Réseaux récurrents

Compétences travaillées : propagation avant dans un RNN, rétropropagation à travers le temps (BPTT), gradient vanishing, mécanisme de la cellule LSTM, comptage de paramètres (RNN, GRU, LSTM).

Version web

Ex 1 — Propagation forward dans un RNN simple

pK4m

★ ★

On considère un RNN simple à un seul état caché, défini par les équations :

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h), \qquad y_t = W_y h_t + b_y$$

avec les dimensions suivantes : entrée $x_t \in \mathbb{R}$, état caché $h_t \in \mathbb{R}^2$, sortie $y_t \in \mathbb{R}$.
Les paramètres du réseau sont :

$$W_x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad W_h = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}, \quad b_h = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad W_y = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad b_y = 0$$

L'état caché initial est $h_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. On traite la séquence d'entrée $x_1 = 1$, $x_2 = 0$, $x_3 = -1$.

1. Calculer h_1, h_2, h_3 en appliquant la formule de récurrence pas à pas. On rappelle que tanh s'applique composante par composante.

✓

Pas $t = 1$.

$$h_1 = \tanh(W_h h_0 + W_x x_1) = \tanh\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right) = \begin{pmatrix} \tanh(1) \\ \tanh(-1) \end{pmatrix} \approx \begin{pmatrix} 0.762 \\ -0.762 \end{pmatrix}$$

Pas $t = 2$.

$$W_h h_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} \tanh(1) \\ -\tanh(1) \end{pmatrix} = \begin{pmatrix} 0.5\tanh(1) \\ -0.5\tanh(1) \end{pmatrix} \approx \begin{pmatrix} 0.381 \\ -0.381 \end{pmatrix}, \qquad W_x x_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$h_2 = \tanh\left(\begin{pmatrix} 0.381 \\ -0.381 \end{pmatrix}\right) = \begin{pmatrix} \tanh(0.381) \\ \tanh(-0.381) \end{pmatrix} \approx \begin{pmatrix} 0.365 \\ -0.365 \end{pmatrix}$$

Pas $t = 3$.

$$W_h h_2 = \begin{pmatrix} 0.5\tanh(0.381) \\ -0.5\tanh(0.381) \end{pmatrix} \approx \begin{pmatrix} 0.183 \\ -0.183 \end{pmatrix}, \qquad W_x x_3 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}(-1) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

$$h_3 = \tanh\left(\begin{pmatrix} 0.183 \\ -0.183 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} \tanh(-0.817) \\ \tanh(0.817) \end{pmatrix} \approx \begin{pmatrix} -0.674 \\ 0.674 \end{pmatrix}$$

2. En déduire les sorties y_1, y_2, y_3.

✓ On applique $y_t = W_y h_t + b_y$. Puisque $W_y = \begin{pmatrix}1 & 1\end{pmatrix}$ et $b_y = 0$:

$$y_t = \begin{pmatrix}1 & 1\end{pmatrix} \begin{pmatrix} h_t^{(1)} \\ h_t^{(2)} \end{pmatrix} = h_t^{(1)} + h_t^{(2)}$$

$y_1 = \tanh(1)+\tanh(-1) \approx 0.762+(-0.762) = 0, \quad y_2 = \tanh(0.381)+\tanh(-0.381) \approx 0, \quad y_3 = \tanh(-0.817)+\tanh($

La sortie est identiquement nulle car les deux composantes de h_t sont systématiquement opposées, propriété induite par la structure antisymétrique de W_x et la parité impaire de tanh.

3. Décrire qualitativement ce que fait ce réseau : comment l'état caché h_t évolue-t-il lorsque l'entrée change de signe ?

✓ Les deux composantes de h_t sont en permanence opposées ($h_t^{(2)} = -h_t^{(1)}$), pour trois raisons conjointes :

- W_x injecte x_t dans la première composante et $-x_t$ dans la seconde ;
- $W_h = 0.5\, I_2$ atténue l'état précédent de façon identique sur les deux composantes, préservant leur relation d'opposition ;
- tanh est une fonction impaire, donc $\tanh(-u) = -\tanh(u)$ conserve l'opposition.

Lorsque x_t change de signe, les composantes s'inversent progressivement. Le réseau encode donc la polarité courante de l'entrée, atténuée par l'historique. La sortie y_t étant toujours nulle, l'information pertinente réside uniquement dans l'état caché h_t.

4. (*Bonus*) Que se passerait-il si on remplaçait tanh par la fonction identité ? Réécrire alors h_t comme une combinaison linéaire des entrées passées $x_1, \ldots, x_t$.

✓ Avec l'activation identité, la récurrence devient $h_t = W_h h_{t-1} + W_x x_t$. Par développement itératif (avec $h_0 = 0$) :

$$h_t = W_h^{t-1} W_x x_1 + W_h^{t-2} W_x x_2 + \cdots + W_h^0 W_x x_t = \sum_{k=1}^{t} W_h^{t-k} W_x x_k.$$

Puisque $W_h = 0.5\, I_2$, on a $W_h^{t-k} = (0.5)^{t-k} I_2$. En substituant :

$$h_t = \sum_{k=1}^{t} (0.5)^{t-k} I_2\, W_x x_k = W_x \left(\sum_{k=1}^{t} (0.5)^{t-k} x_k \right).$$

Le vecteur h_t est proportionnel à $W_x = \begin{pmatrix}1 \\ -1\end{pmatrix}$, le facteur de proportionnalité étant une moyenne exponentielle pondérée des entrées passées, avec des poids $(0.5)^{t-k}$ décroissant géométriquement vers le passé.

Ex 2 — Rétropropagation à travers le temps (BPTT)

★ ★ ★ Tz9w

On considère un RNN simple scalaire ($h_t \in \mathbb{R}$) défini par :

$$h_t = \tanh(w_h h_{t-1} + w_x x_t), \qquad y_t = w_y h_t$$

On dispose de la séquence $x_1 = 1$, $x_2 = 0$, $x_3 = -1$ et des cibles $\hat{y}_1 = 0$, $\hat{y}_2 = 0$, $\hat{y}_3 = 1$.

La fonction de perte est :

$$\mathcal{L} = \frac{1}{3}\sum_{t=1}^{3}(y_t - \hat{y}_t)^2$$

Les paramètres sont $w_x = 1$, $w_h = 0.5$, $w_y = 1$, et $h_0 = 0$.

1. (*Forward pass*) Calculer h_1, h_2, h_3, puis y_1, y_2, y_3, et enfin la perte $\mathcal{L}$.

✓

$$h_1 = \tanh(1) \approx 0.762, \quad h_2 = \tanh(0.5 \times 0.762) \approx \tanh(0.381) \approx 0.363$$

$$h_3 = \tanh(0.5 \times 0.363 - 1) = \tanh(-0.818) \approx -0.674$$

$$y_1 \approx 0.762, \quad y_2 \approx 0.363, \quad y_3 \approx -0.674$$

$$\mathcal{L} = \frac{1}{3}\left[(0.762)^2 + (0.363)^2 + (-0.674 - 1)^2\right] \approx \frac{1}{3}(0.581 + 0.132 + 2.802) \approx 1.172$$

2. Calculer $\dfrac{\partial \mathcal{L}}{\partial y_t}$ pour chaque t.

✓

$$\frac{\partial \mathcal{L}}{\partial y_t} = \frac{2}{3}(y_t - \hat{y}_t)$$

$$\frac{\partial \mathcal{L}}{\partial y_1} \approx \frac{2}{3}(0.762) \approx 0.508, \quad \frac{\partial \mathcal{L}}{\partial y_2} \approx \frac{2}{3}(0.363) \approx 0.242, \quad \frac{\partial \mathcal{L}}{\partial y_3} \approx \frac{2}{3}(-1.674) \approx -1.116$$

3. On pose $z_t = w_h h_{t-1} + w_x x_t$ (l'entrée du neurone avant activation) et $\delta_t = \dfrac{\partial \mathcal{L}}{\partial z_t}$. Montrer que :

$$\delta_t = \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot w_y + \delta_{t+1} \cdot w_h\right)(1 - h_t^2)$$

avec la convention $\delta_4 = 0$. Calculer numériquement δ_3, δ_2, δ_1 en remontant le temps.

✓

Par la règle de dérivation en chaîne, la perte dépend de z_t à travers y_t et à travers l'état suivant z_{t+1} :

$$\delta_t = \frac{\partial \mathcal{L}}{\partial z_t} = \frac{\partial \mathcal{L}}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial z_t} + \frac{\partial \mathcal{L}}{\partial z_{t+1}}\frac{\partial z_{t+1}}{\partial h_t}\frac{\partial h_t}{\partial z_t}$$

Comme $h_t = \tanh(z_t)$, on a $\dfrac{\partial h_t}{\partial z_t} = 1 - h_t^2$. De plus, $\dfrac{\partial y_t}{\partial h_t} = w_y$ et $\dfrac{\partial z_{t+1}}{\partial h_t} = w_h$. On obtient donc :

$$\delta_t = \frac{\partial \mathcal{L}}{\partial y_t} \cdot w_y \cdot (1 - h_t^2) + \delta_{t+1} \cdot w_h \cdot (1 - h_t^2) = \left(\frac{\partial \mathcal{L}}{\partial y_t} \cdot w_y + \delta_{t+1} \cdot w_h\right)(1 - h_t^2)$$

Calcul numérique ($\delta_4 = 0$) :

$$\delta_3 = (-1.116 \times 1 + 0) \times (1 - (-0.674)^2) \approx -1.116 \times 0.546 \approx -0.609$$

$$\delta_2 = (0.242 \times 1 + (-0.609) \times 0.5) \times (1 - 0.363^2) \approx (0.242 - 0.305) \times 0.868 \approx -0.054$$

$$\delta_1 = (0.508 \times 1 + (-0.054) \times 0.5) \times (1 - 0.762^2) \approx (0.508 - 0.027) \times 0.419 \approx 0.202$$

4. En déduire les gradients par rapport aux poids :

$$\frac{\partial \mathcal{L}}{\partial w_y} = \sum_{t=1}^{3} \frac{\partial \mathcal{L}}{\partial y_t} h_t, \qquad \frac{\partial \mathcal{L}}{\partial w_h} = \sum_{t=1}^{3} \delta_t h_{t-1}, \qquad \frac{\partial \mathcal{L}}{\partial w_x} = \sum_{t=1}^{3} \delta_t x_t$$

Calculer numériquement ces trois gradients.

✓

$$\frac{\partial \mathcal{L}}{\partial w_y} = 0.508 \times 0.762 + 0.242 \times 0.363 + (-1.116) \times (-0.674) \approx 0.387 + 0.088 + 0.752 \approx 1.227$$

$$\frac{\partial \mathcal{L}}{\partial w_h} = 0.202 \times 0 + (-0.054) \times 0.762 + (-0.609) \times 0.363 \approx 0 - 0.041 - 0.221 \approx -0.262$$

$$\frac{\partial \mathcal{L}}{\partial w_x} = 0.202 \times 1 + (-0.054) \times 0 + (-0.609) \times (-1) \approx 0.202 + 0 + 0.609 \approx 0.811$$

5. Le calcul de δ_t fait intervenir un produit récurrent en remontant le temps. Que se passe-t-il si la séquence est très longue ? Pourquoi le fait d'avoir $w_h < 1$ est-il un problème ?

✓

Le calcul de δ_t implique de remonter la chaîne de dépendances sur T pas. La contribution d'un pas de temps ancien au gradient est multipliée de façon répétée par le terme $w_h \cdot (1 - h_k^2)$. Si $|w_h| < 1$, ce produit de termes strictement inférieurs à 1 s'effondre exponentiellement vers zéro (surtout si les activations sont saturées, $|h_t| \approx 1$, ce qui donne $1 - h_t^2 \approx 0$). Les contributions des premiers pas de temps deviennent négligeables : le réseau perd sa capacité à apprendre des dépendances à long terme. C'est le phénomène de la *disparition du gradient* (gradient vanishing).

Ex 3 — Le problème du gradient vanishing

Hv2R

★ ★ ★

On considère un RNN simple scalaire sans entrée, pour isoler la dynamique de l'état caché :

$$h_t = \tanh(w_h \, h_{t-1})$$

On s'intéresse à la propagation du gradient sur T pas de temps. Par la règle de dérivation en chaîne :

$$\frac{\partial h_T}{\partial h_0} = \prod_{t=1}^{T} \frac{\partial h_t}{\partial h_{t-1}} = \prod_{t=1}^{T} w_h \cdot (1 - h_t^2)$$

1. Justifier la formule ci-dessus en appliquant la règle de dérivation en chaîne à $h_T = f(h_{T-1}) = f(f(\cdots f(h_0) \cdots))$.

✓

La fonction $f : x \mapsto \tanh(w_h x)$ est composée T fois. Par la règle de dérivation des fonctions composées :

$$\frac{\partial h_T}{\partial h_0} = \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial h_{T-1}}{\partial h_{T-2}} \cdots \frac{\partial h_1}{\partial h_0} = \prod_{t=1}^{T} f'(h_{t-1})$$

Or $f'(x) = w_h \cdot (1 - \tanh(w_h x)^2) = w_h \cdot (1 - h_t^2)$, d'où la formule.

2. Montrer que pour tout $h \in \mathbb{R}$, on a $0 < 1 - \tanh(h)^2 \leqslant 1$, avec égalité uniquement en $h = 0$. Que peut-on en conclure sur chaque facteur $w_h \cdot (1 - h_t^2)$ lorsque $|w_h| \leqslant 1$?

✓

On a $\tanh(h)^2 \in [0, 1)$ pour tout $h \in \mathbb{R}$ (puisque $|\tanh(h)| < 1$), donc $1 - \tanh(h)^2 \in (0, 1]$, avec la valeur 1 atteinte seulement en $h = 0$ (car $\tanh(0) = 0$).
Si $|w_h| \leqslant 1$, alors $|w_h \cdot (1 - h_t^2)| \leqslant 1$, et dès que $h_t \neq 0$, la majoration est stricte. Chaque facteur du produit est donc strictement inférieur à 1 en valeur absolue, ce qui implique que le produit de T tels facteurs tend vers 0 exponentiellement vite quand $T \to +\infty$.

3. On suppose $h_t \approx 0.9$ pour tout t, et $w_h = 0.8$. Calculer numériquement $\left|\dfrac{\partial h_T}{\partial h_0}\right|$ pour $T = 5$, $T = 10$, $T = 20$. Commenter.

✓

Chaque facteur vaut $|w_h \cdot (1 - h_t^2)| = 0.8 \times (1 - 0.9^2) = 0.8 \times (1 - 0.81) = 0.8 \times 0.19 = 0.152$.

$$T = 5 : \quad 0.152^5 \approx 8.15 \times 10^{-5}$$
$$T = 10 : \quad 0.152^{10} \approx 6.65 \times 10^{-9}$$
$$T = 20 : \quad 0.152^{20} \approx 4.42 \times 10^{-17}$$

Le gradient devient infinitésimal (pratiquement nul) dès les premiers pas de temps : toute information sur h_0 est perdue lors de la rétropropagation, rendant l'apprentissage de dépendances longues impossible.

4. Montrer que le gradient ne s'évanouit pas si et seulement si $|w_h| \cdot (1 - h_t^2) \geqslant 1$ pour tout t. Quelle condition sur w_h cela impose-t-il lorsque $h_t \approx 0$? Quel nouveau problème apparaît alors ?

✓

Pour que $\left|\dfrac{\partial h_T}{\partial h_0}\right|$ ne tende pas vers 0, il faut que le produit ne s'effondre pas, ce qui requiert que chaque facteur soit au moins 1 en valeur absolue : $|w_h| \cdot (1 - h_t^2) \geqslant 1$.
Lorsque $h_t \approx 0$, on a $1 - h_t^2 \approx 1$, donc la condition devient $|w_h| \geqslant 1$.
Mais si $|w_h| > 1$ et que les activations ne saturent pas, les facteurs sont tous strictement supérieurs à 1, et le produit diverge exponentiellement : c'est le problème du *gradient exploding*, qui rend l'entraînement instable (mises à jour de poids de taille arbitrairement grande).

5. On cherche à minimiser une perte $\mathcal{L}$ dépendant de h_T. Exprimer $\dfrac{\partial \mathcal{L}}{\partial w_h}\Big|_{t=1}$, la contribution du tout premier pas de temps au gradient de w_h, en fonction de $\dfrac{\partial \mathcal{L}}{\partial h_T}$ et du produit ci-dessus. Pourquoi les dépendances à long terme sont-elles si difficiles à apprendre ?

✓

Par la règle de dérivation en chaîne, la contribution du pas $t = 1$ est :

$$\left.\frac{\partial \mathcal{L}}{\partial w_h}\right|_{t=1} = \frac{\partial \mathcal{L}}{\partial h_T} \cdot \frac{\partial h_T}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_h} = \frac{\partial \mathcal{L}}{\partial h_T} \cdot \left(\prod_{t=2}^{T} w_h (1 - h_t^2)\right) \cdot h_0 (1 - h_1^2)$$

Ce terme est proportionnel au produit de $(T - 1)$ facteurs, chacun inférieur à 1 si $|w_h| \leqslant 1$. Il devient donc exponentiellement petit : le signal de gradient provenant de la perte à T ne parvient pratiquement plus au premier pas de temps, empêchant le réseau d'ajuster w_h pour capturer une dépendance entre h_1 et h_T.

Ex 4 **La cellule LSTM à la main** Xb5N

★ ★

Une cellule LSTM maintient deux vecteurs d'état : la *cell state* $c_t \in \mathbb{R}$ et l'état caché $h_t \in \mathbb{R}$ (on reste scalaire pour les calculs). À chaque pas de temps, quatre quantités intermédiaires sont calculées à partir de x_t et h_{t-1} :

$$
\begin{aligned}
f_t &= \sigma(w_f x_t + u_f h_{t-1} + b_f) && \text{(porte d'oubli)} \\
i_t &= \sigma(w_i x_t + u_i h_{t-1} + b_i) && \text{(porte d'entrée)} \\
\tilde{c}_t &= \tanh(w_c x_t + u_c h_{t-1} + b_c) && \text{(candidat)} \\
o_t &= \sigma(w_o x_t + u_o h_{t-1} + b_o) && \text{(porte de sortie)}
\end{aligned}
$$

La mise à jour des états suit :

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \qquad h_t = o_t \cdot \tanh(c_t)$$

où σ désigne la fonction sigmoïde.

Les paramètres sont tous les biais nuls et :

	w	u
Porte d'oubli f	0	0
Porte d'entrée i	2	0
Candidat $\tilde{c}$	2	0
Porte de sortie o	1	0

Les états initiaux sont $h_0 = 0$, $c_0 = 0$. On traite la séquence $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = -1$. *Pour gagner du temps, vous pouvez utiliser :* $\sigma(0) = 0.5$, $\sigma(1) \approx 0.731$, $\sigma(-1) \approx 0.269$, $\sigma(2) \approx 0.88$, $\sigma(-2) \approx 0.12$, $\tanh(2) \approx 0.96$, $\tanh(0) = 0$.

1. (*Pas $t = 1$*) Calculer $f_1, i_1, \tilde{c}_1, o_1$, puis c_1 et h_1.

✓

$$f_1 = \sigma(0 \cdot 1 + 0 \cdot 0) = \sigma(0) = 0.5$$

$$i_1 = \sigma(2 \cdot 1) = \sigma(2) \approx 0.88$$
$$\tilde{c}_1 = \tanh(2 \cdot 1) = \tanh(2) \approx 0.96$$
$$o_1 = \sigma(1 \cdot 1) = \sigma(1) \approx 0.731$$
$$c_1 = 0.5 \times 0 + 0.88 \times 0.96 \approx 0.845$$
$$h_1 = 0.731 \times \tanh(0.845) \approx 0.731 \times 0.688 \approx 0.503$$

2. (*Pas $t = 2$*) Calculer $f_2, i_2, \tilde{c}_2, o_2$, puis c_2 et h_2. Que remarque-t-on sur c_2 par rapport à c_1 ?

✓

Les poids u sont tous nuls, donc h_1 n'intervient pas :

$$f_2 = \sigma(0) = 0.5, \quad i_2 \approx 0.88, \quad \tilde{c}_2 \approx 0.96, \quad o_2 \approx 0.731$$

$$c_2 = 0.5 \times 0.845 + 0.88 \times 0.96 \approx 0.423 + 0.845 = 1.268$$
$$h_2 = 0.731 \times \tanh(1.268) \approx 0.731 \times 0.853 \approx 0.624$$

La cell state $c_2 > c_1$: la porte d'entrée a de nouveau injecté de l'information positive, qui s'accumule dans c_t grâce à la porte d'oubli (qui conserve environ la moitié de l'état précédent à chaque pas).

3. (*Pas* $t = 3$) L'entrée est $x_3 = 0$. Calculer c_3 et h_3. Que fait la porte d'oubli ici ? Comparer c_3 et c_2.

✓

$$f_3 = \sigma(0) = 0.5, \quad i_3 = \sigma(0) = 0.5, \quad \tilde{c}_3 = \tanh(0) = 0, \quad o_3 = \sigma(0) = 0.5$$

$$c_3 = 0.5 \times 1.268 + 0.5 \times 0 = 0.634$$

$$h_3 = 0.5 \times \tanh(0.634) \approx 0.5 \times 0.561 \approx 0.281$$

Avec une entrée nulle, la porte d'oubli conserve la moitié de c_2 et la porte d'entrée n'ajoute rien ($\tilde{c}_3 = 0$). La cell state diminue donc de moitié : $c_3 \approx c_2/2$. Le réseau « oublie » progressivement l'information accumulée.

4. (*Pas* $t = 4$) L'entrée est $x_4 = -1$. Calculer c_4 et h_4. Commenter le rôle de la porte d'entrée.

✓

$$f_4 = \sigma(0) = 0.5, \quad i_4 = \sigma(-2) \approx 0.12, \quad \tilde{c}_4 = \tanh(-2) \approx -0.96, \quad o_4 = \sigma(-1) \approx 0.269$$

$$c_4 = 0.5 \times 0.634 + 0.12 \times (-0.96) \approx 0.317 - 0.115 \approx 0.202$$

$$h_4 = 0.269 \times \tanh(0.202) \approx 0.269 \times 0.199 \approx 0.054$$

La porte d'entrée $i_4 \approx 0.12$ est faible pour une entrée négative : le candidat négatif $\tilde{c}_4$ n'est injecté que partiellement dans c_t. Cela atténue l'effondrement brutal de la cell state, qui reste positive. La porte d'entrée régule ainsi la quantité d'information nouvelle incorporée selon le signe de l'entrée.

5. Dans un RNN simple, l'état caché h_t était directement écrasé à chaque pas. Expliquer en quoi le mécanisme de cell state c_t de la LSTM permet de mieux préserver l'information sur le long terme.

✓

Dans le RNN simple, $h_t = \tanh(\cdots)$ est une transformation non linéaire saturante de h_{t-1} : l'état précédent est systématiquement « compressé » dans $[-1, 1]$, ce qui fait que le gradient peut facilement tendre vers 0 au fil des pas de temps (phénomène du *gradient vanishing*).
Dans la LSTM, la mise à jour de c_t est *additive* et modulée par des portes :

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$$

Si la porte d'oubli est ouverte ($f_t \approx 1$) et la porte d'entrée fermée ($i_t \approx 0$), alors $c_t \approx c_{t-1}$: la cell state est copiée presque intacte. Dans ce cas, la dérivée $\frac{\partial c_t}{\partial c_{t-1}} \approx 1$. Le gradient peut ainsi se propager linéairement et sans s'atténuer sur de longues séquences, ce qui résout le problème de l'oubli à long terme.

Ex 5 — Compter les paramètres d'un RNN, LSTM et GRU

Lq8D

★

Le nombre de paramètres d'un réseau est un indicateur clé de sa capacité et de son coût d'entraînement. On note :

- d : dimension de l'entrée x_t,
- h : dimension de l'état caché,
- p : dimension de la sortie y_t.

Chaque architecture possède une **couche de sortie linéaire** $y_t = W_y h_t + b_y$.

Rappel des architectures.
RNN simple :

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$

LSTM — quatre portes, chacune de la forme $Wx_t + Uh_{t-1} + b$:

$$f_t, \quad i_t, \quad \tilde{c}_t, \quad o_t$$

GRU — trois portes :

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) && \text{(porte de réinitialisation)} \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) && \text{(porte de mise à jour)} \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) && \text{(candidat)} \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned}$$

1. Justifier que le nombre de paramètres du RNN simple (hors couche de sortie) est $N_{\text{RNN}} = h(d + h + 1)$. Détailler la contribution de chaque matrice et biais.

✓ Le RNN simple possède trois jeux de paramètres :
— $W_x \in \mathbb{R}^{h \times d}$: hd paramètres,
— $W_h \in \mathbb{R}^{h \times h}$: h^2 paramètres,
— $b_h \in \mathbb{R}^h$: h paramètres.
Total : $hd + h^2 + h = h(d + h + 1)$.

2. Montrer que le nombre de paramètres de la LSTM (hors couche de sortie) est $N_{\text{LSTM}} = 4h(d + h + 1)$. Pourquoi ce facteur 4 ?

✓ Chacune des quatre portes (f, i, $\tilde{c}$, o) possède exactement la même structure qu'un RNN simple : une matrice $W \in \mathbb{R}^{h \times d}$, une matrice $U \in \mathbb{R}^{h \times h}$ et un biais $b \in \mathbb{R}^h$, soit $h(d + h + 1)$ paramètres par porte. Le facteur 4 provient du fait qu'il y a quatre portes indépendantes :

$$N_{\text{LSTM}} = 4 \times h(d + h + 1).$$

3. Montrer que le nombre de paramètres du GRU (hors couche de sortie) est $N_{\text{GRU}} = 3h(d + h + 1)$.

✓ Le GRU possède trois portes (r, z, $\tilde{h}$), chacune avec la même structure : $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ et $b \in \mathbb{R}^h$. (Remarque : dans $\tilde{h}_t$, U_h multiplie $r_t \odot h_{t-1}$, mais U_h reste de taille $h \times h$.) D'où :

$$N_{\text{GRU}} = 3 \times h(d + h + 1).$$

4. Ajouter la contribution de $W_y \in \mathbb{R}^{p \times h}$ et $b_y \in \mathbb{R}^p$. Écrire le nombre total de paramètres N_{total} pour chacune des trois architectures.

✓ La couche de sortie ajoute $ph + p = p(h + 1)$ paramètres dans les trois cas.

$$\begin{aligned} N_{\text{total}}^{\text{RNN}} &= h(d + h + 1) + p(h + 1) \\ N_{\text{total}}^{\text{LSTM}} &= 4h(d + h + 1) + p(h + 1) \\ N_{\text{total}}^{\text{GRU}} &= 3h(d + h + 1) + p(h + 1) \end{aligned}$$

5. Application numérique : $d = 10$, $h = 64$, $p = 5$. Compléter le tableau suivant.

Architecture	Paramètres (hors sortie)	Paramètres (avec sortie)
RNN simple		
GRU		
LSTM		

✅

$h(d + h + 1) = 64 \times (10 + 64 + 1) = 64 \times 75 = 4\,800$.
$p(h + 1) = 5 \times 65 = 325$.

Architecture	Hors sortie	Avec sortie
RNN simple	4 800	5 125
GRU	14 400	14 725
LSTM	19 200	19 525

6. La LSTM a davantage de paramètres que le GRU, mais est-elle toujours plus performante ? Dans quel cas préférerait-on le GRU ?

✅

Non, une plus grande capacité ne garantit pas de meilleures performances. Le GRU est souvent préféré dans les situations suivantes :

- **Données peu abondantes** : moins de paramètres réduit le risque de surapprentissage.
- **Contrainte de temps ou de mémoire** : le GRU est environ 25 % plus rapide à entraîner que la LSTM pour une même taille de couche cachée.
- **Séquences de longueur modérée** : la LSTM apporte un avantage surtout pour des dépendances très longues ; sur des séquences courtes, les deux architectures sont comparables.

En pratique, le choix se fait souvent par validation croisée sur la tâche cible.