Fiche Méthodes

Mathématiques pour le Machine Learning

Module : SM604

Un recueil de méthodes pour accompagner vos révisions

Doryan Denis

Version du document	**v1.0**
Première édition	22 juin 2026
Dernière mise à jour	22 juin 2026

Table des matières

1 La régression et la descente de gradient **2**
1.1 Calculer la droite de régression par la méthode des moindres carrés 2
1.2 Exprimer une fonction de coût (MSE) sous forme de polynôme 3
1.3 Caractériser un point critique à l'aide de la Hessienne 3
1.4 Exécuter à la main un pas de descente de gradient 4
1.5 Déterminer le learning rate critique de divergence par un développement de Taylor à l'ordre 2 . . 5
1.6 Estimer les paramètres d'un modèle par maximum de vraisemblance 6

2 La classification binaire **7**
2.1 Identifier le rôle d'une fonction de scores et d'une fonction d'activation 7
2.2 Calculer le score et la sortie d'un modèle selon la fonction d'activation choisie 8
2.3 Calculer le gradient de l'erreur par la règle de la chaîne et diagnostiquer un neurone mort 9
2.4 Calculer l'erreur MSE d'un modèle de classification pour une fonction de scores donnée 10
2.5 Calculer le gradient de la MSE (cas général) et observer l'évanouissement du gradient 10
2.6 Calculer l'entropie de Shannon, l'entropie croisée et la divergence de Kullback-Leibler 11
2.7 Calculer la fonction de coût Log Loss, son gradient, et exécuter la descente de gradient 12

3 Les réseaux de neurones **14**
3.1 Calculer la sortie d'un perceptron pour une entrée donnée 14
3.2 Construire un réseau de neurones réalisant une opération logique ou délimitant une région 14
3.3 Propager en avant dans un réseau multicouche avec une sortie Softmax 16

4 La rétropropagation **18**
4.1 Calculer le gradient de la fonction de coût par rapport aux poids de la couche de sortie 18
4.2 Rétropropager le gradient vers les couches cachées et diagnostiquer un neurone mort 19

1 La régression et la descente de gradient

1.1 Calculer la droite de régression par la méthode des moindres carrés

Méthode.
On considère un échantillon de n couples de données (x_i, y_i). La droite de régression de Y en fonction de X est la droite $y = ax + b$ qui minimise l'erreur quadratique moyenne (MSE) commise. On obtient directement ses coefficients par les formules des moindres carrés :

$$a = \frac{\sigma_{xy}}{\sigma_x^2} \qquad b = \overline{y} - a\,\overline{x}$$

où $\overline{x}$, $\overline{y}$ sont les moyennes empiriques, $\sigma_{xy} = \overline{xy} - \overline{x}\,\overline{y}$ est la covariance et $\sigma_x^2 = \overline{x^2} - \overline{x}^2$ la variance empirique de X.

On peut retrouver ce résultat en annulant le gradient de $E(a,b) = \frac{1}{n}\sum_{i=1}^{n}(ax_i + b - y_i)^2$, c'est-à-dire en résolvant le système $\partial E/\partial a = 0$, $\partial E/\partial b = 0$.

Exemple.
Soient les matrices des données X et des vraies étiquettes Y suivantes :

$$X = \begin{pmatrix} -4 \\ -1 \\ 2 \\ 1 \end{pmatrix} \qquad Y = \begin{pmatrix} -2 \\ -1 \\ 2 \\ 1 \end{pmatrix}$$

Calculer la droite de régression y en fonction de x.

Solution.
On calcule les moyennes empiriques :

$$\overline{x} = \frac{-4-1+2+1}{4} = -\frac{1}{2} \qquad \overline{y} = \frac{-2-1+2+1}{4} = 0$$

Puis la covariance et la variance :

$$\overline{xy} = \frac{(-4)(-2)+(-1)(-1)+(2)(2)+(1)(1)}{4} = \frac{14}{4} = 3{,}5 \quad \Rightarrow \quad \sigma_{xy} = 3{,}5 - (-\tfrac{1}{2})(0) = 3{,}5$$

$$\overline{x^2} = \frac{16+1+4+1}{4} = 5{,}5 \quad \Rightarrow \quad \sigma_x^2 = 5{,}5 - \Big(-\frac{1}{2}\Big)^2 = 5{,}25$$

D'où :

$$a = \frac{3{,}5}{5{,}25} = \frac{2}{3} \qquad b = 0 - \frac{2}{3}\Big(-\frac{1}{2}\Big) = \frac{1}{3}$$

$$\boxed{y = \frac{2}{3}x + \frac{1}{3}}$$

1.2 Exprimer une fonction de coût (MSE) sous forme de polynôme

Méthode.
Pour un modèle F dépendant de paramètres $A = (a_1, \ldots, a_m)$, la MSE s'écrit

$$E(A) = \frac{1}{n} \sum_{i=1}^{n} \big(F(\vec{x}_i) - y_i\big)^2$$

Pour l'exprimer sous forme d'un polynôme en les paramètres, on substitue l'expression de F dans chaque terme, on développe les carrés, puis on regroupe les termes de même degré. On obtient ainsi une forme quadratique (ou de degré supérieur pour un modèle non linéaire), ce qui permet ensuite de calculer facilement son gradient et sa Hessienne (méthode 1.3).

Exemple.
Soient $X = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$ la matrice des données et $Y = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ celle des vraies étiquettes. Pour ces $n = 2$ données avec $p = 1$ variable, on considère le modèle de régression $F(x) = ax$. Exprimer $MSE(a)$ sous forme d'un polynôme du second degré.

Solution.
On a $MSE(a) = \frac{1}{n} \sum_{i=1}^{n} (y_i - ax_i)^2$ avec $n = 2$:

$$MSE(a) = \frac{1}{2}\Big[\big(-1 - a(-2)\big)^2 + \big(1 - a(2)\big)^2\Big] = \frac{1}{2}\Big[(2a-1)^2 + (1-2a)^2\Big]$$

Les deux carrés sont identiques, donc :

$$MSE(a) = \frac{1}{2} \times 2(2a-1)^2 = (2a-1)^2$$

$$\boxed{MSE(a) = 4a^2 - 4a + 1}$$

1.3 Caractériser un point critique à l'aide de la Hessienne

Méthode.
Soit f une fonction de classe C^2 sur un ouvert convexe $\Omega \subset \mathbb{R}^n$. Un point $a \in \Omega$ est un **point critique** de f si $\nabla f(a) = 0$. On caractérise sa nature à l'aide de la matrice hessienne $H_f(a) = \left(\dfrac{\partial^2 f}{\partial x_i \partial x_j}(a)\right)_{i,j}$:

- si $H_f(a)$ est définie positive, a est un **minimum local** ;
- si de plus $H_f(y)$ est semi-définie positive en tout point $y \in \Omega$, alors f est convexe sur Ω et a est un **minimum global** ;
- de façon symétrique avec la négativité, on conclut à un maximum (local ou global).

Pour une matrice symétrique 2×2, $M = \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$ est définie positive si et seulement si ses mineurs principaux sont strictement positifs :

$$\alpha > 0 \qquad \text{et} \qquad \alpha\gamma - \beta^2 > 0$$

Exemple.
Pour le nuage de points $(-4,-1),(-1,-1),(2,2),(4,1)$, on a établi :

$$E(a,b) = \frac{37}{4}a^2 + b^2 + \frac{1}{2}ab - \frac{13}{2}a - \frac{1}{2}b + \frac{7}{4}$$

Trouver le point critique de E et déterminer sa nature.

Solution.
On annule le gradient :

$$\nabla E(a,b) = \begin{pmatrix} \frac{37}{2}a + \frac{1}{2}b - \frac{13}{2} \\ 2b + \frac{1}{2}a - \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \iff \begin{cases} a = \frac{17}{49} \simeq 0{,}3469 \\ b = \frac{8}{49} \simeq 0{,}1633 \end{cases}$$

La matrice hessienne est constante :

$$H = \begin{pmatrix} \frac{37}{2} & \frac{1}{2} \\ \frac{1}{2} & 2 \end{pmatrix}$$

Ses mineurs principaux valent $\frac{37}{2} > 0$ et $\frac{37}{2} \times 2 - \left(\frac{1}{2}\right)^2 = 36{,}75 > 0$: H est définie positive, et ceci en tout point puisqu'elle est constante.

E est convexe et $(a,b) = \left(\frac{17}{49}, \frac{8}{49}\right)$ est l'unique minimum global.

1.4 Exécuter à la main un pas de descente de gradient

Méthode.
L'algorithme de la descente de gradient s'écrit, pour un pas δ fixé :

$$A_{k+1} = A_k - \delta \nabla E(A_k)$$

En pratique, on calcule d'abord l'expression symbolique du gradient ∇E, puis on l'évalue numériquement au point courant A_k pour obtenir A_{k+1}. On répète l'opération autant de fois que demandé.

Exemple. Application n°1
On reprend $MSE(a) = 4a^2 - 4a + 1$ (méthode précédente). On initialise $a_0 = 10$. Calculer le gradient $\nabla MSE(a_0)$, puis effectuer le premier pas de descente avec un learning rate $\delta = 0{,}6$. Que remarque-t-on sur la valeur de a_1 ?

Solution.
Le gradient est la dérivée : $\nabla MSE(a) = 8a - 4$.
Pour $a_0 = 10$: $\nabla MSE(10) = 8(10) - 4 = 76$.

$$a_1 = a_0 - \delta \cdot \nabla MSE(a_0) = 10 - 0{,}6 \times 76 = 10 - 45{,}6 = \boxed{-35{,}6}$$

On remarque que a_1 a « sauté » par-dessus le minimum ($a^* = \frac{1}{2}$) pour finir très loin dans les valeurs négatives : le pas choisi est trop grand et l'algorithme devient instable (voir méthode suivante).

Exemple. Application n°2
Pour les données de la méthode 1.1 (nuage $(-4,-2),(-1,-1),(2,2),(1,1)$), on établit de même $MSE(a,b) = 5{,}5a^2 + b^2 - ab - 7a + 2{,}5$. On part du point $A_0 = (a_0, b_0) = (10,10)$ avec un pas constant $\delta = 0{,}05$. Donner à la main les trois premières itérations de la descente de gradient.

Solution.

Le gradient vaut $\nabla MSE(a,b) = \begin{pmatrix} 11a - b - 7 \\ 2b - a \end{pmatrix}$.

k	a_k	b_k
0	10	10
1	5,35	9,5
2	3,2325	8,8175
3	2,2455	8,0974

Détail du premier pas : $\nabla MSE(10,10) = (93, 10)$, donc $A_1 = (10 - 0{,}05 \times 93,\ 10 - 0{,}05 \times 10) = (5{,}35;\ 9{,}5)$. On répète ensuite le même calcul à chaque itération.

On observe que a_k diminue rapidement tandis que b_k décroît plus lentement : la descente corrige en priorité la coordonnée où la pente est la plus forte, avant de converger lentement vers le minimum $(2/3,\ 1/3)$ trouvé en méthode 1.1.

1.5 Déterminer le learning rate critique de divergence par un développement de Taylor à l'ordre 2

Méthode.
Lorsque la descente de gradient à un seul paramètre converge vers une valeur a^*, on étudie sa stabilité au voisinage de a^* à l'aide d'un développement de Taylor à l'ordre 2 de la fonction de coût f :

$$f(a) \approx f(a^*) + \frac{1}{2} f''(a^*)(a - a^*)^2 \quad \Longrightarrow \quad f'(a) \approx f''(a^*)(a - a^*)$$

En notant $e_n = a_n - a^*$ l'erreur à l'itération n, la récurrence $a_{n+1} = a_n - \delta f'(a_n)$ donne :

$$a_{n+1} - a^* = (a_n - a^*) - \delta f''(a^*)(a_n - a^*) \quad \Longrightarrow \quad e_{n+1} = \big(1 - \delta f''(a^*)\big) e_n$$

L'algorithme converge si et seulement si l'erreur diminue en valeur absolue à chaque étape, c'est-à-dire $-1 < 1 - \delta f''(a^*) < 1$, ce qui équivaut à :

$$\boxed{0 < \delta < \frac{2}{f''(a^*)}}$$

La valeur critique est donc $\delta_c = \dfrac{2}{f''(a^*)}$: au-delà de cette valeur, l'algorithme diverge.

Exemple.
Pour $MSE(a) = 4a^2 - 4a + 1$, déterminer la valeur critique δ_c du learning rate au-delà de laquelle l'algorithme diverge.

Solution.
On a $f(a) = MSE(a)$, donc $f''(a) = 8$ (constante). D'après la méthode précédente :

$$\delta_c = \frac{2}{f''(a^*)} = \frac{2}{8} = \boxed{0{,}25}$$

On retrouve ainsi que le pas $\delta = 0{,}6$ utilisé en méthode 1.4 dépassait largement $\delta_c = 0{,}25$, ce qui explique l'instabilité observée.

1.6 Estimer les paramètres d'un modèle par maximum de vraisemblance

Méthode.
Sous l'hypothèse que le bruit $E_i = y_i - F(\vec{x}_i)$ suit une loi $\mathcal{N}(0, \sigma^2)$ et que les observations sont indépendantes, maximiser la vraisemblance des étiquettes observées équivaut à minimiser la somme des carrés des résidus $\sum_i (F(\vec{x}_i) - y_i)^2$: c'est la méthode des moindres carrés.

Pour un modèle **linéaire sans biais** $F(\vec{x}_i) = \sum_j a_j x_{i,j}$, l'estimateur du maximum de vraisemblance s'écrit sous forme matricielle :

$$\boxed{A_{MLE} = (X^t X)^{-1} X^t Y}$$

Pour un modèle **polynomial** $F(x) = \sum_{j=0}^{m-1} a_j x^j$, on construit la matrice de Vandermonde $M = \left(x_i^j\right)_{i,j}$ et l'on obtient de la même façon $A_{MLE} = (M^t M)^{-1} M^t Y$.

Exemple.
Soient

$$X = \begin{pmatrix} -4 \\ -1 \\ 2 \\ 1 \\ 5 \\ 8 \\ 10 \end{pmatrix} \qquad Y = \begin{pmatrix} -2 \\ -1 \\ 2 \\ 1 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Faire à la main une régression avec le modèle $F(x) = ax$, en utilisant le maximum de vraisemblance.

Solution.
Le modèle est linéaire sans biais (une seule colonne dans X), donc $a_{MLE} = \dfrac{X^t Y}{X^t X}$:

$$X^t X = \sum_i x_i^2 = 16 + 1 + 4 + 1 + 25 + 64 + 100 = 211$$

$$X^t Y = \sum_i x_i y_i = 8 + 1 + 4 + 1 + 15 + 32 + 50 = 111$$

$$\boxed{a_{MLE} = \frac{111}{211} \simeq 0{,}526}$$

Remarque. Régression polynomiale
La même idée se généralise à un modèle polynomial $F(x) = a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}$: il suffit de remplacer X par la matrice de Vandermonde M et d'appliquer la même formule $A_{MLE} = (M^t M)^{-1} M^t Y$.

Récap.
Régression et descente de gradient :

- Droite de régression : $a = \sigma_{xy}/\sigma_x^2$, $b = \overline{y} - a\overline{x}$
- Descente de gradient : $A_{k+1} = A_k - \delta \nabla E(A_k)$
- Condition de convergence (1 paramètre) : $0 < \delta < 2/f''(a^*)$
- Maximum de vraisemblance (modèle linéaire) : $A_{MLE} = (X^t X)^{-1} X^t Y$

2 La classification binaire

2.1 Identifier le rôle d'une fonction de scores et d'une fonction d'activation

Méthode.
Un modèle de classification est la composée d'une fonction de scores f (qui combine les entrées, souvent de façon affine ou polynomiale) et d'une fonction d'activation ϕ :

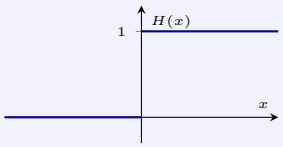
$$F(\vec{x}) = \phi\big(f(\vec{x})\big) \qquad \text{avec } o = f(\vec{x}) \text{ le score (ou logit)}$$

Le rôle de f : la frontière de décision est l'ensemble des points où $f(\vec{x}) = 0$ (de façon équivalente $\phi(f(\vec{x})) = 0{,}5$ pour une activation à valeurs dans $[0,1]$). Une fonction de scores **affine** donne une frontière **rectiligne** ; une fonction **polynomiale d'ordre 2** donne une **conique** (cercle, ellipse, parabole, hyperbole).

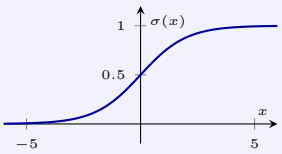
Le rôle de ϕ : elle ne déplace pas la frontière (qui ne dépend que de f), mais règle la netteté de la transition entre les deux classes au voisinage de cette frontière.

Définition. Quatre fonctions d'activation usuelles

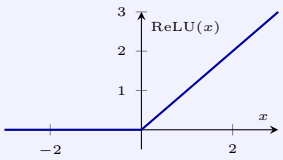
$$H(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$



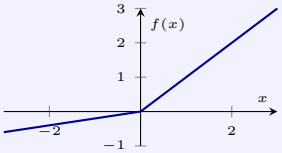
$$\sigma(x) = \frac{1}{1+e^{-x}}, \text{ avec } \sigma' = \sigma(1-\sigma)$$



$$\mathrm{ReLU}(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}$$



Leaky ReLU (α petit, ex. $0{,}01$) : $f(x) = \begin{cases} x & \text{si } x > 0 \\ \alpha x & \text{si } x \leq 0 \end{cases}$

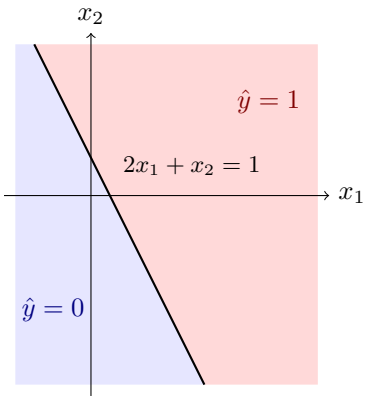


(pente négative exagérée ici pour la lisibilité)

Exemple.
Soit la fonction de scores affine $f(x_1, x_2) = 2x_1 + x_2 - 1$, utilisée pour classifier des points (étiquette 0 pour les ronds bleus, 1 pour les carrés rouges). Décrire la frontière de décision et la zone d'activation (a) pour la fonction de Heaviside, (b) pour la sigmoïde.

Solution.
(a) Avec Heaviside, $F(x_1, x_2) = H(2x_1 + x_2 - 1)$. La frontière de décision est la droite $2x_1 + x_2 = 1$: au-dessus (ou sur la droite) $F = 1$, en dessous $F = 0$. Par exemple pour $(x_1, x_2) = (5, 2)$: $2(5) + 2 - 1 = 11 \geq 0$ donc $H = 1$.

(b) Avec la sigmoïde, $F(x_1, x_2) = \sigma(2x_1 + x_2 - 1)$: la frontière (où $F = 0{,}5$) est exactement la **même droite**, mais la transition devient progressive : plus un point est loin de la droite, plus F se rapproche de 0 ou 1 ; près de la frontière, $F \simeq 0{,}5$ traduit une grande incertitude sur la prédiction.



2.2 Calculer le score et la sortie d'un modèle selon la fonction d'activation choisie

Méthode.
On calcule d'abord le **score** $o = f(\vec{x})$ en substituant les valeurs numériques de $\vec{x}$ et des paramètres dans la fonction de scores. On applique ensuite la fonction d'activation ϕ choisie pour obtenir la **sortie** $P = \phi(o)$, qui s'interprète comme la probabilité prédite (ou l'intensité d'activation) que la vraie étiquette soit $y = 1$.

Exemple.
Soit une donnée d'entrée $\vec{x} = (x_1, x_2) = (1, 2)$ associée à l'étiquette $y = 1$. Le score du modèle est $O = O(x_1, x_2) = x_2 - ax_1^4 - b$, avec les paramètres initialisés à $a = 2$, $b = 1$. La sortie P est l'activation du score O. Calculer O, puis P pour : (I) une activation sigmoïde ; (II) une activation ReLU ; (III) une activation Leaky ReLU de paramètre $\alpha = 0{,}01$.

Solution.
Le score ne dépend pas de l'activation choisie :

$$O = x_2 - ax_1^4 - b = 2 - 2 \times 1^4 - 1 = \boxed{-1}$$

Partie I (sigmoïde) : $P = \sigma(O) = \dfrac{1}{1 + e^{-O}} = \dfrac{1}{1 + e} \simeq \boxed{0{,}27}$

Partie II (ReLU) : comme $O = -1 < 0$, $P = \mathrm{ReLU}(O) = \max(0, O) = \boxed{0}$

Partie III (Leaky ReLU) : comme $O = -1 < 0$, $P = \alpha\, O = 0{,}01 \times (-1) = \boxed{-0{,}01}$

2.3 Calculer le gradient de l'erreur par la règle de la chaîne et diagnostiquer un neurone mort

Méthode.
Pour une fonction de coût E (par exemple $E = \frac{1}{2}(P-y)^2$) et un modèle $P = \phi(O)$ avec $O = f(\vec{x}; a)$, le gradient par rapport à un paramètre a s'obtient par la règle de la chaîne :

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial P} \cdot \frac{\partial P}{\partial O} \cdot \frac{\partial O}{\partial a} = (P-y) \cdot \phi'(O) \cdot \frac{\partial O}{\partial a}$$

La dérivée $\phi'(O)$ dépend de l'activation choisie : $\sigma'(O) = P(1-P)$; $\mathrm{ReLU}'(O) = 1$ si $O > 0$, 0 si $O < 0$; Leaky ReLU$'(O) = 1$ si $O > 0$, α si $O < 0$.

Neurone mort : si $\phi'(O) = 0$ (cas de ReLU pour $O < 0$), le gradient est nul quelle que soit l'erreur commise : le paramètre ne se met plus à jour. On dit que le neurone est « mort ».

Exemple.
On reprend les données de la méthode précédente ($O = -1$, $x_1 = 1$, $y = 1$, $a = 2$, learning rate $\delta = 0{,}1$).
Partie I (sigmoïde) : montrer que $\dfrac{\partial E}{\partial a} = (P-y)\,P(1-P)\,(-x_1^4)$, calculer sa valeur numérique, puis la nouvelle valeur de a après un pas de descente.
Partie II (ReLU) : calculer $\mathrm{ReLU}'(O)$, le gradient, puis la nouvelle valeur de a. Pourquoi le neurone est-il dit « mort » ?
Partie III (Leaky ReLU, $\alpha = 0{,}01$) : reprendre le même calcul, comparer avec la partie II et conclure.

Solution.
Partie I (sigmoïde) : Par dérivation en chaîne, avec $\frac{\partial O}{\partial a} = -x_1^4$:

$$\frac{\partial E}{\partial a} = (P-y)\,\sigma'(O)\,(-x_1^4) = (P-y)\,P(1-P)\,(-x_1^4)$$

Avec $P \simeq 0{,}27$, $y = 1$, $x_1 = 1$:

$$\frac{\partial E}{\partial a} = (0{,}27-1) \times 0{,}27 \times (1-0{,}27) \times (-1) \simeq \boxed{0{,}14}$$

$$a_{\text{new}} = a - \delta \frac{\partial E}{\partial a} = 2 - 0{,}1 \times 0{,}14 = \boxed{1{,}986}$$

Partie II (ReLU) : comme $O = -1 < 0$, $\mathrm{ReLU}'(O) = 0$, donc :

$$\frac{\partial E}{\partial a} = (P-y)\,\mathrm{ReLU}'(O)\,(-x_1^4) = (0-1) \times 0 \times (-1) = \boxed{0}$$

$$a_{\text{new}} = 2 - 0{,}1 \times 0 = \boxed{2} \text{ (inchangé)}$$

Le neurone est dit **mort** car le gradient est nul : le paramètre ne peut plus être mis à jour, quelle que soit l'erreur commise.
Partie III (Leaky ReLU) : comme $O = -1 < 0$, $f'(O) = \alpha = 0{,}01$, donc :

$$\frac{\partial E}{\partial a} = (P-y)\,f'(O)\,(-x_1^4) = (-0{,}01-1) \times 0{,}01 \times (-1) \simeq \boxed{0{,}0101}$$

$$a_{\text{new}} = 2 - 0{,}1 \times 0{,}0101 = \boxed{1{,}99899}$$

Avec ReLU, le gradient était exactement nul ; avec Leaky ReLU, il reste **faible mais non nul** : le neurone peut continuer à apprendre, même lentement, et sortir progressivement de l'état de neurone mort.

Remarque. Pourquoi préférer Leaky ReLU à la sigmoïde dans un réseau profond ?
La sigmoïde sature pour les grandes valeurs de $|O|$ ($\sigma'(O) \to 0$), ce qui provoque l'évanouissement du gradient (méthode 2.5). ReLU et ses variantes évitent ce problème car leur dérivée vaut 1 pour $O > 0$, quelle que soit l'amplitude du score. Mais ReLU peut alors produire des neurones morts pour $O < 0$; Leaky ReLU corrige ce défaut en conservant une pente faible mais non nulle côté négatif.

2.4 Calculer l'erreur MSE d'un modèle de classification pour une fonction de scores donnée

Méthode.
Pour une fonction de scores f et une activation ϕ, l'erreur quadratique moyenne d'un modèle de classification s'écrit :

$$E = \frac{1}{n}\sum_{i=1}^{n}\big(\phi(f(\vec{x}_i)) - y_i\big)^2$$

On calcule le score $o_i = f(\vec{x}_i)$ pour chaque donnée, on l'active avec ϕ, et on compare au y_i correspondant.

Remarque importante : avec la fonction de Heaviside, deux droites séparant parfaitement les données donnent toutes deux $E = 0$, sans permettre de les départager. La sigmoïde, elle, donne une valeur d'erreur continue qui permet de comparer la qualité (la marge) de plusieurs frontières candidates.

Exemple.
Soient les données $X = \big((-2,0),(0,-1),(1,1),(2,0)\big)$ et les étiquettes $Y = (0,0,1,1)$. Calculer l'erreur MSE de toutes les entrées avec la fonction de Heaviside comme activation pour la droite de classification $x_2 = -x_1$, puis pour la droite $x_2 = 1{,}5 - x_1$. Laquelle est la meilleure ?

Solution.
Avec Heaviside. Pour $x_2 = -x_1$, on prend $f(x_1,x_2) = x_1 + x_2$. On calcule f_i pour chaque point : -2, -1, 2, 2, donc $H(f_i) = 0,0,1,1$, ce qui correspond exactement à Y : $E = 0$.
Pour $x_2 = 1{,}5 - x_1$, on prend $f(x_1,x_2) = x_1 + x_2 - 1{,}5$: on trouve $f_i = -3{,}5,\ -2{,}5,\ 0{,}5,\ 0{,}5$, donc $H(f_i) = 0,0,1,1$ également : $E = 0$.
Les deux droites séparent parfaitement les données : impossible de les départager avec Heaviside.

Avec la sigmoïde, on calcule $P_i = \sigma(f_i)$ pour chaque droite.
Pour $x_2 = -x_1$: $P = (0{,}1192;\ 0{,}2689;\ 0{,}8808;\ 0{,}8808)$, d'où

$$E_1 = \frac{1}{4}\Big[0{,}1192^2 + 0{,}2689^2 + (0{,}8808-1)^2 + (0{,}8808-1)^2\Big] \simeq \boxed{0{,}029}$$

Pour $x_2 = 1{,}5 - x_1$: $P = (0{,}0293;\ 0{,}0759;\ 0{,}6225;\ 0{,}6225)$, d'où

$$E_2 = \frac{1}{4}\Big[0{,}0293^2 + 0{,}0759^2 + (0{,}6225-1)^2 + (0{,}6225-1)^2\Big] \simeq \boxed{0{,}073}$$

Comme $E_1 < E_2$, la droite $x_2 = -x_1$ est la meilleure : elle place les points plus loin de la frontière (plus grande marge), ce que seule la sigmoïde permet de détecter.

2.5 Calculer le gradient de la MSE (cas général) et observer l'évanouissement du gradient

Méthode.
Pour une fonction de scores quelconque $o = f(\vec{x}; A)$, une activation sigmoïde et un coût MSE, le gradient par rapport à un paramètre générique q de A vaut :

$$\frac{\partial E}{\partial q} = \frac{2}{n}\sum_{i=1}^{n}\frac{\partial o_i}{\partial q}\,\sigma'(o_i)\,\big(\sigma(o_i) - y_i\big)$$

Le facteur $\sigma'(o_i)$ devient **très petit dès que** $|o_i|$ **est grand** (saturation de la sigmoïde) : le gradient peut alors s'annuler numériquement même loin de l'optimum, et les paramètres se « gèlent ». C'est l'**évanouissement du gradient** (vanishing gradient).

Exemple.
Soit la fonction de scores $f(x_1,x_2;a) = ax_1 + x_2$ (droite $x_2 = -ax_1$), pour les données $X = \big((-2,0),(0,-1),(1,1),(2,0)\big)$, $Y = (0,0,1,1)$, avec activation sigmoïde et coût MSE. On part de $a_0 = 10$

avec $\delta = 0{,}01$. Calculer le gradient $\nabla E(a_0)$ et commenter.

Solution.
Les scores valent $o_i = a_0 x_{1,i} + x_{2,i}$: $o_1 = -20$, $o_2 = -1$, $o_3 = 11$, $o_4 = 20$. On a $\sigma(-20) \simeq 0$, $\sigma(11) \simeq 1$, $\sigma(20) \simeq 1$, et donc $\sigma'(o_i) = \sigma(o_i)(1 - \sigma(o_i)) \simeq 0$ pour $i = 1, 3, 4$ (les trois points classés avec un score de grande amplitude). Seul $o_2 = -1$ a un σ' non négligeable, mais $\partial o_2 / \partial a = x_{1,2} = 0$: sa contribution est nulle elle aussi !

$$\nabla E(a_0) = \frac{2}{4} \sum_i x_{1,i}\, \sigma'(o_i)\, (\sigma(o_i) - y_i) \simeq \boxed{0}$$

$$a_1 = a_0 - \delta \nabla E(a_0) \simeq a_0 = 10 \text{ (paramètre figé)}$$

Pourtant $a_0 = 10$ **n'est pas la valeur optimale** : c'est bien un cas d'évanouissement du gradient, et non de convergence.

Remarque. Causes et remèdes de l'évanouissement du gradient
Causes : activations qui saturent (sigmoïde, tanh) pour $|o|$ grand ; paramètres d'initialisation trop grands ; produit de nombreuses petites dérivées dans un réseau profond (chapitre 4).
Remèdes : activations ReLU et variantes ; meilleure initialisation ; normalisation des données ; ou changer de fonction de coût — c'est l'objet de la méthode suivante avec la Log Loss.

2.6 Calculer l'entropie de Shannon, l'entropie croisée et la divergence de Kullback-Leibler

Définition. Entropie, entropie croisée, divergence KL
Pour une distribution discrète (p_k) : entropie de Shannon $H_p = -\sum_k p_k \log_2(p_k)$ (en bits).
Pour deux distributions (p_k), (q_k) sur le même alphabet : entropie croisée $H(p, q) = -\sum_k p_k \log_2(q_k)$, et divergence de Kullback-Leibler $D_{KL}(p, q) = H(p, q) - H_p = \sum_k p_k \log_2\left(\frac{p_k}{q_k}\right) \geq 0$, avec égalité si et seulement si $p = q$.

Attention : D_{KL} n'est pas une distance : elle n'est en général pas symétrique, $D_{KL}(p, q) \neq D_{KL}(q, p)$.

Méthode.
Pour calculer ces quantités à la main : on calcule d'abord les probabilités p_k (ou q_k) si elles ne sont pas déjà données (par exemple à partir d'effectifs observés), puis on applique directement les formules ci-dessus terme à terme, en utilisant $\log_2(x) = \ln(x)/\ln(2)$.

Exemple. Application n°1
Une source de symboles émet quatre symboles A, B, C, D avec les effectifs suivants, observés sur un échantillon de 100 symboles : A : 45, B : 13, C : 12, D : 30. Calculer les probabilités, puis l'entropie de Shannon H_p en bits, et interpréter.

Solution.
Les probabilités sont $P(A) = 0{,}45$, $P(B) = 0{,}13$, $P(C) = 0{,}12$, $P(D) = 0{,}30$.

$$H_p = -\big(0{,}45 \log_2 0{,}45 + 0{,}13 \log_2 0{,}13 + 0{,}12 \log_2 0{,}12 + 0{,}30 \log_2 0{,}30\big)$$

$$H_p \simeq 0{,}518 + 0{,}383 + 0{,}367 + 0{,}521 = \boxed{1{,}79 \text{ bits}}$$

Cela signifie qu'il faut, en moyenne, $1{,}79$ bits par symbole pour coder cette source sans perte (c'est la borne théorique d'un codage optimal, par exemple de Huffman) : sensiblement moins que les $\log_2(4) = 2$ bits nécessaires si les quatre symboles étaient équiprobables.

Exemple. Application n°2
Soient les deux distributions discrètes sur $\{A, B, C\}$: p : (0,50; 0,30; 0,20) et q : (0,40; 0,40; 0,20). Calculer

$D_{KL}(p,q)$ et $D_{KL}(q,p)$ (en bits), et comparer.

Solution.

$$D_{KL}(p,q) = 0{,}5\log_2\frac{0{,}5}{0{,}4} + 0{,}3\log_2\frac{0{,}3}{0{,}4} + 0{,}2\log_2\frac{0{,}2}{0{,}2} \simeq 0{,}161 - 0{,}125 + 0 = \boxed{0{,}037 \text{ bits}}$$

$$D_{KL}(q,p) = 0{,}4\log_2\frac{0{,}4}{0{,}5} + 0{,}4\log_2\frac{0{,}4}{0{,}3} + 0{,}2\log_2\frac{0{,}2}{0{,}2} \simeq -0{,}129 + 0{,}166 + 0 = \boxed{0{,}037 \text{ bits}}$$

On trouve ici des valeurs très proches (à l'arrondi), mais elles ne sont pas rigoureusement égales ($0{,}0365$ contre $0{,}0372$ en gardant plus de décimales) : ceci confirme que D_{KL} n'est pas symétrique, donc pas une distance au sens mathématique.

2.7 Calculer la fonction de coût Log Loss, son gradient, et exécuter la descente de gradient

Méthode.
La Log Loss (entropie croisée binaire) s'écrit, pour une activation sigmoïde :

$$L = -\frac{1}{n}\sum_{i=1}^{n}\Big(y_i \ln\big(\sigma(o_i)\big) + (1-y_i)\ln\big(1-\sigma(o_i)\big)\Big)$$

Son gradient par rapport à un paramètre générique q est remarquablement plus simple que celui de la MSE : **le facteur σ' disparaît** :

$$\frac{\partial L}{\partial q} = \frac{1}{n}\sum_{i=1}^{n}\big(\sigma(o_i) - y_i\big)\frac{\partial o_i}{\partial q}$$

C'est cette simplification algébrique qui évite l'évanouissement du gradient rencontré avec la MSE (méthode 2.5) : même si $|o_i|$ est grand, le terme $\sigma'(o_i)$ ne vient plus multiplier (et donc potentiellement annuler) l'erreur.

Exemple.
Soient les données $X = \big((-2,0),(-5,-3),(-1,0),(0,-1),(3,1),(3,4),(6,2)\big)$ et $Y = (0,0,0,0,1,1,1)$, avec activation sigmoïde et fonction de scores $o = ax_1 + bx_2 + c$. On part de $A_0 = (a_0,b_0,c_0) = (1,1,1)$ avec $\delta = 0{,}01$. Calculer le gradient de la Log Loss en A_0 et donner la première itération.

Solution.
On calcule les scores $o_i = x_{1,i} + x_{2,i} + 1$: $o = (-1,-7,0,0,5,8,9)$, puis $\sigma(o_i) \simeq (0{,}269;\ 0{,}001;\ 0{,}5;\ 0{,}5;\ 0{,}993;\ 0{,}9997;\ 0{,}9999)$. Les résidus $e_i = \sigma(o_i) - y_i$ valent $(0{,}269;\ 0{,}001;\ 0{,}5;\ 0{,}5;\ -0{,}007;\ -0{,}0003;\ -0{,}0001)$.

$$\frac{\partial L}{\partial a} = \frac{1}{7}\sum_i e_i x_{1,i} \simeq \boxed{-0{,}152} \qquad \frac{\partial L}{\partial b} = \frac{1}{7}\sum_i e_i x_{2,i} \simeq \boxed{-0{,}073} \qquad \frac{\partial L}{\partial c} = \frac{1}{7}\sum_i e_i \simeq \boxed{0{,}180}$$

$$A_1 = A_0 - \delta\nabla L(A_0) \simeq (1{,}0015;\ 1{,}0007;\ 0{,}9982)$$

À la différence de la méthode 2.5, le gradient garde ici une amplitude raisonnable malgré des scores eux aussi très saturés ($o_6 = 8$, $o_7 = 9$) : **aucune composante ne s'annule numériquement.**

Remarque. Lien avec le maximum de vraisemblance
Comme pour la régression (méthode 1.6), on montre que la Log Loss est l'opposé de la log-vraisemblance d'un modèle de Bernoulli : minimiser la Log Loss équivaut donc à maximiser la vraisemblance du modèle de classification.

Récap.

Classification binaire :

- Gradient (règle de la chaîne) : $\dfrac{\partial E}{\partial a} = (P - y)\,\phi'(O)\,\dfrac{\partial O}{\partial a}$
- Neurone mort : $\phi'(O) = 0$ (ReLU pour $O < 0$) $\Rightarrow$ gradient nul, plus de mise à jour
- MSE + sigmoïde : $\dfrac{\partial E}{\partial q} = \dfrac{2}{n}\sum_i \dfrac{\partial o_i}{\partial q}\sigma'(o_i)(\sigma(o_i) - y_i)$ — risque d'évanouissement du gradient
- Log Loss + sigmoïde : $\dfrac{\partial L}{\partial q} = \dfrac{1}{n}\sum_i(\sigma(o_i) - y_i)\dfrac{\partial o_i}{\partial q}$ — pas d'évanouissement
- Entropie croisée : $H(p,q) = -\sum_k p_k \log_2(q_k), \quad D_{KL}(p,q) = H(p,q) - H_p \geq 0$

3 Les réseaux de neurones

3.1 Calculer la sortie d'un perceptron pour une entrée donnée

Méthode.
Un perceptron (neurone formel) calcule un score affine puis lui applique une fonction d'activation :

$$F(\vec{x}) = \phi\Big(a_0 + \sum_{i=1}^{p} a_i x_i\Big)$$

Pour évaluer F en une entrée donnée, on substitue les valeurs numériques de $\vec{x}$ et des poids (a_i) pour obtenir le score, puis on applique ϕ.

Exemple.
Soient les quatre perceptrons suivants, de poids (a_1, a_2, a_0) et fonction d'activation indiquée :

$$F_1 = H(7x_1 - x_2 - 5) \quad F_2 = H(-2x_1 - x_2 + 3) \quad F_3 = \sigma(7x_1 - x_2 - 5) \quad F_4 = \sigma(-2x_1 - x_2 + 3)$$

Calculer F_1, F_2, F_3, F_4 pour $(x_1, x_2) \in \{(0,0),\ (1,-2),\ (3,2),\ (-2,1),\ (-1,-1)\}$.

Solution.
Détail pour $(1,-2)$ *:* le score de F_1, F_3 vaut $7(1)-(-2)-5 = 4$ et celui de F_2, F_4 vaut $-2(1)-(-2)+3 = 3$.
D'où $F_1 = H(4) = 1$, $F_2 = H(3) = 1$, $F_3 = \sigma(4) \simeq 0{,}982$, $F_4 = \sigma(3) \simeq 0{,}953$.
On répète ce calcul pour chaque point :

(x_1, x_2)	score (F_1, F_3)	score (F_2, F_4)	F_1	F_2	F_3 / F_4
$(0,0)$	-5	3	0	1	$0{,}007$ / $0{,}953$
$(1,-2)$	4	3	1	1	$0{,}982$ / $0{,}953$
$(3,2)$	14	-5	1	0	$1{,}000$ / $0{,}007$
$(-2,1)$	-20	6	0	1	$0{,}000$ / $0{,}998$
$(-1,-1)$	-11	6	0	1	$0{,}000$ / $0{,}998$

On retrouve que F_3 et F_4 donnent des valeurs très proches de 0 ou 1, sauf lorsque le score est proche de 0 (incertitude), conformément à la méthode 2.2.

3.2 Construire un réseau de neurones réalisant une opération logique ou délimitant une région

Méthode.
Un **perceptron unique** (avec Heaviside) sépare le plan par une seule droite : il ne peut réaliser qu'une fonction booléenne **linéairement séparable** (OU, ET, NON, NAND, NOR…).

Pour **intersecter** p demi-plans (par exemple pour délimiter un polygone convexe), on place un neurone « ET » sur une seconde couche : tous ses poids valent 1 et son biais $-p$, ce qui n'active la sortie que si les p demi-plans sont actifs simultanément. De même, un neurone « OU » sur la seconde couche (poids 1, biais -1) réalise l'union des demi-plans.

Certaines fonctions booléennes, comme le **OU EXCLUSIF**, ne sont linéairement séparables par **aucune** droite : il faut alors au moins **deux couches**, en combinant par exemple une couche cachée « OU »/« ET » puis une seconde couche qui les recombine.

Propriété. Le OU EXCLUSIF est impossible avec un seul perceptron
Supposons qu'il existe a_1, a_2, a_0 tels que $H(a_1x_1 + a_2x_2 + a_0) = x_1 \oplus x_2$ pour tout $(x_1, x_2) \in \{0,1\}^2$. Les quatre cas donnent :

$$\text{XOR}(0,0) = 0 \Rightarrow a_0 < 0 \qquad \text{XOR}(1,1) = 0 \Rightarrow a_1 + a_2 + a_0 < 0$$

$$\text{XOR}(1,0) = 1 \Rightarrow a_1 + a_0 \geq 0 \qquad \text{XOR}(0,1) = 1 \Rightarrow a_2 + a_0 \geq 0$$

En sommant les deux dernières inégalités : $a_1 + a_2 \geq -2a_0$. En la combinant avec $a_1 + a_2 < -a_0$ (troisième inégalité), il vient $-2a_0 \leq a_1 + a_2 < -a_0$, ce qui impose $-2a_0 < -a_0$, c'est-à-dire $a_0 > 0$: ceci **contredit** $a_0 < 0$. Aucun perceptron à un seul neurone ne réalise donc le OU EXCLUSIF.

Exemple. Application n°1
On considère les trois demi-plans $-x_1 + 3x_2 + 1 \geq 0$, $2x_1 + x_2 + 1 \geq 0$ et $-4x_2 - x_1 + 6 \geq 0$. Construire un réseau de neurones dont la sortie ne s'active que sur leur intersection (un triangle).

Solution.
On place une première couche de trois neurones Heaviside, chacun associé à un demi-plan : $z_1 = H(-x_1 + 3x_2 + 1)$, $z_2 = H(2x_1 + x_2 + 1)$, $z_3 = H(-x_1 - 4x_2 + 6)$.
On ajoute une seconde couche, un neurone « ET » de poids $(1,1,1)$ et de biais -3 :

$$\text{sortie} = H(z_1 + z_2 + z_3 - 3)$$

Cette sortie ne vaut 1 que si les trois neurones de la première couche valent tous 1 simultanément, c'est-à-dire sur l'intersection des trois demi-plans : le triangle qu'ils délimitent.

Exemple. Application n°2
Construire un réseau de neurones réalisant : VRAI, FAUX, NON-ET (NAND), NON-OU (NOR), et NON-OUEX (XNOR), pour deux variables booléennes x_1, x_2.

Solution.
VRAI : poids nuls, biais 1 : $H(0 \cdot x_1 + 0 \cdot x_2 + 1) = H(1) = 1$, toujours.
FAUX : poids nuls, biais -1 : $H(-1) = 0$, toujours.
NAND : poids $(-1,-1)$, biais $1{,}5$: $H(-x_1 - x_2 + 1{,}5)$. On vérifie $(1,1) \mapsto H(-0{,}5) = 0$, et $(0,0), (1,0), (0,1) \mapsto 1$: c'est bien la négation du ET (poids $(1,1)$, biais $-1{,}5$, cf. cours 4.3.2).
NOR : poids $(-1,-1)$, biais $0{,}5$: $H(-x_1 - x_2 + 0{,}5)$. On vérifie $(0,0) \mapsto H(0{,}5) = 1$ et les trois autres cas $\mapsto 0$.
XNOR : comme pour XOR (propriété précédente), une seule couche ne suffit pas. On réutilise les neurones $h_1 = \text{OU} = H(x_1 + x_2 - 0{,}5)$ et $h_2 = \text{ET} = H(x_1 + x_2 - 1{,}5)$ d'une couche cachée, puis on calcule en sortie

$$\text{XNOR} = H(h_2 - h_1 + 0{,}5)$$

En effet $(0,0)$: $h_1 = h_2 = 0 \Rightarrow H(0{,}5) = 1$; $(1,1)$: $h_1 = h_2 = 1 \Rightarrow H(0{,}5) = 1$; $(1,0)$ et $(0,1)$: $h_1 = 1, h_2 = 0 \Rightarrow H(-0{,}5) = 0$. On retrouve bien $x_1 = x_2 \Rightarrow 1$, ce qui est la négation du OU EXCLUSIF.

3.3 Propager en avant dans un réseau multicouche avec une sortie Softmax

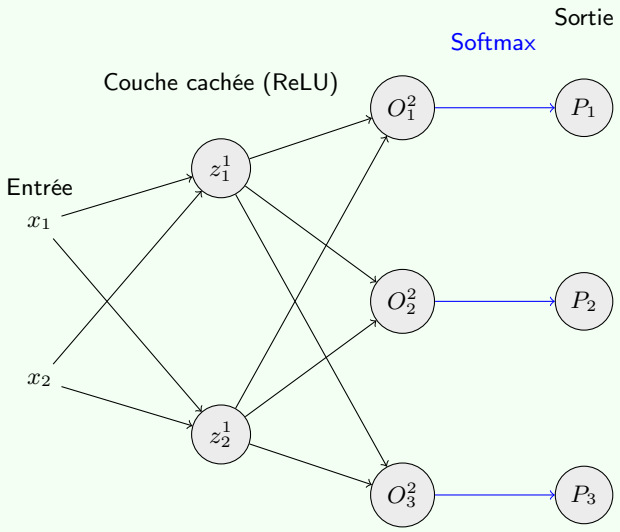
Méthode.
Dans un perceptron multicouche, on propage le signal couche par couche, de l'entrée vers la sortie :

1. pour chaque neurone q de la première couche cachée, calculer la pré-activation $O_q^1 = \sum_k a_{qk}^1 x_k$, puis l'activation $z_q^1 = \phi_1(O_q^1)$;
2. pour chaque neurone q de la couche de sortie, calculer $O_q^2 = \sum_k a_{qk}^2 z_k^1$;
3. pour une classification à C classes, appliquer la fonction Softmax :

$$P_q = \text{softmax}_q(\vec{O}^2) = \frac{e^{O_q^2}}{\sum_{j=1}^{C} e^{O_j^2}}$$

La classe prédite est $\hat{y} = \text{argmax}_q(P_q)$.

Exemple.
On considère le réseau ci-dessous :



Soit l'entrée $\vec{x} = (x_1, x_2) = (1, 0)$, appartenant à la classe 2 (donc $\vec{y} = (0, 1, 0)$). Les paramètres sont :

$$A^{(1)} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \qquad A^{(2)} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Calculer les neurones de la couche cachée, puis le vecteur de probabilités P, et donner la classe prédite.

Solution.
Couche cachée :

$$z_1^1 = \text{ReLU}(a_{11}^1 x_1 + a_{12}^1 x_2) = \text{ReLU}(1 \times 1 - 1 \times 0) = \text{ReLU}(1) = 1$$

$$z_2^1 = \text{ReLU}(a_{21}^1 x_1 + a_{22}^1 x_2) = \text{ReLU}(0 \times 1 + 1 \times 0) = \text{ReLU}(0) = 0$$

(de façon équivalente, $A^{(1)}\vec{x} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, puis on applique ReLU terme à terme.)

Couche de sortie :

$$\vec{O}^2 = A^{(2)} \begin{pmatrix} z_1^1 \\ z_2^1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Softmax : $\sum_k e^{O_k^2} = e^1 + e^1 + e^0 \simeq 2{,}72 + 2{,}72 + 1 = 6{,}44$, donc

$$P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} 2{,}72/6{,}44 \\ 2{,}72/6{,}44 \\ 1/6{,}44 \end{pmatrix} \simeq \boxed{\begin{pmatrix} 0{,}422 \\ 0{,}422 \\ 0{,}155 \end{pmatrix}}$$

La classe prédite est $\hat{y} = \operatorname{argmax}(P)$: les classes 1 et 2 sont ici ex æquo avec un score de $0{,}422$ chacune.

Récap.

Réseaux de neurones :

- Perceptron : $F(\vec{x}) = \phi(a_0 + \sum_i a_i x_i)$
- Couche « ET » de p demi-plans : poids 1, biais $-p$ — couche « OU » : poids 1, biais -1
- OU EXCLUSIF (et NON-OUEX) : impossibles en une seule couche, nécessitent une couche cachée
- Propagation avant : $O_q^h = \sum_k a_{qk}^h z_k^{h-1}$, $z_q^h = \phi_h(O_q^h)$
- Softmax : $P_q = e^{O_q} / \sum_j e^{O_j}$, prédiction $\hat{y} = \operatorname{argmax}_q(P_q)$

4 La rétropropagation

4.1 Calculer le gradient de la fonction de coût par rapport aux poids de la couche de sortie

Propriété. Gradient simplifié pour Softmax + entropie croisée
Lorsque la couche de sortie utilise l'activation Softmax et que le coût est l'entropie croisée (Log Loss multi-classes), le gradient par rapport aux scores de sortie se simplifie remarquablement, exactement comme pour sigmoïde + Log Loss (méthode 2.7) :

$$\frac{\partial L}{\partial O_q} = P_q - y_q$$

Méthode.
Pour la couche de sortie (notée H), une fois connu $\partial L/\partial O_q^H$ (cf. propriété ci-dessus pour Softmax + entropie croisée), le gradient par rapport au poids a_{qk}^H s'obtient par la règle de la chaîne :

$$\frac{\partial L}{\partial a_{qk}^H} = \frac{\partial L}{\partial O_q^H} \cdot \frac{\partial O_q^H}{\partial a_{qk}^H} = (P_q - y_q)\, z_k^{H-1}$$

puisque $O_q^H = \sum_k a_{qk}^H z_k^{H-1}$: le gradient est le produit de l'erreur de sortie sur le neurone q par l'entrée z_k^{H-1} qu'il a reçue.

Exemple.
On reprend le réseau de la méthode 3.3 ($P = (0{,}422;\ 0{,}422;\ 0{,}155)$, $y = (0, 1, 0)$, $z^1 = (1, 0)$). Calculer l'ensemble des gradients $\partial L/\partial a_{qk}^2$ de la couche de sortie.

Solution.

$\nabla a_{1,1}^2 = (0{,}422-0)\times 1 = \boxed{0{,}422}$ $\quad$ $\nabla a_{2,1}^2 = (0{,}422-1)\times 1 = \boxed{-0{,}578}$ $\quad$ $\nabla a_{3,1}^2 = (0{,}155-0)\times 1 = \boxed{0{,}155}$

$\nabla a_{1,2}^2 = (0{,}422-0)\times 0 = \boxed{0}$ $\quad$ $\nabla a_{2,2}^2 = (0{,}422-1)\times 0 = \boxed{0}$ $\quad$ $\nabla a_{3,2}^2 = (0{,}155-0)\times 0 = \boxed{0}$

Remarque : sous forme matricielle, ce gradient s'écrit

$$\frac{\partial L}{\partial A^{(2)}} = (P - y)\,(z^1)^\top = \begin{pmatrix} 0{,}422 \\ -0{,}578 \\ 0{,}155 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0{,}422 & 0 \\ -0{,}578 & 0 \\ 0{,}155 & 0 \end{pmatrix}$$

On retrouve bien les six valeurs calculées ci-dessus.

4.2 Rétropropager le gradient vers les couches cachées et diagnostiquer un neurone mort

Méthode.
Pour une couche cachée h, la règle de rétropropagation (cours, Proposition 5.2) donne :

$$\frac{\partial L}{\partial a^h_{qk}} = \underbrace{\left(\sum_j \frac{\partial L}{\partial O^{h+1}_j}\, a^{h+1}_{jq}\right)}_{\text{erreur rétropropagée}} \times \underbrace{\phi'_h(O^h_q)}_{\text{activation locale}} \times \underbrace{z^{h-1}_k}_{\text{entrée reçue}}$$

On reconnaît trois facteurs : l'erreur qui « redescend » depuis la couche suivante (pondérée par les poids qui relient le neurone q à cette couche), la dérivée locale de l'activation du neurone, et l'entrée que ce neurone a reçue.

Neurone mort (rappel méthode 2.3) : si $\phi'_h(O^h_q) = 0$ — typiquement un neurone ReLU dont la pré-activation O^h_q est négative ou nulle — alors **tout le gradient entrant dans ce neurone s'annule**, quels que soient les poids qui le précèdent : aucun de ses poids ne se met à jour.

Exemple.
On reprend le réseau des méthodes 3.3 et 4.1 (couche cachée ReLU : $z^1 = (1, 0)$, $O^1 = (1, 0)$; couche de sortie Softmax). Calculer les gradients $\partial L/\partial a^1_{21}$ et $\partial L/\partial a^1_{22}$, puis déterminer si l'un des deux neurones de la couche cachée est « mort » pour cet échantillon.

Solution.
La formule générale donne $\dfrac{\partial L}{\partial a^1_{qk}} = \left(\sum_j (P_j - y_j) a^2_{jq}\right) \times \mathrm{ReLU}'(O^1_q) \times x_k$.

Pour $q = 2$: $O^1_2 = 0$, donc $\mathrm{ReLU}'(O^1_2) = 0$ (par convention, comme dans les frameworks tels que PyTorch ou TensorFlow, qui prolongent arbitrairement la dérivée en 0 par 0). Ainsi, quelle que soit l'erreur rétropropagée :

$$\frac{\partial L}{\partial a^1_{21}} = \boxed{0} \qquad \frac{\partial L}{\partial a^1_{22}} = \boxed{0}$$

Le neurone z^1_2 est mort : sa pré-activation est nulle, donc la dérivée de ReLU y est nulle, et aucune erreur ne peut être rétropropagée à travers lui — on observe bien que les deux poids qui y mènent (a^1_{21} et a^1_{22}) ont un gradient nul.

Remarque : sous forme matricielle, $\dfrac{\partial L}{\partial A^{(1)}} = \left[(A^{(2)})^\top (P - y) \odot \mathrm{ReLU}'(O^1)\right] \vec{x}^{\,\top}$, où $\odot$ désigne le produit de Hadamard (terme à terme).

Exemple. Application n°2
(Généralisation, d'après le cours) Pour un réseau à deux couches cachées sigmoïdes et une sortie identité, avec un coût MSE, retrouver la règle de rétropropagation pour la couche de sortie et pour la couche cachée 2.

Solution.
Pour la couche de sortie (identité), $F(\vec{x}_i) = O^3_i = \sum_k a^3_k z^2_{i,k}$ et $E_i = \frac{1}{n}(F(\vec{x}_i) - y_i)^2$, donc :

$$\frac{\partial E_i}{\partial a^3_k} = \frac{2}{n}\big(F(\vec{x}_i) - y_i\big)\, z^2_{i,k}$$

Pour la couche cachée 2 (sigmoïde), avec $\sigma'(O^2_{i,q}) = z^2_{i,q}(1 - z^2_{i,q})$, la règle de rétropropagation donne :

$$\frac{\partial E_i}{\partial a^2_{qk}} = \underbrace{\frac{2}{n}\big(F(\vec{x}_i) - y_i\big)\, a^3_q}_{\text{erreur rétropropagée}} \times \underbrace{z^2_{i,q}(1 - z^2_{i,q})}_{\text{activation locale}} \times \underbrace{z^1_{i,k}}_{\text{entrée reçue}}$$

On retrouve exactement la **même structure** que dans l'exemple Softmax : (erreur de sortie pondérée par les poids) × (dérivée locale de l'activation) × (entrée reçue par le neurone) — seule la nature de l'activation (et donc de sa dérivée) change.

Récap.

Rétropropagation :

- Couche de sortie : $\dfrac{\partial L}{\partial a_{qk}^H} = \dfrac{\partial L}{\partial O_q^H}\, z_k^{H-1}$, avec $\dfrac{\partial L}{\partial O_q} = P_q - y_q$ pour Softmax + entropie croisée
- Couche cachée h : $\dfrac{\partial L}{\partial a_{qk}^h} = \Big(\sum_j \dfrac{\partial L}{\partial O_j^{h+1}} a_{jq}^{h+1}\Big) \phi_h'(O_q^h)\, z_k^{h-1}$
- Neurone mort : $\phi_h'(O_q^h) = 0 \Rightarrow$ gradient nul pour tous les poids entrant dans ce neurone
- Entraînement : on alterne propagation avant (méthode 3.3) et rétropropagation des erreurs, couche par couche, de la sortie vers l'entrée