ADAM: A method for Stochastic Optimization
Résumé d'article

Guillaume BERNARD-REYMOND, Guillaume BOULAND,
Camille MOTTIER, Abel SILLY

5 octobre 2024

Table des matières

1 Introduction

L'article « ADAM : A method for Stochastic Optimization [1] » a été publié en 2014 (et corrigé jusqu'en 2017) par Diederik P. Kingma (Université d'Amsterdam, OpenAI) et Jimmy Lei Ba (Université de Toronto) dans le cadre de l'International Conference on Learning Representations (ICLR) de 2015.

Cet article présente l'algorithme Adam, algorithme d'optimisation stochastique, basé sur une descente de gradient, dans le cadre d'un espace de paramètres à grande dimension. Outre le fait que cet algorithme est simple à implémenter, efficace computationnellement et nécessite peu de mémoire, il semble offrir une méthode qui marche bien dans un large panel de cas, y compris dans les cas problématiques de gradients éparses ou de fonctions-objectifs non stationnaires. En cela, il combine les qualités d'algorithmes existants au préalable, tels que AdaGrad et RMSProp.

L'article présente une description précise de l'algorithme Adam, fournit un résultat de convergence de la méthode et aborde l'apport de l'algorithme Adam vis-à-vis d'autres algorithmes.

2 Algorithme Adam

2.1 Objectif

On considère une fonction-objectif stochastique $f(\theta)$ de paramètres θ, qu'on suppose différentiable. L'algorithme Adam est une méthode d'ordre 1 (c'est-à-dire qui repose sur des évaluations de la fonctionnelle f et du gradient $\nabla_\theta f$), qui a pour objectif d'optimiser les paramètres θ afin de minimiser l'espérance $\mathbb{E}[f(\theta)]$.
L'aspect stochastique peut venir d'une fonction-objectif intrinsèquement bruitée ou bien d'un échantillonnage réalisé à chaque pas de l'algorithme. Typiquement, f peut être une fonction perte, de la forme
$f(\theta) = \displaystyle\sum_{i=1}^{n} \ell(x_i|\theta)$, où le calcul de gradient $\nabla_\theta f(\theta) = \displaystyle\sum_{i=1}^{n} \nabla_\theta \ell(x_i|\theta)$ est trop coûteux en nombre d'évaluations de gradients. Il est alors remplacé à chaque étape t par le calcul de $\nabla_\theta f_t(\theta) = \displaystyle\sum_{i\in I_t} \nabla_\theta \ell(x_i|\theta)$ pour I_t un sous-échantillon et f_t définie par $f_t(\theta) = \displaystyle\sum_{i\in I_t} \ell(x_i|\theta)$.

2.2 Description de l'algorithme Adam

Outre la fonction $f(\theta)$, l'algorithme nécessite la donnée d'un pas α, de taux $\beta_1, \beta_2 \in [0, 1[$, d'une constante de stabilisation numérique $\varepsilon > 0$ et de paramètres initiaux θ_0 (valeurs par défaut : $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$). Il exécute alors le schéma suivant :

Algorithme 1 : Adam

Entrées : $f(\theta)$, α, β_1, β_2, ε, θ_0
$m_0 \longleftarrow 0$
$v_0 \longleftarrow 0$
$t \longleftarrow 0$
tant que θ_t *ne converge pas* **faire**
    $t \longleftarrow t+1$
    $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \longleftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$
    $v_t \longleftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2$
    $\widehat{m}_t \longleftarrow m_t/(1-\beta_1^t)$
    $\widehat{v}_t \longleftarrow v_t/(1-\beta_2^t)$
    $\theta_t \longleftarrow \theta_{t-1} - \alpha\cdot \widehat{m}_t/(\sqrt{\widehat{v}_t}+\varepsilon)$
fin
Sorties : θ_t

Pour comprendre cet algorithme et identifier les apports de la méthode Adam, observons les différentes étapes et comparons-les avec celles d'autres algorithmes classiques de descente de gradient stochastique (présentés dans l'annexe).

- $m_t \longleftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$

  Fournit une estimation de $\mathbb{E}[g_t]$ par moyenne mobile à décroissance exponentielle : $m_t = (1-\beta_1)\displaystyle\sum_{i=1}^{t} \beta_1^{t-i}\cdot g_i$.
  Permet de garder mémoire des directions de descente précédentes afin d'atténuer les variations liées au bruit de la fonction.
  On trouve une idée similaire dans l'algorithme SGD avec moment.

- $v_t \longleftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2$
  Fournit une estimation de $\mathbb{E}[g_t^2]$, là aussi par moyenne mobile à décroissance exponentielle.
  Permettra, lors de la mise à jour des paramètres, une mise à l'échelle du gradient, c'est-à-dire qu'on ne va pas utiliser le même pas pour tous les paramètres. On ralentit le pas en cas de forte variation liée à un paramètre, et on l'accélère en cas de faible variation. On trouve une idée similaire dans les algorithmes AdaGrad (sans moyenne mobile) et RMSProp.

- $\widehat{m}_t \longleftarrow m_t/(1-\beta_1^t)$ et $\widehat{v}_t \longleftarrow v_t/(1-\beta_2^t)$
  Permet de réduire le biais vers 0 de m_t et v_t venant de l'initialisation de m_0 et v_0 à 0.
  C'est une innovation fournie par l'algorithme Adam.

On peut donc observer que l'algorithme Adam combine les idées fournies par les précédents algorithmes de descente de gradient, tels que la personnalisation des pas à chaque paramètre et la mémorisation du passé par le biais des moyennes mobiles, tout en apportant un élément supplémentaire : la correction des biais des moments d'ordre 1 et 2.

3 Efficacité et points forts de la méthode

3.1 Contraction du biais

La grande différence de l'algorithme Adam par rapport aux autres algorithmes de descente de gradients est fournie par les étapes déterminant $\widehat{m}_t$ et $\widehat{v}_t$, qui ont pour effet de contracter le biais des estimateurs des moments m_t de $\mathbb{E}[g_t]$ et v_t de $\mathbb{E}[g_t^2]$. En effet, nous avons :

$$\mathbb{E}[m_t] = (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}\mathbb{E}[g_i] = \mathbb{E}[g_t]\underbrace{(1-\beta_1^t)}_{<0} + \underbrace{(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}(\mathbb{E}[g_i]-\mathbb{E}[g_t])}_{\zeta}$$

avec ζ qui pourra être rendu petit par un bon choix de β_1. La division par $(1-\beta_1^t)$ permet donc de réduire le biais de m_t. Il en est de même pour v_t.

En particulier, en cas de gradients parcimonieux (sparse gradient) qui nécessitent une bonne mémoire du passé, donc des taux β_1 et β_2 grands, le biais obtenu peut être conséquent. Le calcul de $\widehat{m}_t$ et $\widehat{v}_t$ peut alors s'avérer pertinent.

3.2 Majoration du pas

Dans l'algorithme Adam, le pas à chaque étape est donné par : $\Delta_t = \alpha\cdot\widehat{m}_t/\sqrt{\widehat{v}_t+\varepsilon}$, où $\varepsilon > 0$ évite les divisions par des nombres trop petits. En supposant $\varepsilon = 0$, l'article donne la borne suivante du pas :

$$|\Delta_t| \leqslant \begin{cases} \alpha\cdot(1-\beta_1)/\sqrt{1-\beta_2} & \text{si } (1-\beta_1) > \sqrt{1-\beta_2} \\ \alpha & \text{sinon} \end{cases}$$

La première majoration est en fait utile dans le cadre de gradients parcimonieux. Dans les autres cas, on a une borne de l'ordre de α : $|\Delta_t| \lessapprox \alpha$.

Le choix de α permet donc d'avoir un contrôle sur la taille des pas effectués, et donc de définir une « zone de confiance » autour des paramètres θ_t dans laquelle on peut se déplacer à partir du calcul du gradient g_t.

3.3 Convergence

À reprendre
Nous considérons ici le regret de la méthode défini par :

$$R(T) = \sum_{t=1}^{T}[f_t(\theta_t) - f_t(\theta^*)]$$

où θ^* sont les paramètres optimaux (oracle)bof, pas vraiment ça vers lesquels on cherche à se diriger. Cette quantité croît avec le temps T, mais l'article fournit, sous certaines conditions de gradients bornés et de variations de paramètres bornées à détailler ?!, la relation suivante, qui garantie une croissance de $R(T)$ contrôlée :

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right)$$

faut-il parler de la convergence apportée par l'autre article ?

4 Visu

Sources

[1] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. https://arxiv.org/abs/1412.6980, 2017.

[2] Josh Starmer. Optimization for deep learning (momentum, rmsprop, adagrad, adam). https://www.youtube.com/watch?v=NE88eqLngkg, 2023.

5 Annexes

Algorithme 2 : SGD

Entrées : $f(\theta)$, α, θ_0

$t \longleftarrow 0$

tant que θ_t *ne converge pas* **faire**

- $t \longleftarrow t+1$
- $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$
- $\theta_t \longleftarrow \theta_{t-1} - \alpha \cdot g_t(\theta_{t-1})$

fin

Sorties : θ_t

Algorithme 3 : SGD avec moment (1964)

Entrées : $f(\theta)$, α, ρ, θ_0

$t \longleftarrow 0$

tant que θ_t *ne converge pas* **faire**

- $t \longleftarrow t+1$
- $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$
- $m_t \longleftarrow \rho \cdot m_{t-1} - \alpha \cdot g_t$
- $\theta_t \longleftarrow \theta_{t-1} + m_t$

fin

Sorties : θ_t

Algorithme 4 : AdaGrad (2011)

Entrées : $f(\theta)$, α, ε, θ_0

$v_0 \longleftarrow 0$

$t \longleftarrow 0$

tant que θ_t *ne converge pas* **faire**

- $t \longleftarrow t+1$
- $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$
- $v_t \longleftarrow v_{t-1} + g_t^2$
- $\theta_t \longleftarrow \theta_{t-1} - \alpha \cdot g_t / (\sqrt{v_t} + \varepsilon)$

fin

Sorties : θ_t

Algorithme 5 : RMSProp (2012)

Entrées : $f(\theta)$, α, β_2, ε, θ_0

$v_0 \longleftarrow 0$

$t \longleftarrow 0$

tant que θ_t *ne converge pas* **faire**

  $t \longleftarrow t + 1$

  $g_t \longleftarrow \nabla_\theta f_t(\theta_{t-1})$

  $v_t \longleftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$

  $\theta_t \longleftarrow \theta_{t-1} - \alpha \cdot g_t / (\sqrt{v_t} + \varepsilon)$

fin

Sorties : θ_t