Adapter un modèle selon nos préférences

Introduction à l'intelligence artificielle générative

Théo Lopes Quintas

BPCE Payment Services,
Université Paris Dauphine

2025-2026

Introduction

Orange Mécanique - Stanley Kubrick

;

En résumé

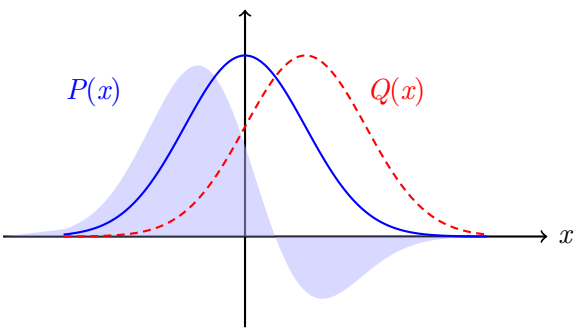
Pour deux mesures de probabilité P et Q on définit :

$$\mathrm{KL}(P\|Q) = \mathop{\mathbb{E}}_{y\sim P}\left[\log\left(\frac{P(y)}{Q(y)}\right)\right] \qquad \text{(Kullback-Leibler)}$$

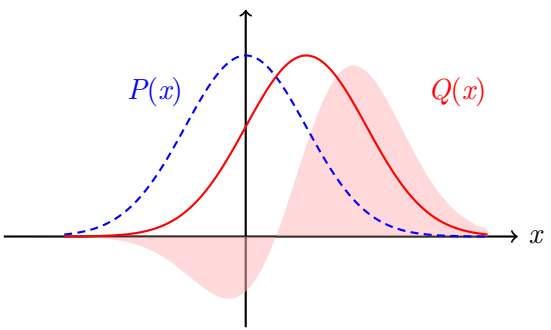
Cette quantité est la **divergence de Kullback-Leibler** qui est strictement positive ou nulle quand $P = Q$ presque partout. Elle mesure le **changement de distribution** entre P et Q, plus précisément l'erreur que l'on fait en approchant la véritable distribution P par la distribution Q.

Une identité que nous exploiterons plus tard est la suivante :

$$\arg\max_{\pi}\left\{\mathop{\mathbb{E}}_{\pi}[f(y)] - \beta\mathrm{KL}(\pi\|\pi_0)\right\} \implies \pi^*(y) = \frac{\pi_0(y)\exp\left(\frac{f(y)}{\beta}\right)}{Z} \qquad \text{(Gibbs variational principle)}$$



(a) KL($P \| Q$)



(b) KL($Q \| P$)

Figure – Kullback-Leibler divergence entre deux gaussienne

L'optimisation sous contrainte de fonction convexe peut être approché sous le prisme de la dualité pour passer de la notion de contraintes à pénalités.

Cadre convexe

On cherche à résoudre :

$$\max_x g(x) \text{ tel que } h(x) \leqslant c \qquad \text{(Primal)}$$

On peut reformuler le problème primal en problème dual :

$$\min_{\lambda \geqslant 0} \sup_x g(x) - \lambda \left(h(x) - c\right) \qquad \text{(Dual)}$$

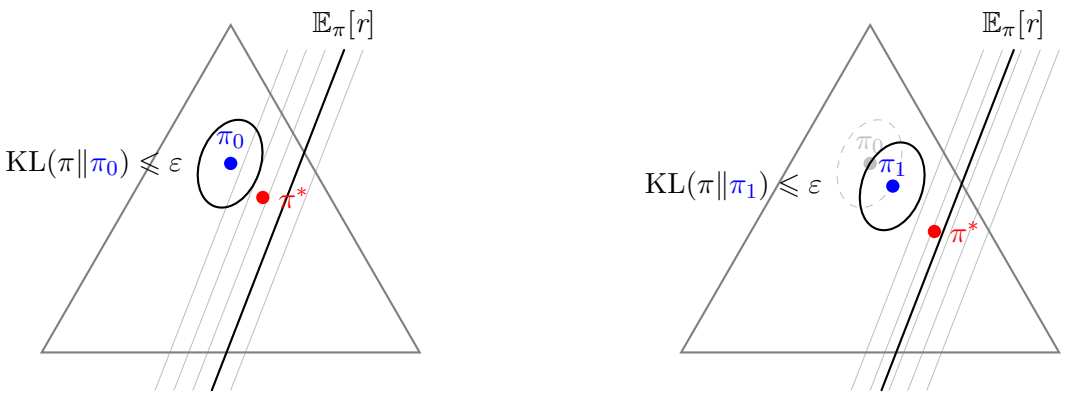
Application

On cherche à résoudre :

$$\max_\pi \mathop{\mathbb{E}}_\pi [r] \text{ tel que } \mathrm{KL}(\pi \| \pi_0) \leqslant \varepsilon \qquad \text{(Primal)}$$

On peut reformuler le problème primal en problème dual, pour $\beta > 0$:

$$\max_\pi \mathop{\mathbb{E}}_\pi [r] - \beta \mathrm{KL}(\pi \| \pi_0) \qquad \text{(Dual)}$$



(a) Etape 1 **(b)** Etape 2

Figure – Deux étapes de mise à jour de politique sous une contrainte de KL-divergence

Il est à noter que les frontière issue d'une divergence de Kullback-Leibler n'est pas forcément symétrique, mais reste convexe.

Préliminaire mathématiques

Méthode policy-gradients

Dans le cadre de l'apprentissage par renforcement, on travaille ici avec une politique paramétrique $\pi_\theta(y|x)$ et un objectif $J(\theta) = \underset{y\sim\pi_{\theta_0}}{\mathbb{E}}[R(y)]$. L'objectif de la méthode REINFORCE [Williams, 1992] est de calculer :

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int \pi_{\theta_0}(y) R(y)\, dy \\
&= \int \nabla_\theta \pi_{\theta_0}(y) R(y)\, dy \qquad \text{par règle de Leibniz} \\
&= \int \frac{\nabla_\theta \pi_{\theta_0}(y)}{\pi_{\theta_0}} \pi_{\theta_0} R(y)\, dy \qquad \text{et } \nabla_\theta \pi_{\theta_0}(y) = \pi_{\theta_0} \nabla_\theta \log\left(\pi_{\theta_0}(y)\right) \\
&= \int \pi_{\theta_0} \nabla_\theta \log\left(\pi_{\theta_0}(y)\right) R(y)\, dy \\
&= \underset{y\sim\pi_{\theta_0}}{\mathbb{E}} \left[\nabla_\theta \log\left(\pi_{\theta_0}(y)\right) R(y)\right]
\end{aligned}$$

Afin de réduire la variance, on peut prouver l'identité pour n'importe quelle fonction b :

$$\nabla_\theta J(\theta) = \underset{y\sim\pi_{\theta_0}}{\mathbb{E}} \left[\nabla_\theta \log\left(\pi_{\theta_0}(y)\right) \left(R(y) - b\right)\right] \qquad \text{où fréquemment } b = \underset{\pi_{\theta_0}}{\mathbb{E}}[R]$$

En résumé

Adapter à sa préférence

Motivation

L'entraînement d'un modèle de langage se réalise en deux phases :

1. **Pré-entraînement** : le modèle apprend à l'aide d'un très large corpus documentaire
2. **Le fine-tuning** : le modèle obtenu à la première étape est aligné avec nos préférences.

Pourquoi a-t-on besoin d'un ré-entraînement ? Ne pourrions-nous pas le faire dès l'étape 1 ?

1. **Objectif** : le pré-entraînement répond à la question *Quel est le prochain token le plus probable ?* alors que le fine-tuning répond à l'espérance de l'utilisateur *Quel est la réponse la plus concise / utile / polie ...*
2. **Distribution** : pour un pré-entraînement on utilise un corpus général, pour un fine-tuning un corpus plus spécifique
3. **Préférence** : Plusieurs réponses peuvent être correcte mais certaine seront inacceptables - trop longue, trop informelle, avec problème d'éthique / légalité...

Adapter à sa préférence

PPO : Proximal Policy Optimization

[Schulman et al., 2017] introduit la méthode PPO. A partir d'un prompt x, d'une réponse y, d'un modèle pré-entraîné π_{θ_0} et d'un modèle de récompense $r(x, y) \in \mathbb{R}$, on cherche à résoudre le problème :

$$\max_{\pi} \mathop{\mathbb{E}}_{\pi}[r] \quad \text{tel que} \quad \mathop{\mathbb{E}}_{x}[\mathrm{KL}(\pi \| \pi_0)] \leqslant \varepsilon \tag{PPO}$$

En utilisant la formulation duale pour $\beta > 0$, on obtient :

$$\mathcal{L}(\pi) = \mathop{\mathbb{E}}_{\pi}[r] - \beta \mathop{\mathbb{E}}_{x}[\mathrm{KL}(\pi \| \pi_0)] \tag{PPO dual}$$

On sait que la solution de ce type de problème est de la forme :

$$\pi^*(y|x) \propto \pi_0(y|x) \exp\left(\frac{r(x, y)}{\beta}\right)$$

On va approcher cette solution π^* en prenant le gradient :

$$\nabla_\theta \mathcal{L} = \mathop{\mathbb{E}}_{\pi_{\theta_0}}\left[\left(r(x, y) - \beta \log \frac{\pi_\theta}{\pi_{\theta_0}}\right) \nabla_\theta \log \pi_{\theta_0}\right] = \mathop{\mathbb{E}}_{\pi_{\theta_0}}\left[A(x, y) \nabla_\theta \log \pi_{\theta_0}\right]$$

Adapter à sa préférence

PPO : Proximal Policy Optimization

$$\nabla_\theta \mathcal{L} = \mathop{\mathbb{E}}_{\pi_{\theta_0}} \left[\left(r(x,y) - \beta \log \frac{\pi_\theta}{\pi_{\theta_0}} \right) \nabla_\theta \log \pi_{\theta_0} \right] = \mathop{\mathbb{E}}_{\pi_{\theta_0}} \left[A(x,y) \nabla_\theta \log \pi_{\theta_0} \right]$$

Appliquer naïvement ce gradient peut amener à des mises à jour trop importante, ne respectant pas la condition $\mathop{\mathbb{E}}_{x} [\mathrm{KL}(\pi \| \pi_0)] \leqslant \varepsilon$. On se propose de faire des mises à jour avec *l'ancienne* politique.

On définit $p_\theta(x,y) = \dfrac{\pi_\theta(x,y)}{\pi_{\theta_{old}}(x,y)}$ et avec l'échantillonnage préférentiel : $\mathop{\mathbb{E}}_{\pi_{\theta_0}} [r] \approx \mathop{\mathbb{E}}_{\pi_{\theta_{old}}} [\rho_\theta(x,y) A(x,y)]$.

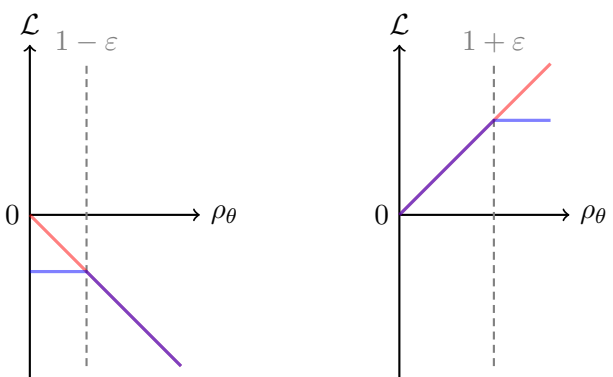
Nous n'avons cependant pas encore régler notre problème de magnitude de mise à jour, PPO utilise la fonction clip :

$$\mathcal{L}_{PPO}(\theta) = \mathop{\mathbb{E}}_{\pi_{old}} \left[\min \left(\rho_\theta A, \mathrm{clip}(\rho_\theta, 1-\varepsilon, 1+\varepsilon) A \right) \right]$$

Adapter à sa préférence

PPO : Proximal Policy Optimization

$$\mathcal{L}_{PPO}(\theta) = \mathop{\mathbb{E}}_{\pi_{old}} \left[\min \left(\rho_\theta A, \text{clip}(\rho_\theta, 1-\varepsilon, 1+\varepsilon) A \right) \right]$$



(a) $A < 0$ **(b)** $A > 0$

Figure – Fonction de perte clippé pour PPO

PPO nécessite d'avoir un modèle de récompense efficace pour permettre l'apprentissage. Il est typiquement obtenu via du RLHF ce qui est coûteux et difficile à mettre en place.

Adapter à sa préférence

DPO : Direct Preference Optimization

[Rafailov et al., 2023] introduit la méthode DPO dont l'objectif est de se séparer d'un modèle de récompense en le remplaçant par une **préférence**. Nous avons maintenant un dataset :

$$\mathcal{D} = \left\{(x, y^+, y^-) | y^+ \succ y^-\right\}$$

Le modèle logistique de Bradley-Terry pour la préférence suppose qu'il existe une fonction de récompense r telle que :

$$\mathbb{P}(y^+ \succ y^- | x) = \sigma\left(r(x, y^+) - r(x, y^-)\right) \qquad \text{(Bradley-Terry)}$$

On cherche à résoudre le même problème que PPO, dont on connait la forme de solution optimale :

$$\pi^*(y|x) \propto \pi_0(y|x) \exp\left(\frac{r(x, y)}{\beta}\right)$$

Adapter à sa préférence

DPO : Direct Preference Optimization

En prenant le ratio de cette solution :

$$\begin{aligned}
\frac{\pi^*(y^+|x)}{\pi^*(y^-|x)} &= \frac{\pi_0(y^+|x)}{\pi_0(y^-|x)} \exp\left(\frac{r(x,y^+) - r(x,y^-)}{\beta}\right) && \text{en prenant le logarithme} \\
r(x,y^+) - r(x,y^-) &= \beta\left(\log\frac{\pi^*(y^+|x)}{\pi^*(y^-|x)} - \log\frac{\pi_0(y^+|x)}{\pi_0(y^-|x)}\right) && \text{et dans le modèle Bradley-Terry} \\
\mathbb{P}(y^+ \succ y^-|x) &= \sigma\left(\beta\left[\log\frac{\pi^*(y^+|x)}{\pi^*(y^+|-)} - \log\frac{\pi_0(y^+|x)}{\pi_0(y^+|-)}\right]\right)
\end{aligned}$$

D'où on obtient la fonction de perte pour la méthode DPO :

$$\mathcal{L}_{DPO} = \mathbb{E}\left[\log\sigma\left(\beta\left[\log\frac{\pi^*(y^+|x)}{\pi^*(y^+|-)} - \log\frac{\pi_0(y^+|x)}{\pi_0(y^+|-)}\right]\right)\right]$$

Adapter à sa préférence

GRPO : Group-Relative Policy Optimization

[Shao et al., 2024] introduit la méthode GRPO pour améliorer les capacités de raisonnement mathématiques des modèles de langages. Cette fois, nous avons un dataset pour chaque prompt x :

$$\mathcal{D}(x) = \left\{ (\, y_i \,,\, r_i \,) \mid 1 \leqslant i \leqslant K \right\}$$

r_i : Récompense $r(x, y_i)$

$y_i \sim \pi_{\theta_{old}}$

Rappel : PPO

Avec l'échantillonnage préférenciel :
$\mathop{\mathbb{E}}_{\pi_{\theta_0}}[r] \approx \mathop{\mathbb{E}}_{\pi_{\theta_{old}}}[\rho_\theta(x,y)A(x,y)]$

Nous avions obtenu :

$$\mathcal{L}_{PPO}(\theta) = \mathop{\mathbb{E}}_{\pi_{old}}[\min(\rho_\theta A, \text{clip}(\rho_\theta, 1-\varepsilon, 1+\varepsilon)A)]$$

Nouveauté de GRPO

On introduit $\overline{r}(x) = \frac{1}{K}\sum_{i=1}^{K} r_i(x, y_i)$ et on définit $A_i = r_i - \overline{r}$. La fonction de perte devient :

$$\mathcal{L}_{GRPO}(\theta) = \mathop{\mathbb{E}}_{\pi_{old}}\left[\frac{1}{K}\sum_{i=1}^{K} \min(\rho_\theta A_i, \text{clip}(\rho_\theta, 1-\varepsilon, 1+\varepsilon)A_i)\right]$$

En résumé

Adapter partiellement à sa préférence

Motivation

[Li et al., 2018] montre que les réseaux de neurones en général ont un espace des paramètres qui est en réalité effectivement de rang inférieur. Puisque les modèles de langage en sont, [Aghajanyan et al., 2020] le démontre.

Partant de ce résultat, on peut se poser la question de réentraîner **tous** les poids.
Une **partie** pourrait suffire.

Adapter partiellement à sa préférence

LoRA : Low-Rank Adaptation

[Hu et al., 2021] introduit LoRA qui se résume comme suit :

1. Figer la matrice de poids W_0 du modèle
2. Initialiser une matrice A selon une loi normale et une matrice B valant 0
3. Calculer l'output à partir d'un input via la multiplication de la matrice $W_0 + \frac{\alpha}{r}AB$

Le terme $\frac{\alpha}{r}$ permet de contrôler la magnitude de modification des poids et s'assurer que la magnitude de AB n'est pas proportionnelle à r.

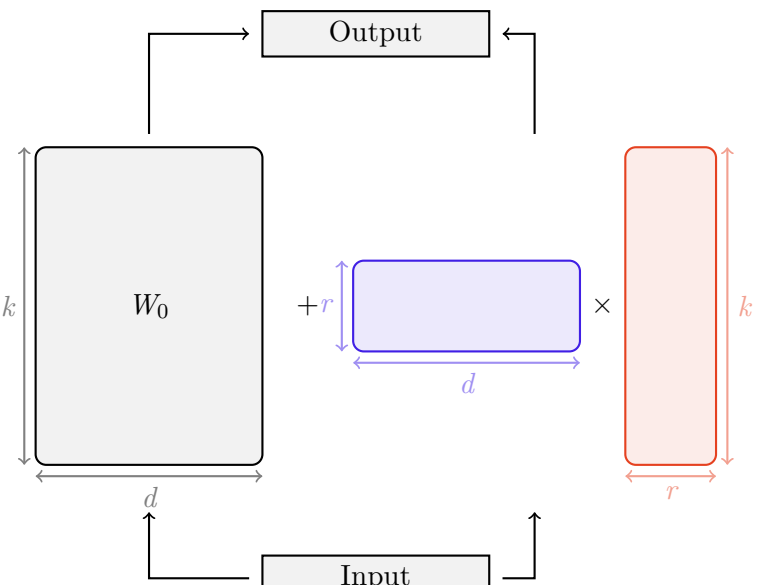


Figure – Schéma du fonctionnement de LoRA

Adapter partiellement à sa préférence

LoRA : Low-Rank Adaptation

L'intérêt de la méthode LoRA est de réduire le nombre de paramètre à l'entraînement. Vérifions-le avec le modèle contemporain à l'article : GTP3

- $d_{model} = 12288$ et 96 couches
- Nombre de paramètre : 175 milliards
- Poids des couches d'attentions : $96 \times (4 \times d^2_{model}) = 58$ milliards

LoRA ne s'applique que sur les 4 couches d'attentions et introduit $2rd$ paramètres pour chaque couches. Ainsi :

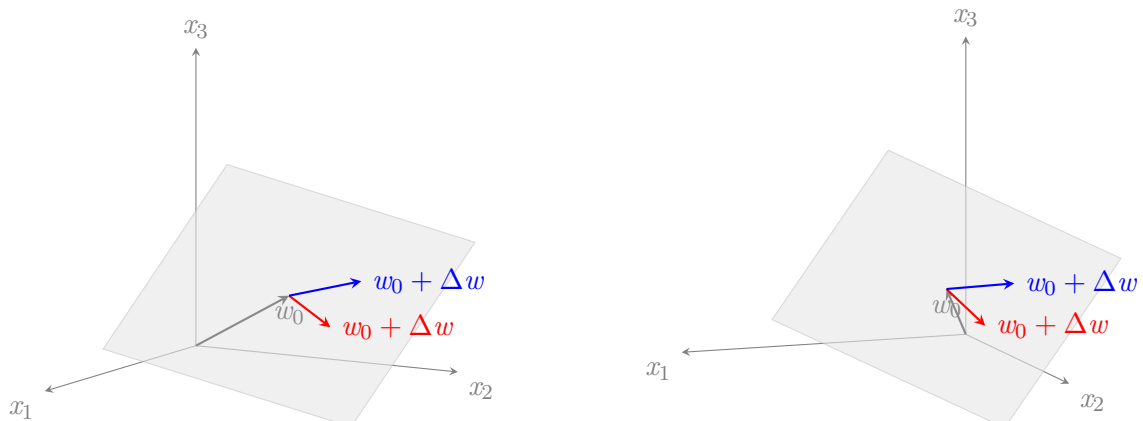
Rang r	Nombre de paramètre LoRA (millions)
1	9.4
2	18.9

Concrètement, cela permet par exemple :

- Avoir plusieurs modèles avec des comportements différents en changeant les matrices A et B, moins imposante en stockage
- Permettre au grand publique d'ajuster des modèles de langage avec des GPU *standard*

Adapter partiellement à sa préférence

LoRA : visuellement



(a) Vue 1 **(b)** Vue 2

Figure – Deux exemples de mise à jour avec LoRA pour $r = 2$

L'optimisation peut être accélérée en suivant une approche *quantizé* [Dettmers et al., 2023], notamment pour réduire l'impact mémoire.

Adapter partiellement à sa préférence

DoRA : Weight-Decomposed Low-Rank Adaptation

[Liu et al., 2024] introduit la méthode DoRA, liée à la méthode LoRA. A partir des paramètres du modèle initial $W_0 \in \mathcal{M}_{d,k}$ on construit deux matrices A et B selon la méthode LoRA, puis on obtient les paramètres du *nouveau* modèle :

$$W = m \times \frac{W_0 + AB}{\|W_0 + AB\|} \qquad \text{(DoRA)}$$

Magnitude $m \in \mathbb{R}^d$

Si l'on regarde le gradient de la méthode DoRA pour un vecteur $w \in \mathbb{R}^d$, en notant $\tilde{w} = w_0 + AB$:

$$\frac{\partial w}{\partial \tilde{w}} = \frac{m}{\|\tilde{w}\|} \left(I - \frac{\tilde{w}}{\|\tilde{w}\|} \left(\frac{\tilde{w}}{\|\tilde{w}\|} \right)^T \right)$$

Cette fois, les matrices A et B ne peuvent pas changer la magnitude, uniquement permettre une rotation, en ajoutant simplement un paramètre par dimension.

Adapter partiellement à sa préférence

DoRA : Weight-Decomposed Low-Rank Adaptation

Comme LoRA, l'intérêt de la méthode DoRA est de réduire le nombre de paramètre à l'entraînement. Vérifions-le à nouveau avec GPT-3.

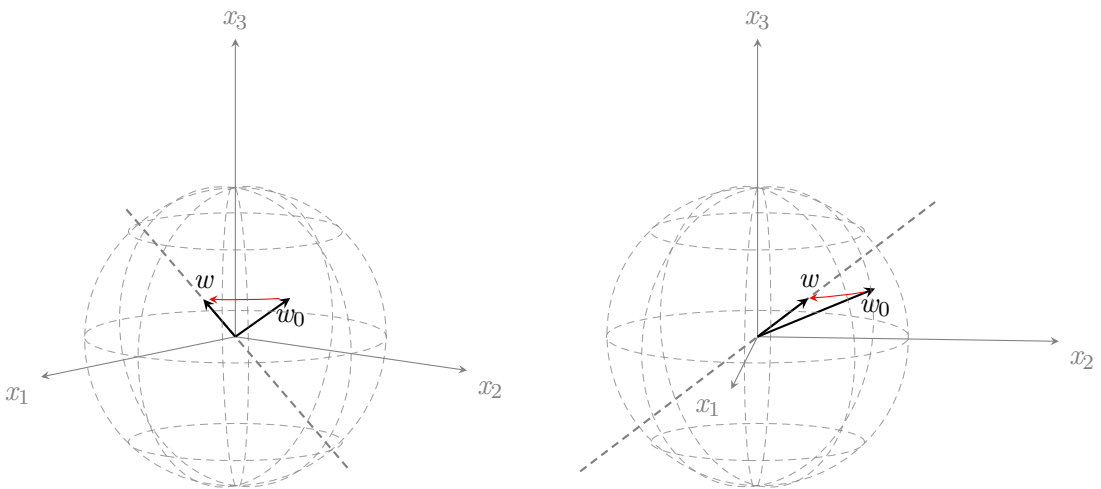
- $d_{model} = 12288$ et 96 couches
- Nombre de paramètre : 175 milliards
- Poids des couches d'attentions : $96 \times (4 \times d^2_{model}) = 58$ milliards

DoRA ne s'applique que sur les 4 couches d'attentions et introduit $2rd + d$ paramètres pour chaque couches. Ainsi :

	Nombre de paramètre (millions)		% de GPT3	
Rang r	LoRA	DoRA	LoRA	DoRA
1	9.4	14.2	0.005	0.008
2	18.9	23.6	0.011	0.013

Adapter partiellement à sa préférence

DoRA : visuellement



(a) Vue 1 **(b)** Vue 2

Figure – Deux exemples de mise à jour avec DoRA pour $r = 2$, sans la mise à jour de magnitude

En résumé

Bibliographie I

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. (2020).
Intrinsic dimensionality explains the effectiveness of language model fine-tuning.
In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1 : long papers).*

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023).
Qlora : Efficient finetuning of quantized llms.
Advances in neural information processing systems.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2021).
Lora : Low-rank adaptation of large language models.
ICLR, 1(2) :3.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018).
Measuring the intrinsic dimension of objective landscapes.
arXiv preprint arXiv :1804.08838.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. (2024).
Dora : Weight-decomposed low-rank adaptation.
In *Forty-first International Conference on Machine Learning.*

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. (2023).
Direct preference optimization : Your language model is secretly a reward model.
Advances in neural information processing systems.

Bibliographie II


Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017).
Proximal policy optimization algorithms.
arXiv preprint arXiv :1707.06347.


Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. (2024).
Deepseekmath : Pushing the limits of mathematical reasoning in open language models.
arXiv preprint arXiv :2402.03300.


Williams, R. J. (1992).
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine learning, 8(3) :229–256.