.5), (4.5, 3), (5, 1), (5, 2.25).$$

- Comment définir les poids  $(a, b, c)$  afin que  $F(x_i, y_i) \simeq 0$  pour tous les ronds bleus et que  $F(x_i, y_i) \simeq 1$  pour tous les carrés rouges ? Si on sait trouver de tels poids alors on pourra colorier (presque) n'importe quel point  $(x, y)$  du plan (et pas seulement nos ronds et nos carrés) : si  $F(x, y) \simeq 0$ , on coloriera le point en bleu, si par contre  $F(x, y) \simeq 1$ , on le coloriera en rouge. Plus précisément, la fonction  $F$  (qui dépend de  $\sigma$ ) a ses valeurs dans  $[0, 1]$  et prend la valeur  $F(x, y) = \frac{1}{2}$  exactement sur la droite  $ax + by + c = 0$ . Ainsi, la fonction  $F$  sépare le plan en deux demi-plans  $\{(x, y) \mid F(x, y) \leq \frac{1}{2}\}$  et  $\{(x, y) \mid F(x, y) \geq \frac{1}{2}\}$  le long de la droite  $ax + by + c = 0$ .



- L'erreur commise par la fonction  $F$  associée aux poids  $a, b, c$  est :

$$E = E(a, b, c) = \frac{1}{N} \sum_{i=1}^N E_i(a, b, c)$$

où  $N$  est le nombre total des données (le nombre de ronds bleus plus le nombre de carrés rouges) et

$$E_i(a, b, c) = (F(x_i, y_i) - z_i)^2.$$

On peut détailler un peu plus pour chaque type de point. En effet, pour un rond bleu la sortie attendue est  $z_i = 0$  donc  $E_i(a, b, c) = (\sigma(ax_i + by_i + c) - 0)^2$ , alors que pour un carré rouge la sortie attendue est  $z_i = 1$  donc  $E_i(a, b, c) = (\sigma(ax_i + by_i + c) - 1)^2$ .

## 2. Descente de gradient

- Pour trouver les poids  $P = (a_1, a_2, \dots)$  qui définissent le meilleur réseau  $\mathcal{R}$  (autrement dit la meilleure fonction  $F$ ), il suffit de minimiser l'erreur  $E$ , vue comme une fonction des poids  $P = (a_1, a_2, \dots)$ . Pour cela on utilise la méthode de la descente de gradient.
- On part de poids initiaux  $P_0 = (a_1, a_2, \dots)$ , par exemple choisis au hasard. On fixe un pas  $\delta$ .
- On construit par itérations des poids  $P_k$  selon la formule de récurrence :

$$P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} E(P_k).$$

À chaque itération, l'erreur  $E(P_k)$  diminue. On s'arrête au bout d'un nombre d'itérations fixé à l'avance.

- Pour calculer le gradient  $\overrightarrow{\text{grad}} E = \frac{1}{N} \sum_{i=1}^N \overrightarrow{\text{grad}} E_i$ , il faut calculer chacun des  $\overrightarrow{\text{grad}} E_i$ , c'est-à-dire les dérivées partielles par rapport à chacun des poids  $a_j$  selon la formule :

$$\frac{\partial E_i}{\partial a_j}(X_i) = 2 \frac{\partial F}{\partial a_j}(X_i)(F(X_i) - z_i).$$

### Exemple 2.1

Poursuivons l'étude de notre exemple.

- La fonction  $E(a, b, c)$  dépend des poids  $a, b, c$ .
- On part de poids initiaux  $P_0 = (a_0, b_0, c_0)$ , par exemple  $P_0 = (0, 1, -2)$  qui correspond à séparer le plan selon la droite horizontale  $y = 2$ . On fixe le pas  $\delta = 1$ .
- On calcule l'erreur locale pour la donnée numéro  $i$  :

$$E_i(a, b, c) = (F(x_i, y_i) - z_i)^2 = (\sigma(ax_i + by_i + c) - z_i)^2$$

avec  $z_i = 0$  ou  $z_i = 1$ .

- Comme  $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ , alors, en notant  $\sigma_i = \sigma(ax_i + by_i + c)$ , on a :

$$\begin{aligned} \frac{\partial E_i}{\partial a}(a, b, c) &= 2x_i \sigma_i(1 - \sigma_i)(\sigma_i - z_i) \\ \frac{\partial E_i}{\partial b}(a, b, c) &= 2y_i \sigma_i(1 - \sigma_i)(\sigma_i - z_i) \\ \frac{\partial E_i}{\partial c}(a, b, c) &= 2\sigma_i(1 - \sigma_i)(\sigma_i - z_i) \end{aligned}$$

et par suite

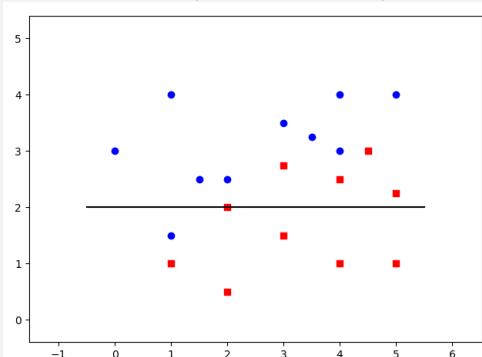
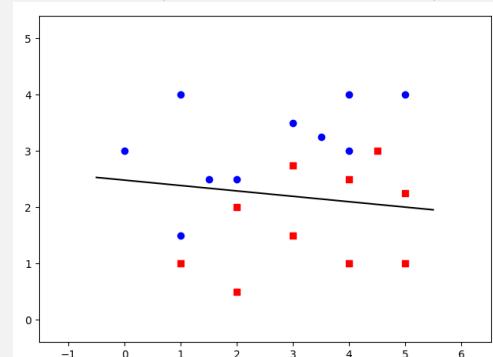
$$\overrightarrow{\text{grad}} E_i(a, b, c) = \left( \frac{\partial E_i}{\partial a}(a, b, c), \frac{\partial E_i}{\partial b}(a, b, c), \frac{\partial E_i}{\partial c}(a, b, c) \right) \quad \text{et} \quad \overrightarrow{\text{grad}} E(a, b, c) = \frac{1}{N} \sum_{i=1}^N \overrightarrow{\text{grad}} E_i.$$

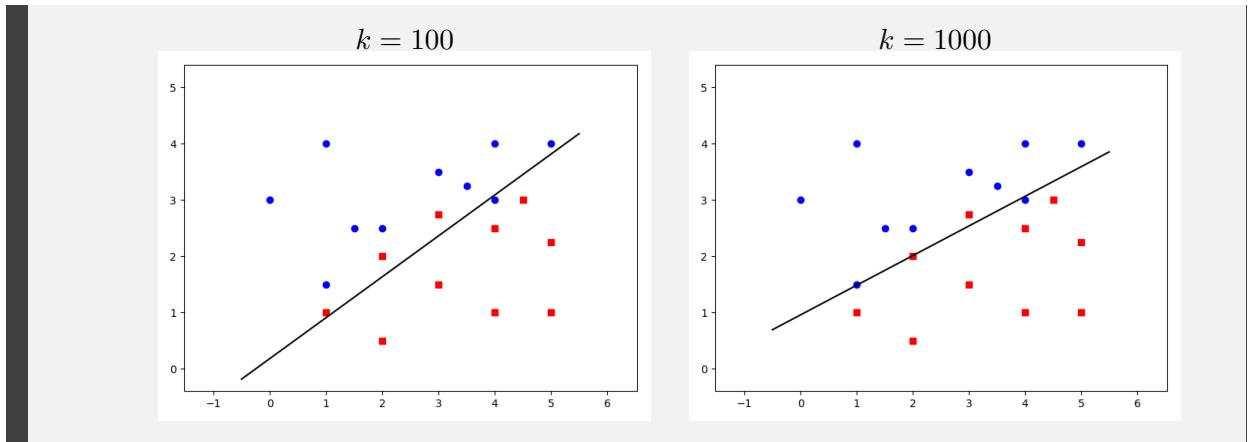
- On part de  $P_0 = (0, 1, -2)$ , on calcule  $\overrightarrow{\text{grad}} E(P_0) = (-0.077, 0.192, 0.005)$ . On obtient le poids suivant par  $P_1 = P_0 - \delta \overrightarrow{\text{grad}} E(P_0) = (0.077, 0.807, -2.005)$  (avec  $\delta = 1$ ).
- On continue avec la formule de récurrence  $P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} E(P_k)$  pour obtenir les poids suivants :

$k$	$P_k = (a_k, b_k, c_k)$	$\overrightarrow{\text{grad}} E(a_k, b_k, c_k)$	$E(a_k, b_k, c_k)$
0	(0, 1, -2)	(-0.077, 0.192, 0.005)	0.450
1	(0.077, 0.807, -2.005)	(-0.072, 0.213, 0.0069)	0.404
2	(0.149, 0.593, -2.012)	(-0.094, 0.188, -0.0062)	0.355
3	(0.244, 0.304, -2.006)	(-0.120, 0.137, -0.0237)	0.313
4	(0.364, 0.267, -1.982)	(-0.090, 0.126, -0.0240)	0.282
5	(0.455, 0.140, -1.958)	(-0.073, 0.109, -0.0263)	0.259
6	(0.528, 0.031, -1.932)	(-0.059, 0.095, -0.0278)	0.242
7	(0.588, -0.063, -1.904)	(-0.051, 0.083, -0.0290)	0.229
8	(0.639, -0.146, -1.875)	(-0.044, 0.074, -0.0297)	0.219
9	(0.684, -0.221, -1.845)	(-0.040, 0.067, -0.0302)	0.211
10	(0.72, -0.288, -1.815)	(-0.036, 0.061, -0.0305)	0.205
...			
100	(1.328, -1.828, 0.333)	(-0.00037, 0.00646, -0.01780)	0.103
...			
1000	(2.032, -3.860, 3.703)	(-0.00034, 0.00073, -0.00086)	0.071

Au bout de  $k = 1000$  itérations, on obtient les poids  $P_{1000} = (2.032, -3.860, 3.703)$ . Le gradient est devenu très petit, ce qui signifie ici que l'on est proche d'un minimum. De plus, l'erreur ne diminue presque plus lors des itérations suivantes, nous avons atteint le minimum recherché. (Attention, l'erreur est très petite, mais ne tend pas vers 0.)

- La droite  $ax + by + c = 0$  qui sépare le plan en deux demi-plans  $\{(x, y) \mid F(x, y) \leq \frac{1}{2}\}$  et  $\{(x, y) \mid F(x, y) \geq \frac{1}{2}\}$  évolue à chaque itération pour finir par séparer le mieux possible les ronds des carrés.

 $k = 0$  (initialisation) $k = 1$  (première itération)



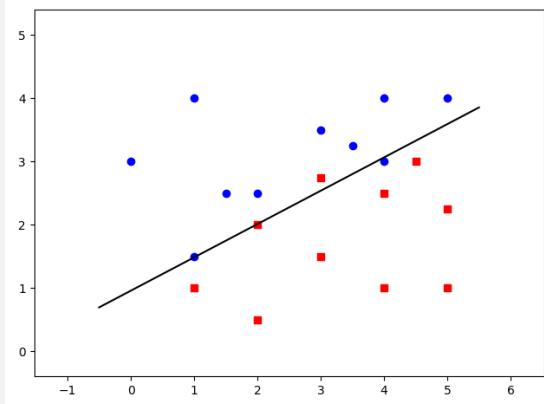
### 3. Prédiction

La conception d'un réseau de neurones est réalisée en modélisant au mieux les données injectées. Mais l'objectif réel est de faire des prédictions pour de nouvelles valeurs, jamais rencontrées auparavant. La descente de gradient produit un ensemble de poids  $P$  qui définit complètement notre réseau  $\mathcal{R}$ . Nous obtenons donc une fonction  $F : \mathbb{R}^n \rightarrow \mathbb{R}^p$ , construite de sorte que  $F(X_i) \simeq z_i$ . Nous pouvons évaluer cette fonction pour tout  $X \in \mathbb{R}^n$ , même pour des  $X$  différents des  $X_i$ .

#### Exemple 3.1

Dans notre exemple, nous avons obtenu par la descente de gradient les poids  $P_{1000} = (a, b, c) = (2.032, -3.860, 3.703)$ . Notre neurone  $\mathcal{R}$  est maintenant opérationnel et définit ici la fonction

$$F(x, y) = \sigma(ax + by + c).$$



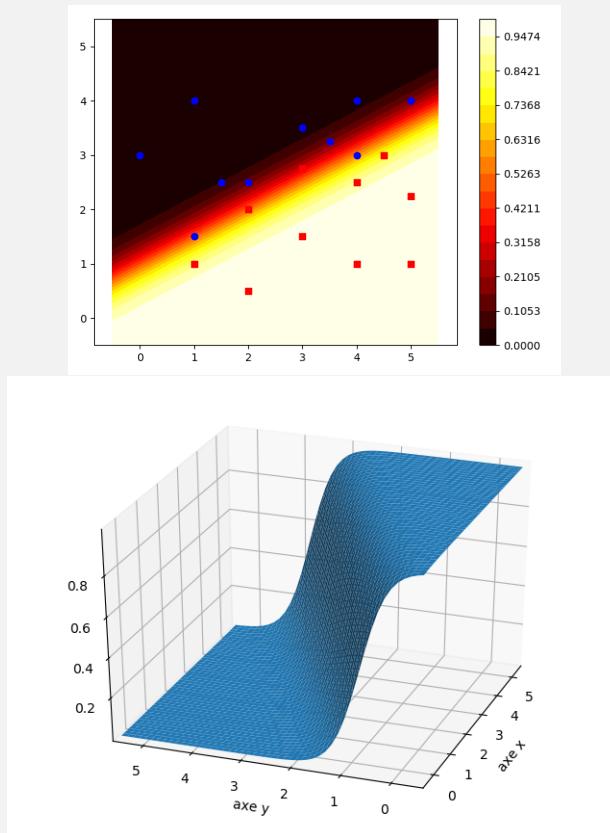
On souhaitait avoir  $F(x_i, y_i) = 0$  pour les ronds bleus et  $F(x_i, y_i) = 1$  pour les carrés rouges. Dans la pratique, on obtient des valeurs approchées, par exemple  $F(0, 3) = 0.0003$  pour le rond bleu en  $(0, 3)$  et  $F(1, 1) = 0.86$  pour le carré rouge en  $(1, 1)$ .

Notre fonction  $F$  ne modélise pas parfaitement toutes nos données. Par exemple, pour le rond bleu en  $(4, 3)$  on a  $F(4, 3) = 0.56$  alors qu'on voudrait une valeur proche de 0 et de même pour le carré rouge en  $(3, 2.75)$  on a  $F(3, 2.75) = 0.30$  alors qu'on voudrait une valeur proche de 1.

Mais l'intérêt principal de  $F$  est d'être définie pour tous les points du plan, ceci permet d'attribuer une couleur à chaque point  $(x, y) \in \mathbb{R}^2$  selon la convention : bleu si  $F(x, y) \simeq 0$ , rouge si  $F(x, y) \simeq 1$ . Il y a bien sûr une « zone grise » entre les zones rouge et bleue.

Par exemple,  $F(2, 3) = 0.02$  donc  $(2, 3)$  mérite d'être colorié en bleu,  $F(2, 1) = 0.98$  donc  $(2, 1)$  mérite d'être colorié en rouge. La frontière en laquelle  $F$  vaut  $\frac{1}{2}$  est la droite d'équation  $ax + by + c = 0$ .

Voici les niveaux de  $F$  dans le plan (à gauche) et le graphe de  $F$  dans l'espace (à droite).

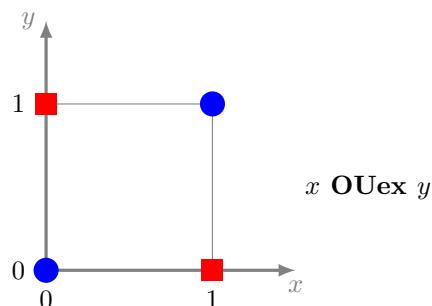


Nous avons choisi un réseau avec un seul neurone, la fonction  $F$  est donc nécessairement très simple (quels que soient les poids) et ne peut pas « coller » parfaitement aux données. Nous avons séparé au mieux les ronds bleus des carrés rouges par une droite. Avec un réseau plus complexe, et donc une frontière plus compliquée, nous aurions pu « coller » parfaitement aux données. Cependant est-ce vraiment ce que nous souhaitons faire ? Pour les données de notre exemple, avoir une séparation par une droite semble raisonnable. Peut-être que les points transfuges sont des erreurs de mesure.

## VII- Exemples

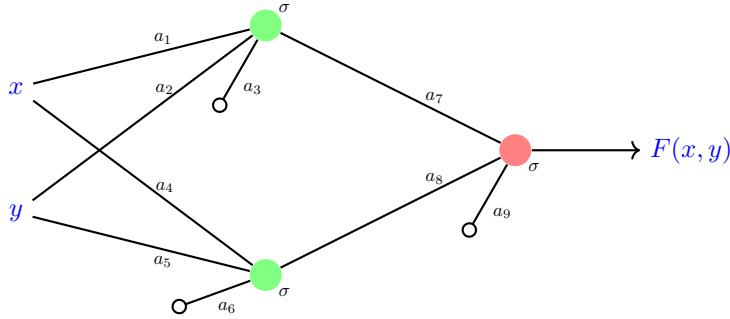
### 1. Le « ou exclusif »

Rappelons le problème du « ou exclusif » : il s'agit de trouver un réseau de neurones dont la fonction associée  $F$  vaut 1 en  $(0, 1)$  et  $(1, 0)$  (les carrés rouges) et vaut 0 en  $(0, 0)$  et  $(1, 1)$  (les rond bleus).



**Réseau.**

On a vu qu'il existe une solution avec un réseau à deux couches de fonction d'activation la fonction de Heaviside (voir le chapitre « Réseau de neurones »). On cherche à l'aide de la machine quels poids pourraient convenir avec la fonction d'activation sigmoïde :



Il y a donc 9 coefficients à déterminer. La fonction d'erreur est :

$$E = (F(0, 0) - 0)^2 + (F(1, 1) - 0)^2 + (F(0, 1) - 1)^2 + (F(1, 0) - 1)^2$$

sachant que  $F$  et donc l'erreur  $E$  dépendent des coefficients  $a_1, \dots, a_9$ . Il s'agit de trouver ceux qui rendent l'erreur  $E$  minimale.

**Descente de gradient.** On applique la méthode de la descente de gradient à la fonction  $E$  pour les variables  $(a_1, \dots, a_9)$ . On utilise la descente de gradient classique avec un pas  $\delta = 1$ . Le choix des poids initiaux est déterminant pour les résultats. On choisit comme poids de départ :

$$P_0 = (a_1, \dots, a_9) = (1.0, 2.0, -3.0, -3.0, -2.0, 1.0, 1.0, 1.0, -1.0).$$

L'erreur initiale vaut  $E(P_0) = 0.2961$ . Le gradient  $\overrightarrow{\text{grad}} E(P_0)$  vaut

$$(0.00349, -0.00303, -0.00832, -0.00696, -0.01350, -0.01047, -0.00324, 0.01258, -0.04671).$$

Avec  $\delta = 1$ , la première itération modifie un petit peu les coefficients :

$$P_1 = (0.9965, 2.0030, -2.9916, -2.9930, -1.9864, 1.0104, 1.0032, 0.9874, -0.9532),$$

l'erreur a légèrement diminué et vaut maintenant  $E(P_1) = 0.2935$ . La seconde itération donne :

$$P_2 = (0.9925, 2.0055, -2.9839, -2.9860, -1.9731, 1.0206, 1.0053, 0.9738, -0.9105)$$

et l'erreur vaut maintenant  $E(P_2) = 0.2911$ .

Au bout de 1000 itérations :

$$P_{1000} = (3.5623, 3.5675, -5.5344, -5.3682, -5.3935, 1.8403, -6.3234, -6.3801, 3.1216)$$

et l'erreur vaut maintenant  $E(P_{1000}) = 0.0097$ .

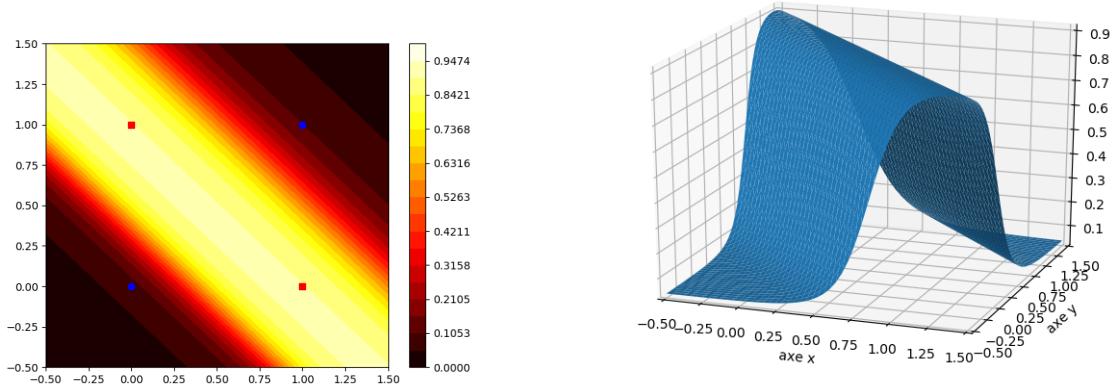
**Validation et prédition.** Au bout de 1000 itérations notre réseau de neurones est assez performant. La fonction  $F(x, y)$  obtenue vérifie :

$$F(0, 0) = 0.08, \quad F(1, 1) = 0.10, \quad F(1, 0) = 0.89, \quad F(0, 1) = 0.89.$$

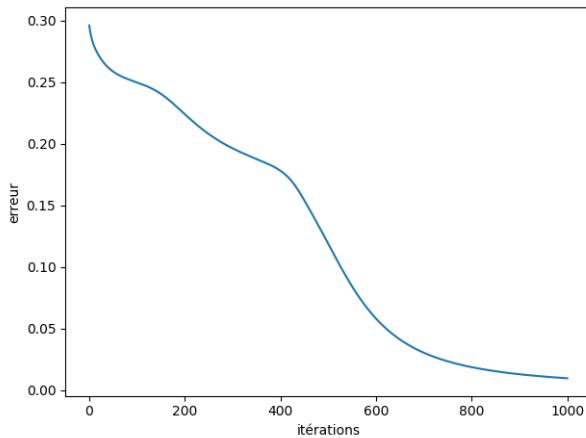
Les ronds bleus sont donc clairement séparés des carrés rouges par les valeurs de  $F$ .

Quelle couleur est-il naturel d'associer au points  $(0.2, 0.9)$ ? On calcule  $F(0.2, 0.9) = 0.87$  qui est assez proche de 1, il est donc naturel de le colorier en rouge.

**Visualisation graphique.** Voici deux représentations graphiques de la fonction  $F$  obtenue. À gauche par les lignes de niveau dans le plan et à droite par son graphe dans l'espace. La fonction  $F$  prend des valeurs proches de 1 autour de la droite passant par  $(1, 0)$  et  $(0, 1)$  et se rapproche de 0 partout ailleurs.



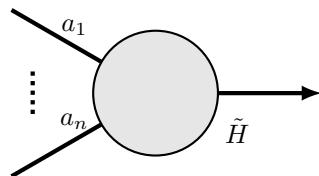
**Analyse.** Voici l'évolution de l'erreur en fonction du nombre d'itérations. L'erreur décroît au fil des itérations : c'est le principe de la descente de gradient !



Comparer les poids obtenus avec ceux du « ou exclusif » définis dans le chapitre « Réseau de neurones ».

## 2. Le perceptron et la règle de Hebb

Nous revenons sur l'exemple historique du cas d'un seul neurone (sans biais). Que donne la rétropropagation dans ce cas très simple ?



Nous souhaitons une fonction d'activation du type marche Heaviside, qui renvoie 0 ou 1, mais pour la descente du gradient nous avons besoin d'une dérivée non nulle. Nous allons imaginer qu'il existe une fonction marche de Heaviside virtuelle  $\tilde{H}$  telle que

$$\tilde{H}(x) = 0 \quad \text{si } x < 0 \quad \tilde{H}(x) = 1 \quad \text{si } x \geq 0 \quad \text{et} \quad \tilde{H}'(x) = 1 \quad \text{pour tout } x \in \mathbb{R}.$$

Il est clair qu'une telle fonction n'existe pas (car la dérivée d'une fonction constante par morceaux est toujours nulle) mais faisons comme si c'était le cas.

Ce neurone définit une fonction  $F$  telle que  $F(x_1, \dots, x_n) = \tilde{H}(a_1x_1 + \dots + a_nx_n)$ . Comme d'habitude, nous avons des données  $(X_i, z_i)$ , ici  $z_i = 0$  ou bien  $z_i = 1$ . Il s'agit de trouver les poids  $(a_1, \dots, a_n)$  tels que  $F(X_i) \simeq z_i$ . L'erreur locale est donnée par  $E_i = (F(X_i) - z_i)^2$ .

La descente de gradient (stochastique) pour la donnée  $i$  s'écrit :

$$P_{k+1} = P_k - \delta \overrightarrow{\text{grad}} E_i(P_k).$$

On a :

$$\frac{\partial E_i}{\partial a_j}(X_i) = 2 \frac{\partial F}{\partial a_j}(X_i)(F(X_i) - z_i).$$

Mais  $\frac{\partial F}{\partial a_j}(X_i) = x_j$  car  $\tilde{H}'(x) = 1$  et comme  $F(X_i)$  et  $z_i$  valent 0 ou 1, alors  $\frac{\partial E_i}{\partial a_j}(X_i)$  vaut  $\pm 2x_j$  ou 0. Autrement dit,  $\overrightarrow{\text{grad}} E_i = 2\epsilon(x_1, \dots, x_n) = 2\epsilon X_i$  avec  $\epsilon = 0, 1$  ou  $-1$ .

Nous avons ainsi obtenu la **règle de Hebb**, qui est en fait l'ancêtre de la rétropropagation.

### Règle de Hebb.

- On fixe un pas  $\delta > 0$ .
- On part d'un poids  $P_0$  (choisi au hasard par exemple).
- On calcule par récurrence les poids  $P_k$  en parcourant la liste  $(X_i, y_i)$  et en distinguant les cas suivants :
  - si la sortie prédite  $F(X_i)$  vaut la sortie attendue  $z_i$  alors on ne change rien :

$$P_{k+1} = P_k,$$

- si la sortie prédite  $F(X_i)$  vaut 1 alors que la sortie attendue  $z_i$  vaut 0 alors :

$$P_{k+1} = P_k - 2\delta X_i,$$

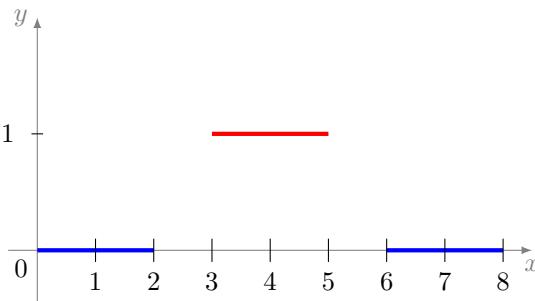
- si la sortie prédite  $F(X_i)$  vaut 0 alors que la sortie attendue  $z_i$  vaut 1 alors :

$$P_{k+1} = P_k + 2\delta X_i.$$

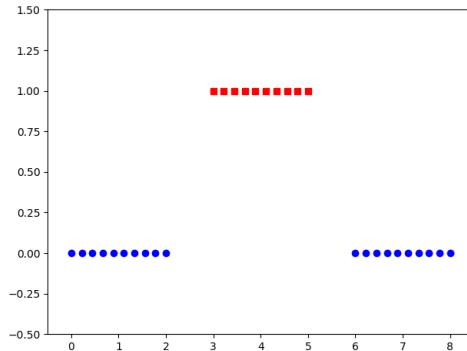
- Une fois toutes les données  $(X_i, z_i)$  utilisées, on recommence depuis la première donnée.
- On s'arrête au bout d'un nombre d'itérations fixé à l'avance.
- Le dernier poids obtenu  $P_k = (a_1, \dots, a_n)$  définit le perceptron et la fonction associée qui répond au problème est :  $F(x_1, \dots, x_n) = 1$  si  $a_1x_1 + \dots + a_nx_n \geq 0$  et  $F(x_1, \dots, x_n) = 0$  sinon.

### 3. Une fonction marche

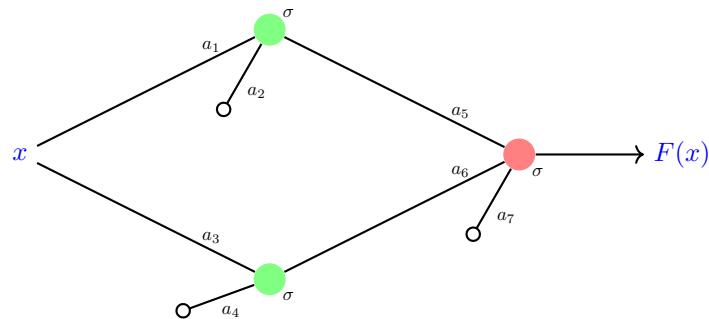
Nous avons vu dans le chapitre « Réseau de neurones » qu'il est facile de réaliser des fonctions « marche » à l'aide d'un réseau simple et de la fonction de Heaviside. Nous souhaitons construire un réseau de neurones dont la fonction  $x \mapsto F(x)$  réalise au mieux les contraintes suivantes :  $F(x)$  vaut 0 sur  $[0, 2]$ , puis vaut 1 sur  $[3, 5]$  et de nouveau vaut 0 sur  $[6, 8]$  (il y a une certaine liberté sur  $[2, 3]$  et  $[5, 6]$ ).



**Données.** On décide de placer 10 ronds bleus espacés régulièrement sur  $[0, 2]$ , la même chose sur  $[6, 8]$ , là où la fonction doit être nulle, et également 10 carrés rouges espacés régulièrement sur  $[3, 5]$  là où la fonction doit valoir 1. Ces points fournissent donc les  $N = 30$  données d'apprentissage.



**Réseau.** On décide d'utiliser un réseau avec deux neurones sur la couche d'entrée et un neurone sur la couche de sortie, tous utilisant la fonction d'activation  $\sigma$ .



Il y a 7 coefficients à déterminer. La fonction d'erreur  $E$  est la somme des  $(F(x_i) - 0)^2$  pour les  $x_i$  abscisses des ronds bleus et des  $(F(x_i) - 1)^2$  pour les  $x_i$  abscisses des carrés rouges.

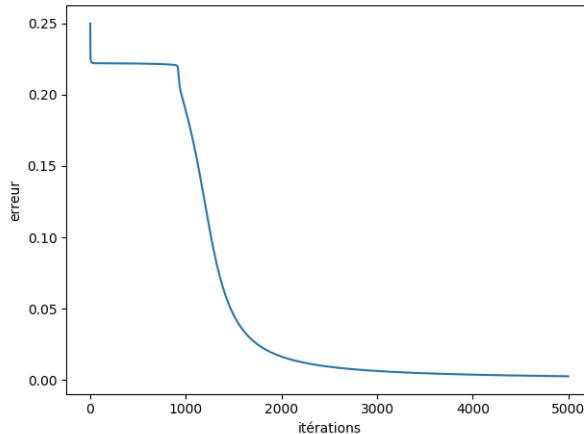
**Descente de gradient.** On applique la descente de gradient à la fonction  $E$  de variables  $(a_1, \dots, a_7)$  pour un pas  $\delta = 1$  et un choix arbitraire de poids initiaux :

$$P_0 = (a_1, \dots, a_7) = (0.0, 1.0, 0.0, -1.0, 1.0, 1.0, -1.0).$$

Au bout de 5000 itérations, on obtient les poids :

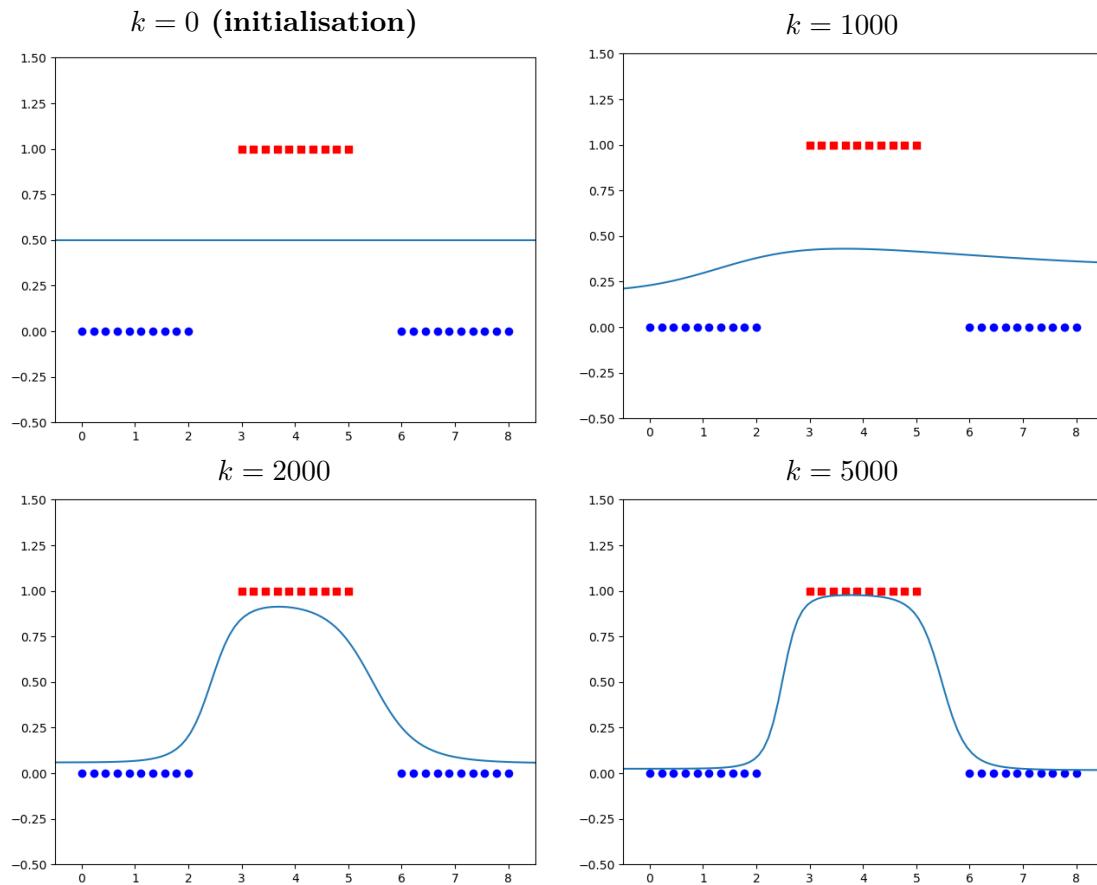
$$P_{5000} = (a_1, \dots, a_7) = (-2.0142, 10.9964, 3.0543, -7.7086, 8.2334, 7.8735, -11.9037).$$

Voici l'évolution de l'erreur en fonction du nombre d'itérations.



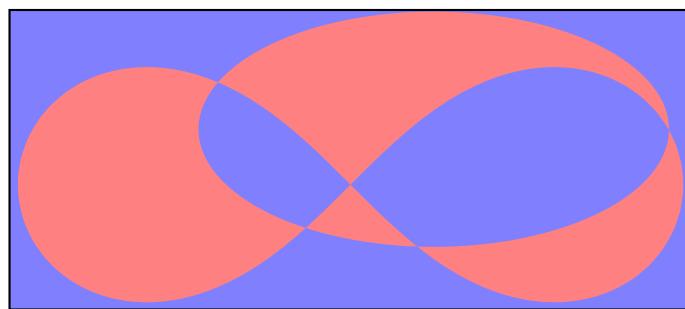
### Visualisation graphique.

Voici le graphe de la fonction  $x \mapsto F(x)$  au bout de différents nombres d'itérations.

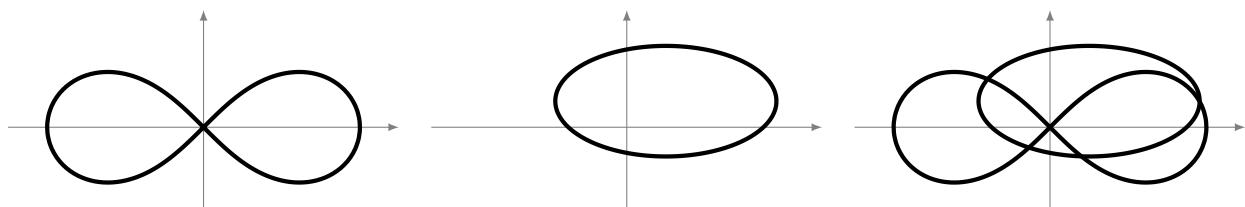


### 4. Plus de neurones

Le but est ici de trouver un réseau de neurones qui distingue deux zones compliquées du plan : la zone bleue et la zone rouge.



Voici comme sont construites ces zones : on part d'une lemniscate de Bernoulli d'équation  $(x^2 + y^2)^2 = 4(x^2 - y^2)$  et d'une ellipse d'équation  $(x - \frac{1}{2})^2 + 4(y - \frac{1}{3})^2 = 2$ .



L’union de ces deux courbes a pour équation :

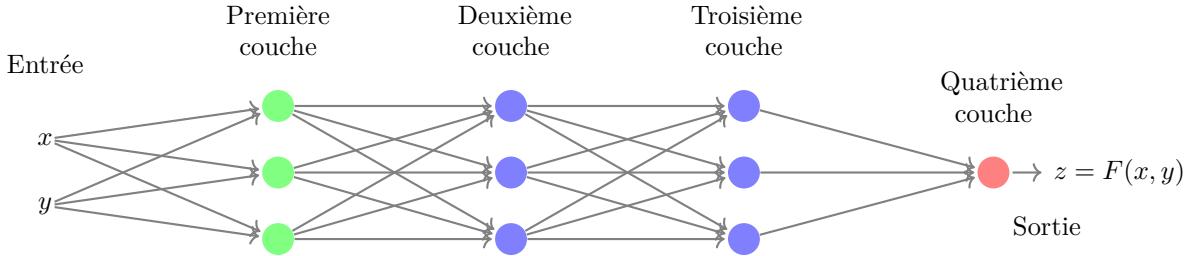
$$f(x, y) = ((x^2 + y^2)^2 - 4(x^2 - y^2)) \cdot ((x - \frac{1}{2})^2 + 4(y - \frac{1}{3})^2 - 2) = 0.$$

La zone rouge correspond aux points de coordonnées  $(x, y)$  pour lesquels  $f(x, y) \leq 0$  et la zone bleue à  $f(x, y) > 0$ . Nous allons limiter notre étude à une zone rectangulaire. Le rectangle est choisi de sorte que l’aire bleue et l’aire rouge soient à peu près égales.

**Objectifs.** On oublie maintenant la fonction  $f$  et on ne retient que quelques points rouges et quelques points bleus. On cherche un réseau et une fonction  $F$  telle que  $F(x, y) \simeq 1$  pour les points rouges et  $F(x, y) \simeq 0$  pour les points bleus.

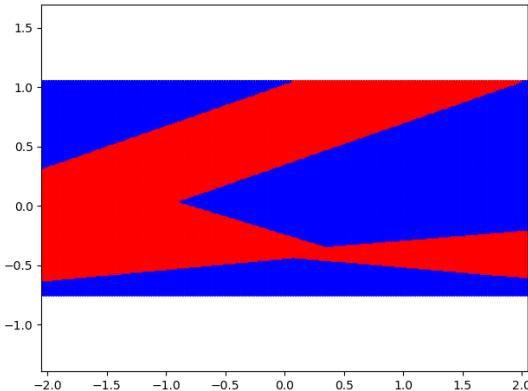
**Données.** On divise notre rectangle en une grille de  $n \times n$  points. Les exemples ci-dessous sont calculés pour  $n = 200$ . Nous avons donc 40 000 points  $(x, y)$ , repartis (à peu près) équitablement entre rouge ( $z = 1$ ) et bleu ( $z = 0$ ).

**Architecture.** On décide de concevoir un réseau à 4 couches, avec le même nombre  $p$  de neurones par couche pour les trois premières couches et un seul neurone sur la couche de sortie. La fonction d’activation choisie pour tous les neurones est ReLU. Voici une illustration de la configuration pour  $p = 3$ .

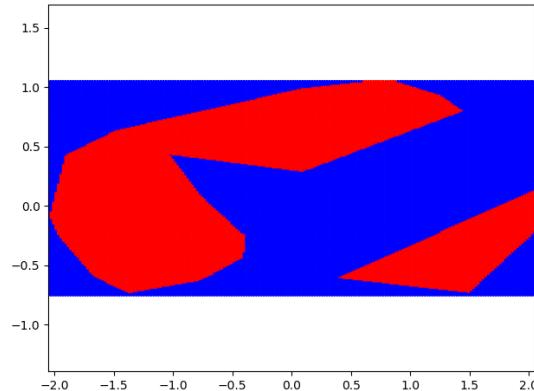


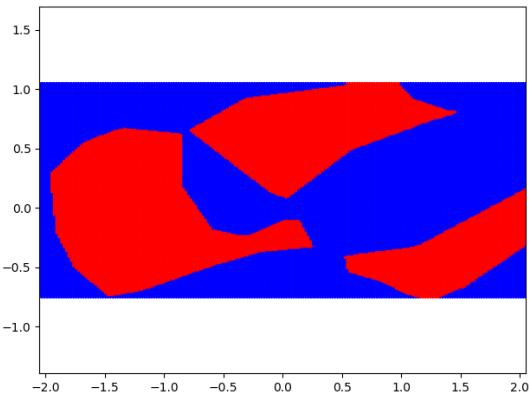
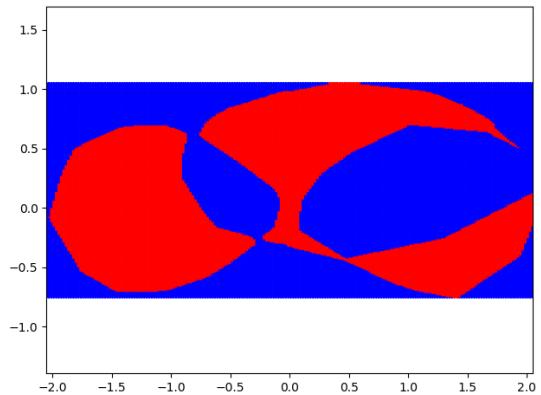
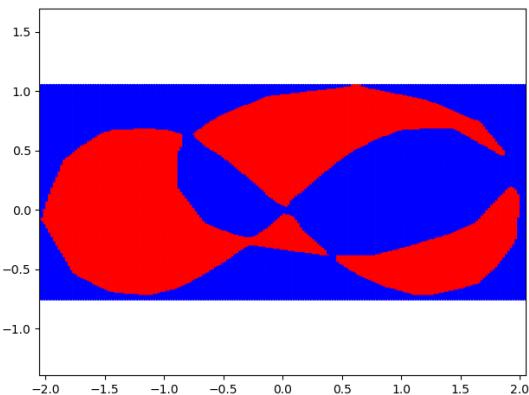
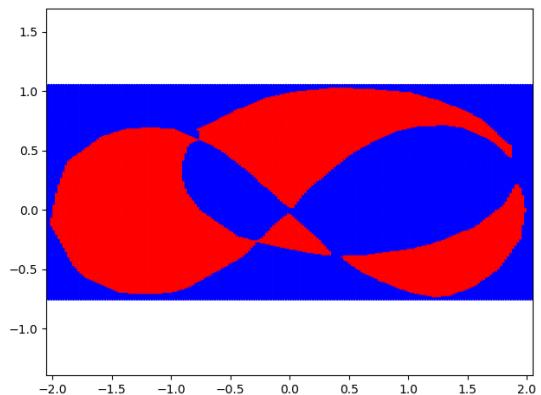
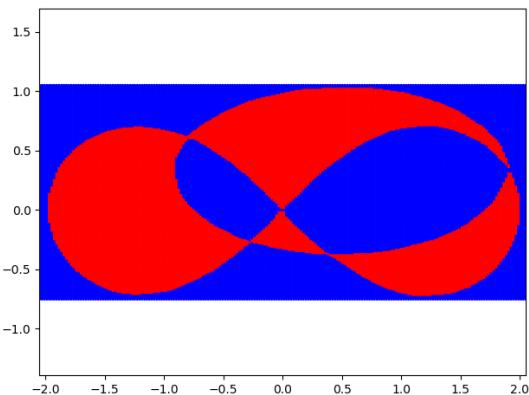
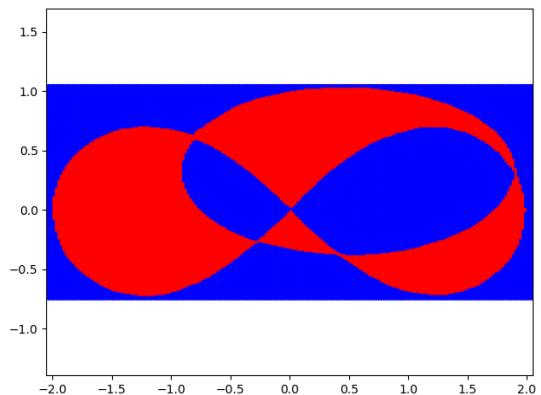
**Poids.** On calcule les poids avec une méthode de descente de gradient stochastique. Les poids initiaux sont choisis aléatoirement, le nombre d’itérations est suffisamment grand. Le réseau obtenu après calculs fournit donc une fonction  $F$ . On colorie en rouge les points pour lesquels  $F(x, y) \simeq 1$  (en fait, là où  $F(x, y) \geq \frac{1}{2}$ ) et en bleu les points pour lesquels  $F(x, y) \simeq 0$  (en fait, là où  $F(x, y) < \frac{1}{2}$ ). On regarde si le résultat obtenu est proche de la situation envisagée. On présente ici quelques résultats typiques intéressants (les résultats pouvant varier selon le choix aléatoire des poids initiaux).

$p = 3, 10$  neurones,  $37$  poids



$p = 5, 16$  neurones,  $81$  poids



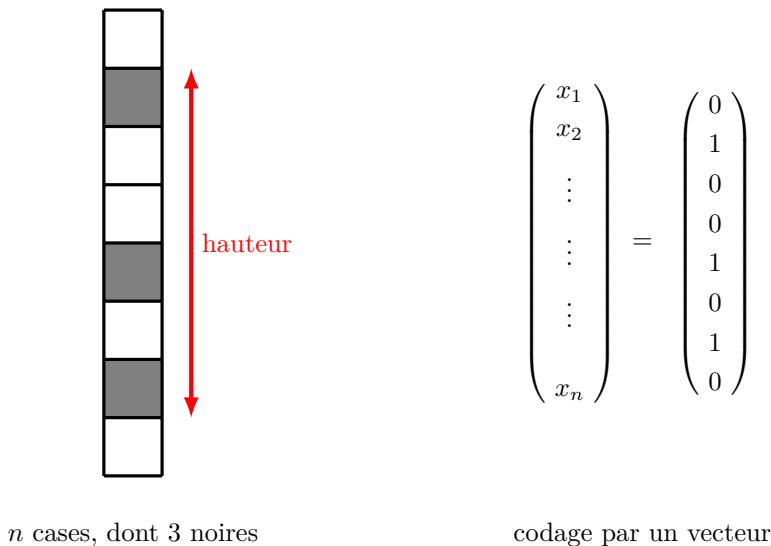
$p = 7, 22$  neurones, 141 poids $p = 10, 31$  neurones, 261 poids $p = 15, 46$  neurones, 541 poids $p = 20, 61$  neurones, 921 poids $p = 50, 151$  neurones, 5301 poids $p = 100, 301$  neurones, 20 601 poids

Conclusion : il faut un nombre assez grand de neurones pour pouvoir modéliser des phénomènes compliqués. Dans l'exemple traité ici, on obtient une bonne approximation à partir de  $p = 20$ , soit plus de 60 neurones et environ 1000 poids.

## 5. Apprentissage

N'oublions pas le but principal des réseaux de neurones : apprendre à partir des données afin de prédire des résultats pour des situations nouvelles.

Voici un problème pour lequel nous allons tester si les prédictions sur des données nouvelles sont correctes ou pas. On dispose de  $n$  cases, dont 3 sont noircies. On appelle *hauteur*, le nombre total de cases entre la plus haute et la plus basse case noircie. Si les trois cases ont pour rang  $i < j < k$  alors la hauteur vaut  $h = k - i + 1$  (la case noircie du milieu n'intervient pas dans la formule). Sur le dessin ci-dessous,  $n = 8$  et la configuration représentée est de hauteur 6.



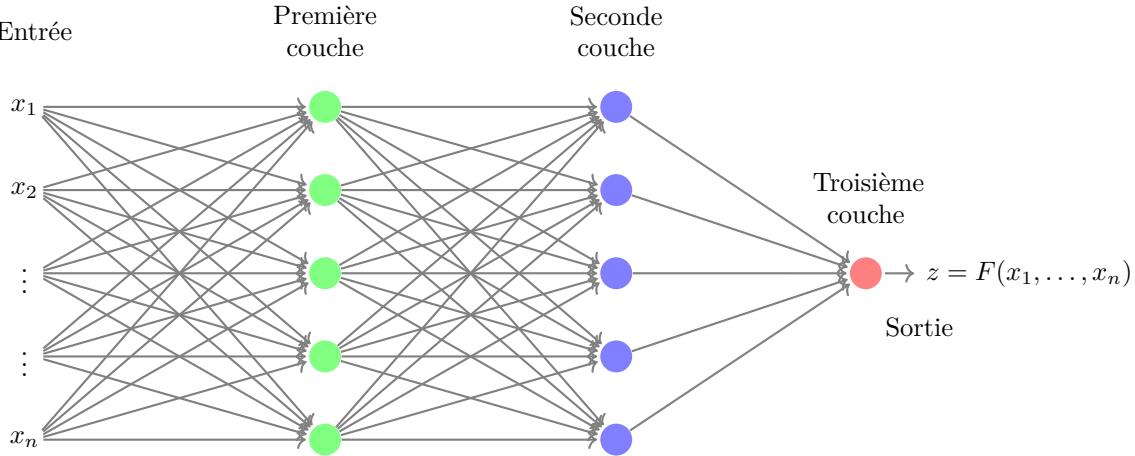
Nous voulons tester si un réseau de neurones permet de retrouver un résultat qu'on peut vérifier ici à l'aide d'une formule simple : étant donné une configuration il s'agit de faire calculer à la machine quelle est sa hauteur.

Il y a en tout  $N = \frac{n(n-1)(n-2)}{6}$  configurations possibles pour les trois cases noires parmi  $n$  cases. On décide d'utiliser la moitié des configurations comme données d'apprentissage et l'autre moitié va nous permettre de tester si le réseau calculé avec les données d'apprentissage se comporte « correctement » sur les nouvelles données.

Dans la suite nous prendrons  $n = 20$  cases, ce qui fait  $N = 1140$  configurations possibles.

**Données en entrées.** On calcule les  $N$  configurations possibles. Après un mélange aléatoire, on ne retient que  $N/2$  configurations pour l'apprentissage. Une configuration est codée sous la forme d'un vecteur  $X = (x_1, x_2, \dots, x_n)$  avec  $x_i = 0$  (case blanche) ou  $x_i = 1$  (case noire). Pour chaque configuration  $X$  des données d'apprentissage, on calcule la hauteur  $z = h(X)$ . Les données d'apprentissage sont les  $N/2$  données  $(X_k, z_k)$  formées des entrées  $X_k$  et de la sortie attendue  $z_k$  correspondant à la hauteur.

**Réseau.** Le réseau comporte  $n$  entrées : une entrée par case. Il est constitué de 3 couches. Les deux premières couches possèdent  $p$  neurones et la couche de sortie un seul. La fonction d'activation est la fonction ReLU pour tous les neurones. Sur la figure ci-dessous est illustré le cas  $p = 5$ .



**Apprentissage.** Le réseau est initialisé avec des poids aléatoires. Ensuite ces poids sont modifiés par une descente de gradient stochastique avec un nombre suffisant d’itérations.

**Sortie prédictive.** Le réseau paramétré fournit une fonction  $F$  à valeur réelle. La sortie attendue étant un entier, on décide que la sortie prédictive sera arrondie à l’entier le plus proche. Pour chaque  $X_k$ , la prédiction est correcte si la valeur arrondie  $F(X_k)$  vaut la hauteur  $h(X_k)$ .

**Tests.** Il faut tester l’efficacité du réseau : tout d’abord le réseau modélise-t-il correctement les données d’apprentissage ? En effet, même si la fonction  $F$  a été construite dans le but d’avoir  $F(X_k) \simeq z_k$ , il n’y a pas nécessairement égalité pour toutes les données d’apprentissage. Mais surtout, il faut tester si la fonction  $F$  donne de bonnes prédictions pour de nouvelles données. Nous disposons pour cela des  $N/2$  données de test non utilisées lors de l’apprentissage.

Les résultats dépendent des poids initiaux (aléatoires) et de la liste des configurations choisies pour l’apprentissage (liste qui a été mélangée), on ne retient que les meilleurs résultats parmi plusieurs essais. Par exemple, pour  $p = 10$ , voici un des meilleurs résultats obtenus :

- *Apprentissage.* 560 données prédictes correctement sur un total de 570 données d’apprentissage, soit 98% de réussite.
- *Test.* 506 données prédictes correctement sur un total de 570 données de test, soit 89% de réussite.

Voici quelques résultats pour différentes valeurs de  $p$ .

$p$	neurones	poids	pourcentage apprentissage	pourcentage test
3	7	79	50%	45%
5	11	141	85%	75%
7	15	211	90%	85%
10	21	331	98%	90%
15	36	571	99%	90%
20	41	861	100%	85%

**Commentaires.** Plus il y a de neurones, plus la fonction  $F$  modélise correctement les données d’apprentissage : à partir de 20 neurones ( $p \geq 10$ ), la modélisation est quasi-parfaite. Les pourcentages de réussite pour les données de test sont toujours inférieurs et plafonnent à 90%. Un phénomène nouveau apparaît avec plus de 40 neurones (pour  $p = 20$ ), les pourcentages de réussite sur les tests régressent par rapport à des réseaux ayant moins de neurones, bien que les données d’apprentissage soient parfaitement modélisées : c’est un phénomène de sur-apprentissage.

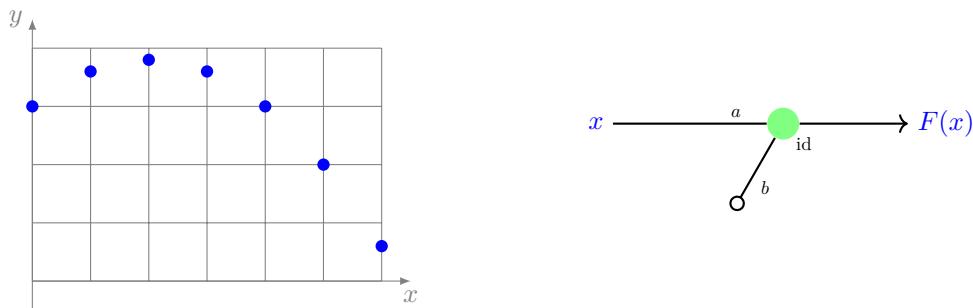
**VIII-****Sur-apprentissage et autres soucis**

Nous allons voir différents problèmes qui peuvent intervenir dans le calcul des poids d'un réseau de neurones, le problème le plus subtil étant le *sur-apprentissage*.

### 1. Modèle insuffisant

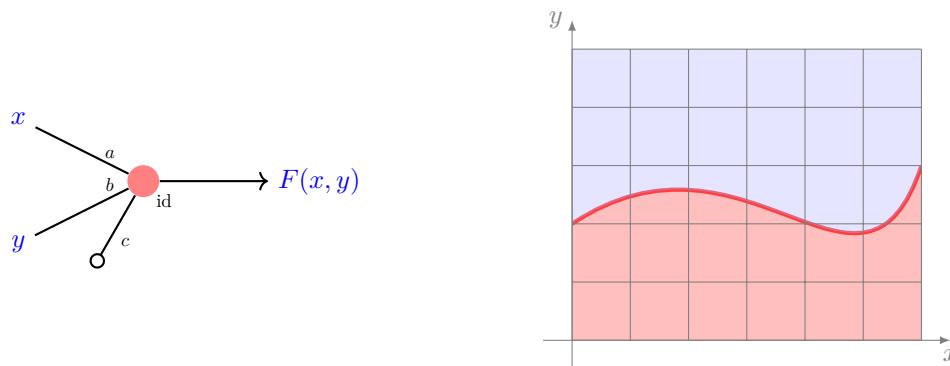
**Problème.** Voici deux exemples de situations dans lesquelles le choix de l'architecture du réseau pose problème, car le réseau est trop simple.

**Exemple 1.** Imaginons que nous voulions modéliser une situation à partir des données fournies par les points  $(x_i, y_i)$  du plan (figure de gauche). Si on construit un réseau d'un seul neurone (figure de droite), alors la fonction de sortie sera du type  $y = F(x) = ax + b$ .



Aucun paramètre  $(a, b)$  ne permettra une modélisation correcte de la situation, car les points ne sont franchement pas alignés.

**Exemple 2.** Le même problème se produirait si on voulait trouver une fonction  $F$ , construite à partir d'un seul neurone, valant 0 pour la zone bleue et 1 pour la zone rouge de la figure ci-dessous. Ce n'est pas possible car la frontière de la solution n'est pas linéaire.

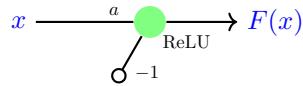


**Conclusion.** Dans ces situations, le problème ne se situe pas au niveau des poids, augmenter la taille des données ou le nombre d'itérations n'y changera rien. La conception du réseau est mauvaise pour répondre à la question posée. C'est comme vouloir faire une course de voitures avec une deux-chevaux. La solution est de changer l'architecture du réseau, par exemple en rajoutant des neurones.

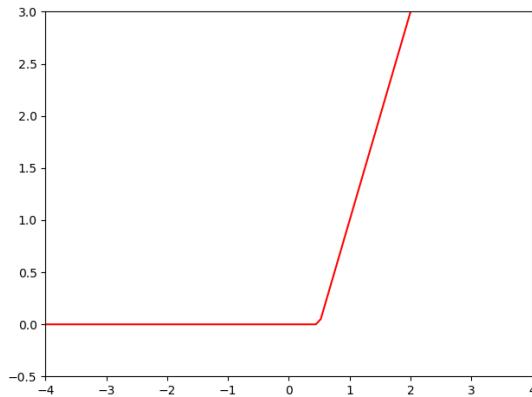
## 2. Minimum local

**Problème.** La descente de gradient a pour objectif de fournir un minimum local. Cependant rien n'affirme que ce minimum local est un minimum global.

**Exemple.** Voici un réseau composé d'un seul neurone de fonction d'activation ReLU et possédant un seul coefficient à déterminer, le biais étant imposé à la valeur  $-1$ .

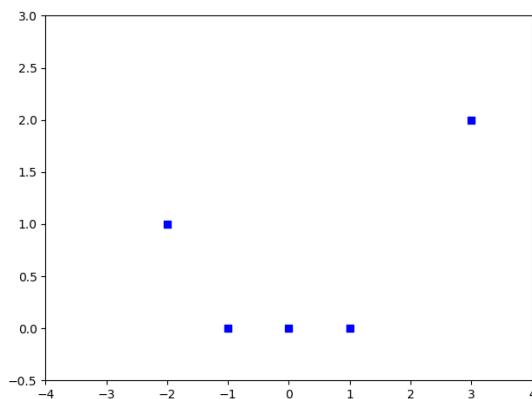


Le graphe de la fonction  $F$  correspondant à ce neurone est une portion de la droite d'équation  $y = ax - 1$ , prolongée par l'axe des abscisses. Voici l'exemple du graphe de  $F$ , avec  $a = +2$ .

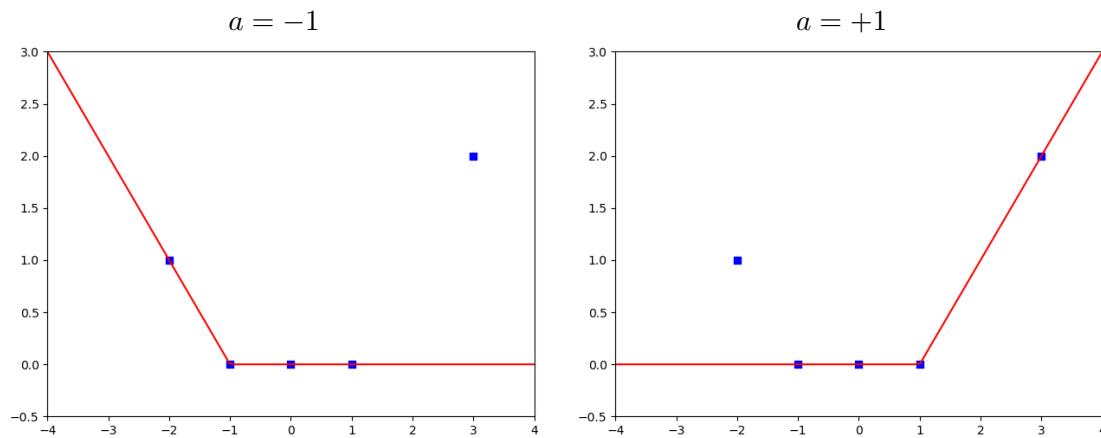


Les données sont des points déterminés par leurs coordonnées  $(x_i, y_i)$  :

$$(-2, 1), \quad (-1, 0), \quad (0, 0), \quad (1, 0), \quad (3, 2).$$

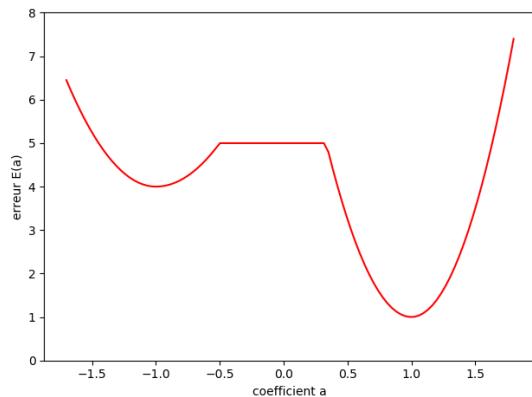


Il s'agit de trouver le meilleur coefficient  $a$ , qui définisse  $F$  tel que  $F(x_i) \simeq y_i$ . Autrement dit, on souhaite minimiser  $E(a) = \sum E_i(a)$  où  $E_i(a) = (F(x_i) - y_i)^2$ . Géométriquement il y a deux paramètres qui semblent meilleurs que les autres  $a = -1$  (figure de gauche) et  $a = +1$  (figure de droite) car alors les portions de droites passent par des carrés.



Cette intuition se vérifie lorsque l'on trace le graphe de la fonction d'erreur  $a \mapsto E(a)$ . La fonction possède deux minimums locaux en  $a = -1$  et  $a = +1$  (et aussi une portion constante, due à l'usage de la fonction ReLU).

**Erreur  $E(a)$  en fonction du coefficient  $a$**



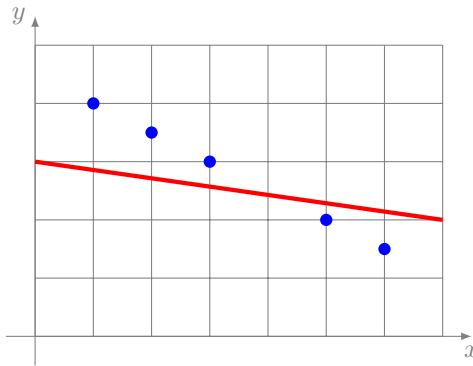
Le minimum en  $a = +1$  est le minimum global. Mais si on applique la descente de gradient en partant d'un coefficient initial  $a_0 < 0$ , il y a de grandes chances d'arriver au minimum local  $a = -1$ , qui ne sera pas la solution optimale.

**Conclusion.** Vous êtes une fourmi qui se promène dans une boîte d'œufs avec des trous de différentes profondeurs : vous ne pouvez pas voir à l'avance quel est l'emplacement le plus profond ! Une solution peut être de tester différentes valeurs initiales, dans le but d'obtenir différents minimums locaux.

### 3. Sous-apprentissage

**Problème.** Le **sous-apprentissage** révèle une conception correcte de l'architecture du réseau mais une mauvaise mise en œuvre. On obtient alors des poids qui ne répondent pas correctement au problème.

**Exemple.** Une droite obtenue par un réseau approche mal une suite de points pourtant alignés.



Cela peut être dû aux raisons suivantes :

- les données ne sont pas en nombre suffisant,
- le nombre d'itérations est insuffisant,
- le pas  $\delta$  est trop grand.

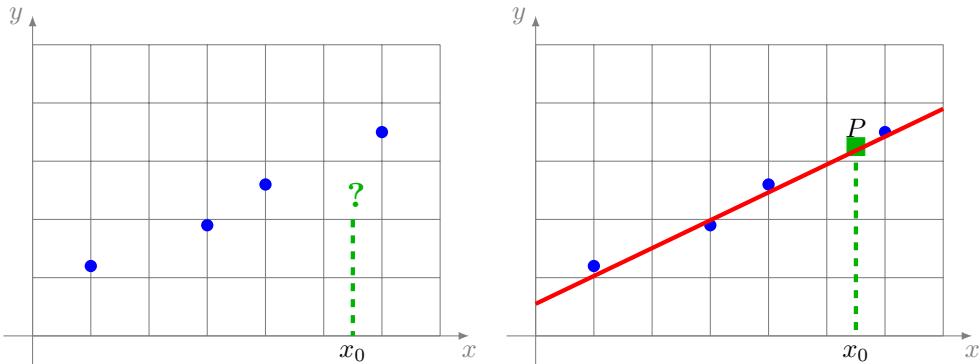
**Conclusion.** Il est difficile de savoir à l'avance quelle est la bonne taille des données à utiliser et combien d'itérations sont nécessaires pour arriver à un modèle correct. Le sous-apprentissage, c'est comme faire une course de voiture avec une porsche mais en ne dépassant jamais les 80 km/h ! A posteriori, la solution est simple : ajouter des données, augmenter le nombre d'itérations ou diminuer la pas.

#### 4. Sur-apprentissage

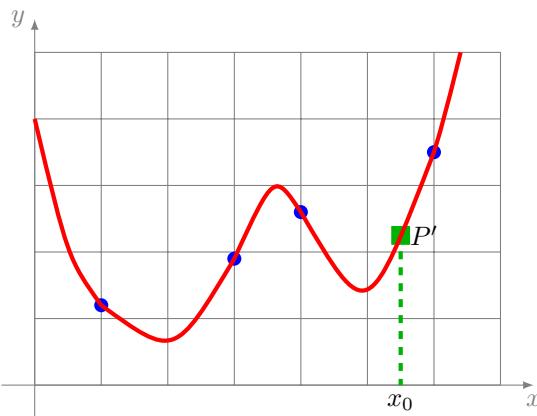
**Problème.** Le modèle obtenu « colle » parfaitement aux données d'apprentissage, mais cependant les prédictions pour de nouvelles valeurs sont mauvaises. Il s'agit donc d'un problème délicat : la fonction  $F$  obtenue vérifie bien  $F(X_i) \simeq z_i$  pour toutes les données, mais pour une nouvelle entrée  $X$ , la sortie  $F(X)$  n'est pas une bonne prédiction. Cela se produit lorsque l'on se concentre uniquement sur l'apprentissage à partir des données, mais que l'on a oublié que le but principal est la prédiction.

**Exemple 1.** Nous avons 4 points. Pour une valeur  $x_0$  donnée, on souhaite prédire une valeur  $y_0$  (figure de gauche) cohérente avec nos données.

On propose dans un premier temps d'approcher la solution en utilisant une droite (figure de droite) même si les points ne sont pas exactement alignés. Cela permet de prédire une valeur  $y_0$  pour placer le point  $P = (x_0, y_0)$ . Ce modèle ne sera jamais parfait car les points ne sont pas alignés, donc aucune droite ne convient exactement, autrement dit l'erreur ne sera jamais nulle.

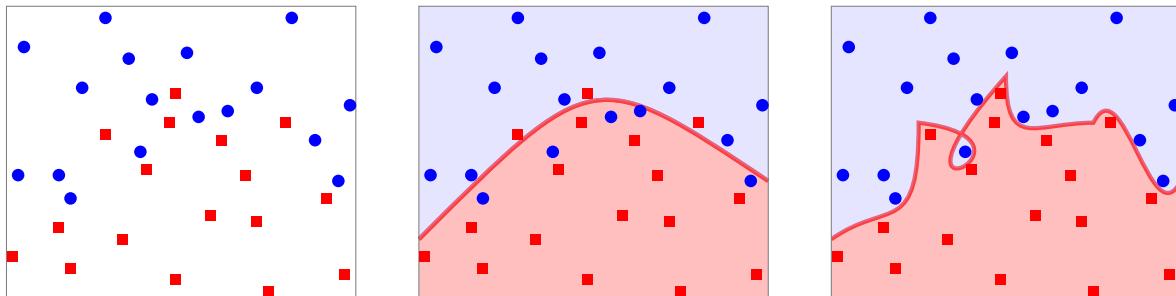


On peut construire un modèle plus compliqué, par exemple chercher une courbe polynomiale de degré 3 ou plus qui passe *exactement* par tous les points d'apprentissage. Ainsi, pour cette courbe, l'erreur sera nulle. Cette courbe permet de prédire une valeur  $y'_0$  pour placer un point  $P'(x_0, y'_0)$ .

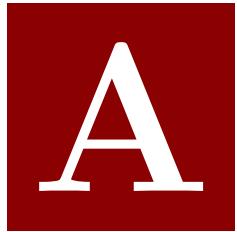


Ce dernier modèle répond entièrement au problème, mais la solution proposée ne semble pas raisonnable par rapport aux données de départ. C'est un cas de sur-apprentissage : le modèle est correct sur les données, mais les prédictions seront mauvaises.

**Exemple 2.** Le problème est similaire si on essaye de séparer les ronds bleus des carrés rouges de la figure de gauche ci-dessous. Une modèle simple permet à une parabole de séparer l'essentiel des ronds bleus des carrés rouges (figure centrale). On peut entraîner un modèle plus complexe pour qu'il délimite parfaitement les deux types de points, mais au prix d'une complexité non nécessairement voulue (figure de droite).



**Conclusion.** Le sur-apprentissage, c'est comme emmener ses enfants à l'école en conduisant à 200 km/h : cela répond de façon correcte à un problème, mais cela provoque beaucoup d'autres ennuis ! Quelle est la meilleure solution entre un modèle simple mais imparfait et un modèle parfait mais compliqué ? Il n'y a pas une réponse immédiate à la question : pour le savoir, il faut tester le modèle sur d'autres données, jamais rencontrées précédemment, afin de déterminer si le réseau n'a pas été sur-entraîné.



---

---

## **Compétences attendues à l'issue de ce cours**