

DESCENTE DE GRADIENT ET VARIANTES
SÉMINAIRE MATHS POUR ML

Théo Lopès-Quintas

BPCE Payment Services,
Université Paris Dauphine

16 novembre 2023

DESCENTE DE GRADIENT

FORMULATION D'UN PROBLÈME DE MACHINE LEARNING

Dans le cadre supervisé, nous avons accès à un dataset \mathcal{D} défini comme :

$$\mathcal{D} = \left\{ (x_i, y_i) \mid \forall i \leq n, x_i \in \mathbb{R}^{d'}, y_i \in \mathcal{Y} \right\}$$

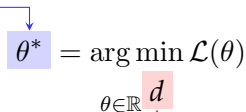
Nombre d'observations

Nombre d'informations

Avec $\mathcal{Y} \subseteq \mathbb{R}$ pour un problème de régression et $\mathcal{Y} \subset \mathbb{N}$ dans le cadre d'une classification. Les problèmes de Machine Learning supervisé peuvent souvent s'écrire sous la forme d'une optimisation d'une fonction de perte $\mathcal{L} : \mathbb{R}^d \times \mathcal{M}_{n,d'} \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ comme :

Vecteur des paramètres optimaux

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta)$$



Dimension du vecteur de paramètres

Dans la suite, pour simplifier les notations, nous omettrons la dépendance de \mathcal{L} en X et y . Notons qu'en général, nous avons $d \neq d'$ et dans le cas du deep learning, très souvent $d \gg d'$.

DESCENTE DE GRADIENT

DESCENTE CLASSIQUE

La méthode la plus utilisée pour résoudre ce genre de problème est la descente de gradient :

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t)$$

↑
Learning rate

La convergence est garantie dans le cas où \mathcal{L} est convexe. Ce n'est pas le cas dans un réseau de neurones, mais il est montré [Choromanska et al., 2015] que la majorité des minima locaux d'un réseau de neurones sont *proches* en terme de valeur optimale. Ainsi, obtenir un minima local est satisfaisant.

Quand on travaille avec des grands datasets, le coût de calcul/temps est grand si l'on calcule $\nabla \mathcal{L}(\theta_t)$ pour la totalité de la base. Ainsi, on préfère la descente de gradient **stochastique** par batch : on applique la descente de gradient à l'ensemble de la base de données en découpant en petit lots (batch) aléatoires et en mettant à jour θ à chaque fois.

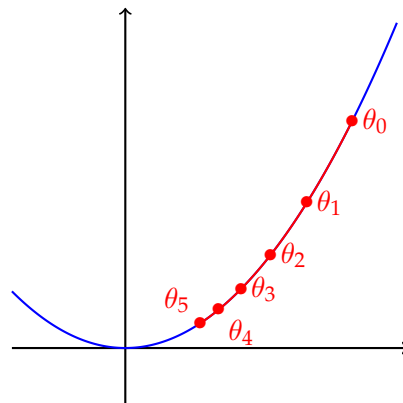


Figure – Exemple d'une descente de gradient pour $f(x) = x^2$

DESCENTE DE GRADIENT

DESCENTE AVEC MOMENTUM

Paramètre du momentum

$$\begin{cases} v_{t+1} &= \beta v_t + (1 - \beta) \nabla \mathcal{L}(w_t) \\ w_{t+1} &= w_t - \eta_t v_{t+1} \end{cases} \quad (1)$$

Vecteur de vélocité

Avec cette version, on conserve dans la mise à jour des poids la *tendance* de déplacement des poids dans l'espace des paramètres pour accélérer la descente.

La volonté d'accélérer la descente permet de gagner des heures voire des jours d'entraînements sur des réseaux de neurones massifs (type Transformers). Cependant, la descente de gradient avec momentum peut parfois *rater* le minimum et faire machine arrière. Ce phénomène peut être mitigé à l'aide d'un choix précis du *learning rate*.

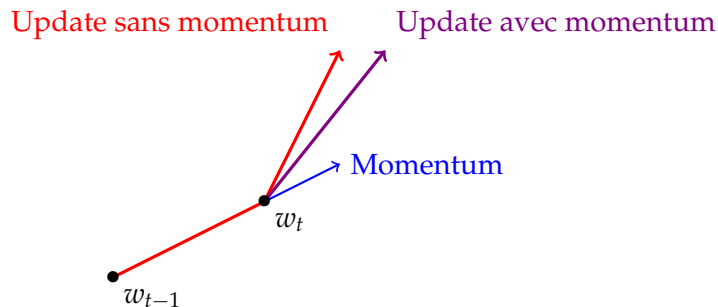


Figure – Effet du momentum sur la mise à jour des poids

DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

ADAGRAD : UNE PREMIÈRE RÉPONSE

$$\begin{cases} g_{t+1} = g_t + \nabla \mathcal{L}(\theta_t)^2 & \text{avec } g_0 = 0 \\ \theta_{t+1} = \theta_t - \eta \frac{\nabla \mathcal{L}(\theta_t)}{\sqrt{g_{t+1}} + \varepsilon} \end{cases} \quad (\text{AdaGrad, 2011})$$

Nombre pour éviter des problèmes numériques

Si le gradient a une magnitude plus importante dans une direction, elle sera privilégiée pendant l'optimisation.

AdaGrad [Duchi et al., 2011] propose de normaliser le gradient pour avoir un apprentissage *uniforme*.

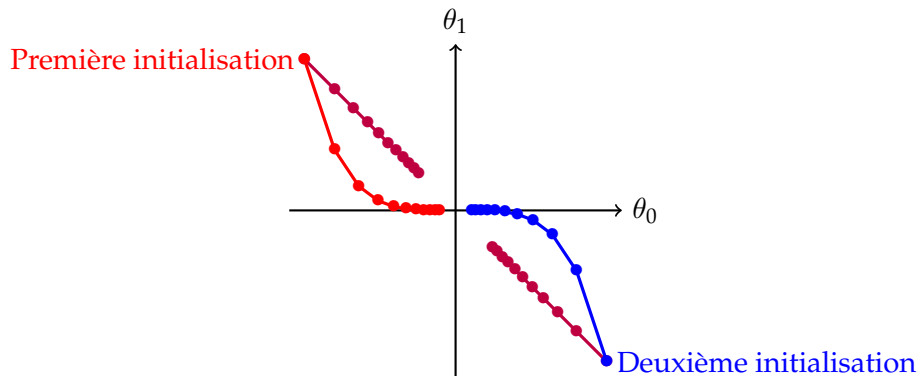


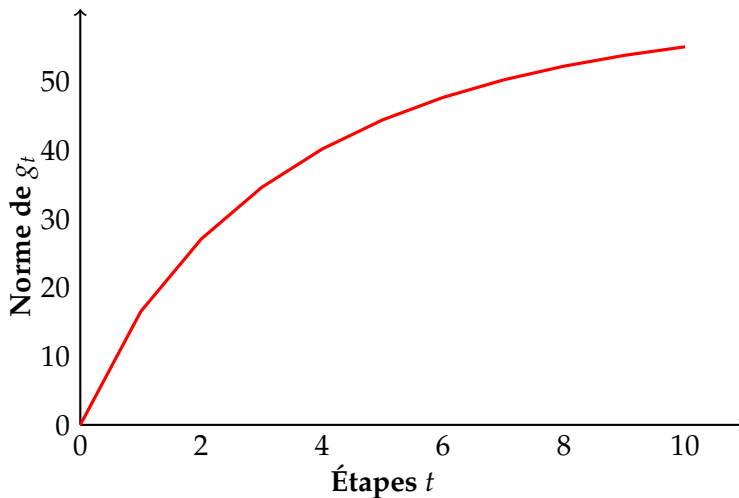
Figure – 10 étapes de descente de gradient pour la fonction $f(x, y) = x^2 + 3y^2$ avec **AdaGrad** pour deux initialisations différentes

DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

ADAGRAD : RÉPONSE IMPARFAITE

$$\begin{cases} g_{t+1} = g_t + \nabla \mathcal{L}(\theta_t)^2 & \text{avec } g_0 = 0 \\ \theta_{t+1} = \theta_t - \eta \frac{\nabla \mathcal{L}(\theta_t)}{\sqrt{g_{t+1}} + \epsilon} \end{cases} \quad (\text{AdaGrad, 2011})$$

Cependant avec ce schéma, AdaGrad tend à avoir un apprentissage qui ralentit au fil de l'entraînement puisque $(g_t)_t$ est croissante.



DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

RMSPROP : DEUXIÈME RÉPONSE

Contrôle la *mémoire* des précédents gradients, $\alpha \geq 0$

$$\begin{cases} g_{t+1} = \alpha g_t + (1 - \alpha) \nabla \mathcal{L}(\theta_t)^2 & \text{avec } g_0 = 0 \\ \theta_{t+1} = \theta_t - \eta \frac{\nabla \mathcal{L}(\theta_t)}{\sqrt{g_{t+1}} + \varepsilon} \end{cases} \quad (\text{RMSProp, 2012})$$

RMSProp [Hinton et al., 2012] cherche aussi à permettre un apprentissage uniforme dans toutes les directions, mais le fait avec une moyenne mobile exponentielle. Cela permet à $(g_t)_t$ de pouvoir décroître.

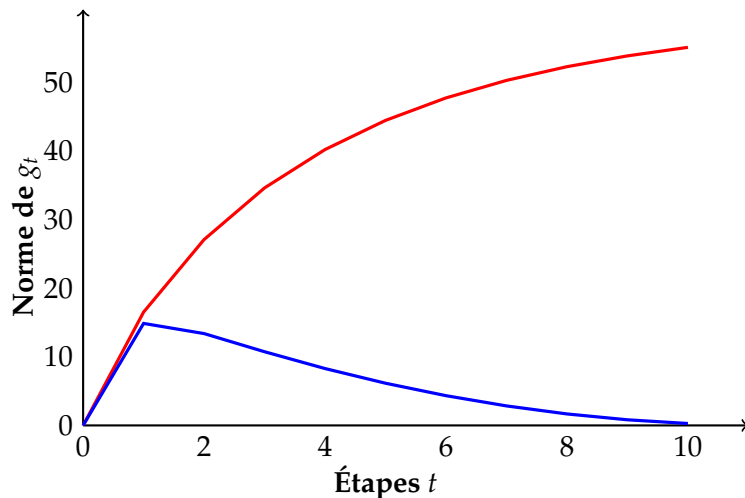


Figure – Norme de g_t pour RMSProp et AdaGrad

DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

ADAM : COMBINER ADAGRAD ET RMSPROP

$$\left\{ \begin{array}{lcl} m_{t+1} & = & \beta_1 m_t + (1 - \beta_1) \nabla \mathcal{L}(\theta_t) \quad \text{avec } m_0 = 0 \\ \hat{m}_{t+1} & = & \frac{m_{t+1}}{1 - \beta_1^{t+1}} \\ v_{t+1} & = & \beta_2 v_t + (1 - \beta_2) \nabla \mathcal{L}(\theta_t)^2 \quad \text{avec } v_0 = 0 \\ \hat{v}_{t+1} & = & \frac{v_{t+1}}{1 - \beta_2^{t+1}} \\ \theta_{t+1} & = & \theta_t - \eta \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1} + \varepsilon}} \end{array} \right. \quad (\text{Adam, 2014})$$

Adam [Kingma and Ba, 2015] ressemble à une version accélérée de RMSProp, mais elle corrige le biais introduit par l'initialisation des moyennes mobiles. Adam s'est rapidement imposé comme un excellent schéma d'optimisation pour l'apprentissage d'un réseau de neurones.

Un de ses avantages est que $|\theta_{t+1} - \theta_t| \lesssim \eta$ donc permet une définition plus éclairée du learning rate.

DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

PROBLÈME DE LA RÉGULARISATION

[Krogh and Hertz, 1991] montre qu'avoir un réseau de neurones avec une magnitude de poids faible permet de limiter le sur-apprentissage.

Une manière classique pour régulariser un réseau de neurones est de modifier la fonction de perte que l'on optimise. Le plus souvent on exploite la régularisation \mathcal{L}_2 :

$$\mathcal{L}_\lambda(w) = \mathcal{L}(w) + \frac{\lambda}{2} \|w\|_2^2 \quad \text{avec } \lambda \geq 0$$

Dans le cas d'une descente de gradient classique, cela donne le schéma d'optimisation :

$$\begin{aligned} w_{t+1} &= w_t - \eta_t \nabla \mathcal{L}_\lambda(w_t) && \text{par définition de la descente de gradient} \\ &= w_t - \eta_t \nabla \mathcal{L}(w_t) - \eta_t \lambda w_t && \text{par définition de } \mathcal{L}_\lambda \end{aligned}$$

Le terme supplémentaire $\eta_t \lambda w_t$ est appelé le weight decay.

DESCENTE DE GRADIENT AVEC PAS ADAPTATIF

PROBLÈME DE LA RÉGULARISATION

Mais si l'on considère une descente de gradient avec momentum :

$$\begin{aligned}v_{t+1} &= \beta v_t + (1 - \beta) \nabla \mathcal{L}_\lambda(w_t) && \text{par définition} \\&= \beta v_t + (1 - \beta) [\nabla \mathcal{L}(w_t) + \lambda w_t] && \text{par définition de } \mathcal{L}_\lambda\end{aligned}$$

$$\begin{aligned}w_{t+1} &= w_t - \eta_t (\beta v_t + (1 - \beta) [\nabla \mathcal{L}(w_t) + \lambda w_t]) \\&= w_t - \eta_t \beta v_t - \eta_t (1 - \beta) \nabla \mathcal{L}(w_t) - \eta_t (1 - \beta) \lambda w_t\end{aligned}$$

Autrement dit, nous avons une propagation de la régularisation dans le schéma de descente, ce qui n'est pas souhaité. Pour conserver le comportement observé dans une descente de gradient classique, [Loshchilov and Hutter, 2019] propose simplement de modifier le schéma de descente plutôt que la fonction de perte.

Ainsi, dès lors que l'on utilise une méthode *adaptive*, il faut préférer le **weight decay** à la régularisation \mathcal{L}_2 . L'omniprésence d'Adam dans la littérature a amené à la création d'AdamW qui est Adam intégrant le weight decay.

COMMENT CHOISIR LE PAS DE DESCENTE ?

ÉCHÉANCIERS CLASSIQUES

Les succès de l'ensemble des schémas précédents sont conditionnés à un bon choix du learning rate η , même dans le cas adaptatif. Il est possible de le conserver constant pour la totalité de l'entraînement mais il existe des **échéanciers** pour modifier η au cours de l'entraînement.

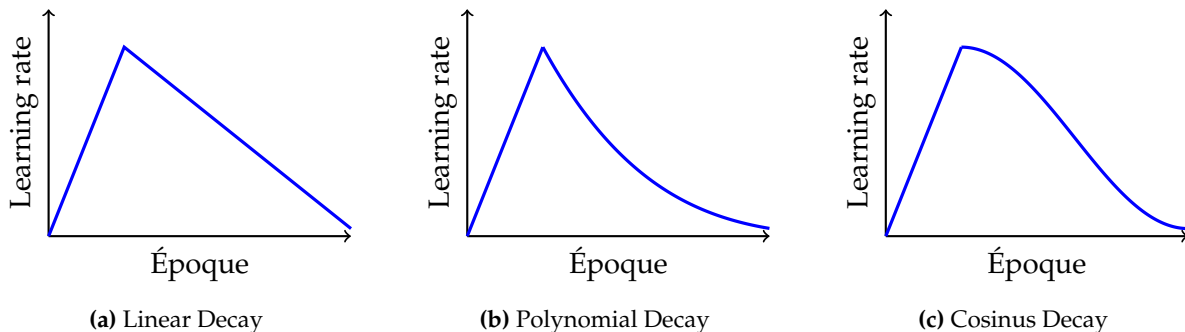







Figure – Exemples d'échéanciers du learning rate avec échauffements

L'échéancier cosinus fait partie d'un plus grand ensemble : les échéanciers cycliques. Les *Large Language Models* utilisent l'échéancier cosinus mais sur un seul cycle entier comme préconisé par une étude de Deepmind [Hoffmann et al., 2022].

BIBLIOGRAPHIE I

-  Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015).
The loss surfaces of multilayer networks.
In *Artificial intelligence and statistics*. PMLR.
-  Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research.
-  Hinton, G., Srivastava, N., and Swersky, K. (2012).
Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.
Cited on.
-  Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022).
Training compute-optimal large language models.
arXiv preprint arXiv :2203.15556.
-  Kingma, D. P. and Ba, J. (2015).
Adam : A method for stochastic optimization.
In *International Conference on Learning Representations (ICLR)*.

BIBLIOGRAPHIE II



Krogh, A. and Hertz, J. (1991).

A simple weight decay can improve generalization.

Advances in neural information processing systems.



Loshchilov, I. and Hutter, F. (2019).

Decoupled weight decay regularization.

In International Conference on Learning Representations (ICLR).