Adam : A method for stochastic Optimization

Maxime Darrin

10 mars 2020

Table des matières

1 Introduction

Adam est une méthode originellement proposée par en 2015 par D.P Kingma et J. Lei Ba à la conférence ICLR. C'est une méthode d'optimisation du premier ordre introduisant une méthode de conservation du moment d'inertie (*momentum* en anglais) couplée à un *learning rate* adaptatif.

Il reprend le principe de l'*adaptative learning rate* utilisé par *Adadelta[?]* et *RMSProp[?]*, c'est à dire qu'il conserve une moyenne temporelle (l'importance du passé diminue exponentiellement avec le temps) du carré des gradients précédemment calculés de sorte à conserve une notion de variance des-dits gradients.

A cela, il ajoute une conservation de l'inertie. Il estime la moyenne des gradients (simples cette fois-ci) précédemment calculés, de même que précédemment en accordant plus d'importance aux observations récentes qu'au passé en faisant diminuer exponentiellement avec le temps l'importance du passé. L'idée est de conserver une notion de moyenne de la pente courante et de continuer à aller "un peu" dans les directions prises dans le passé.

Ces deux propriétés simulent en fait la trajectoire qu'une boule qui roulerait (avec de la friction) sur la surface d'erreur, aurait.

Dans un premier temps

2 Principes et justification

2.1 *Adaptavie learning rate*

Tout d'abord, on rappelle la méthode *RMS Prop* proposée par Geoff Hinton[?]. On maintient une estimation de la moyenne des carrés des coordonnées des gradients, c'est à dire de la variance non centrée. Et on l'utilise pour adapter le *learning rate*. Dans la suite on notera g_t le vecteur des gradient calculé au temps t et l'application des carrés se fait coordonnées à coordonnées.

$$
\begin{aligned}
E[g^2]_t &= 0.9E[g^2]_{t-1} + 0.1g_t^2 \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t
\end{aligned}
$$

FIGURE 1 – Mise à jour pour *RMSProp*

Adam reprend ce principe de manière plus générale : pour $\beta_2 \in [0, 1]$, on calcule $v_t = \beta_2 b_{t-1} + (1 - \beta_2) g_t^2$.

Les auteurs notes que cet estimateur est biaisé vers 0 en particulier lorsque β_2 est proche de 1. Ils proposent alors de redresser cet estimateur pour le rendre non biaisé :

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

2.2 Inertie

En plus de l'adaptation du *learning rate Adam* conserve de l'inertie dans sa descente de gradient, pour ce faire maintien une estimation de la moyenne des prédédents gradients, pour β_1 : $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$

Néanmoins, comme pour la variance cet estimateur de la moyenne est biaisé. On le redresse de la même manière :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

2.3 Couplage des deux grandeurs

En couplant les estimations de l'inertie et de la variance on obtient une règle de mise à jour qui adapte le *learning rate* et qui simule l'inertie de descente :

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad & \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ v_t &= \beta_2 b_{t-1} + (1 - \beta_2) g_t^2 \quad & \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned}$$

On peut alors analyser la règle de mise à jour. On a $\Delta_{t+1} = \theta_{t+1} - \theta_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$. Ainsi, la taille d'un pas de la descente de gradient est de l'ordre de $\frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$, c'est à dire la moyenne sur le carré de la variance non centrée qui est ici le ratio signal bruit. Et on voit donc, que lorsque ce ratio est faible, c'est à dire que le bruit est élevé – et donc l'incertitude sur la direction à suivre, on fait des pas plus petits ce qui correspond intuitivement à ce que l'on voudrait faire. En effet, cette incertitude est en général d'autant plus grande qu'on se rapproche d'un minimum (local ou non). Au contraire, lorsque ce ratio est élevé, on peut se permettre de faire de plus grands pas sans risques.

3 Algorithme ADAM

Algorithm 1 Adam

Require : η stepsize
Require : $\beta_1, \beta_2, \epsilon \in [0, 1]$
Require : $f(\theta)$ loss to minimize
  Initialize θ_0
  Initialize m_0, v_0 to zeros vectors
  $t \leftarrow 0$
  while θ_n has not converged **do**
    $g_t \leftarrow \nabla_{\theta_t} f(\theta_t)$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    $v_t \leftarrow \beta_2 b_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$
    $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$
    $\theta_{t+1} \leftarrow \theta_t - \dfrac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$
    $t \leftarrow t + 1$
  end while

4 Résultats empiriques

4.1 Comportement sur des problèmes jouets

On commence par tester l'algorithme présenté pour optimiser des fonctions jouets et on compare les résultats obtenus avec une descente de gradient usuelle.

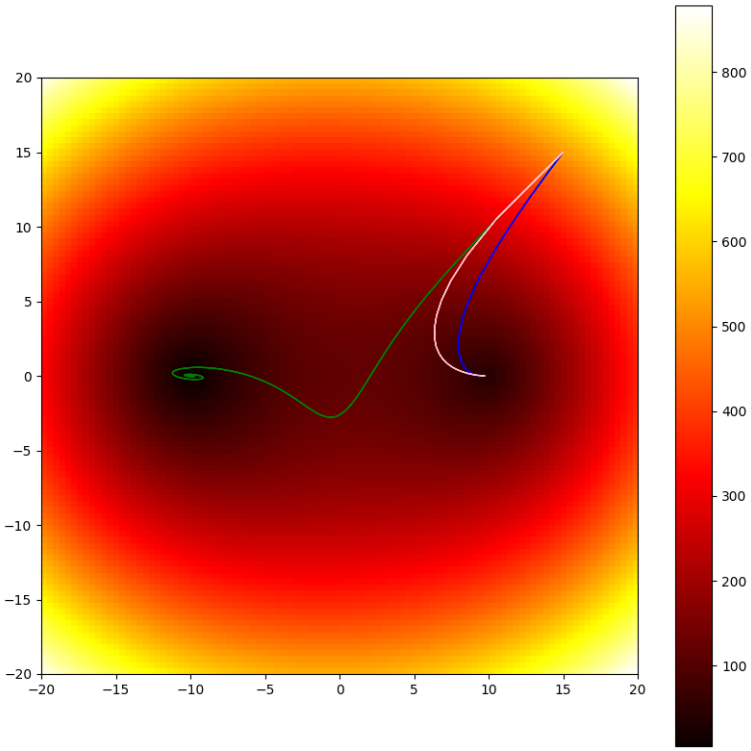

FIGURE 2 – Comparaison entre adam (vert), sgd (bleu) et RMSProp (rose)

4.2 Résultats sur MNIST

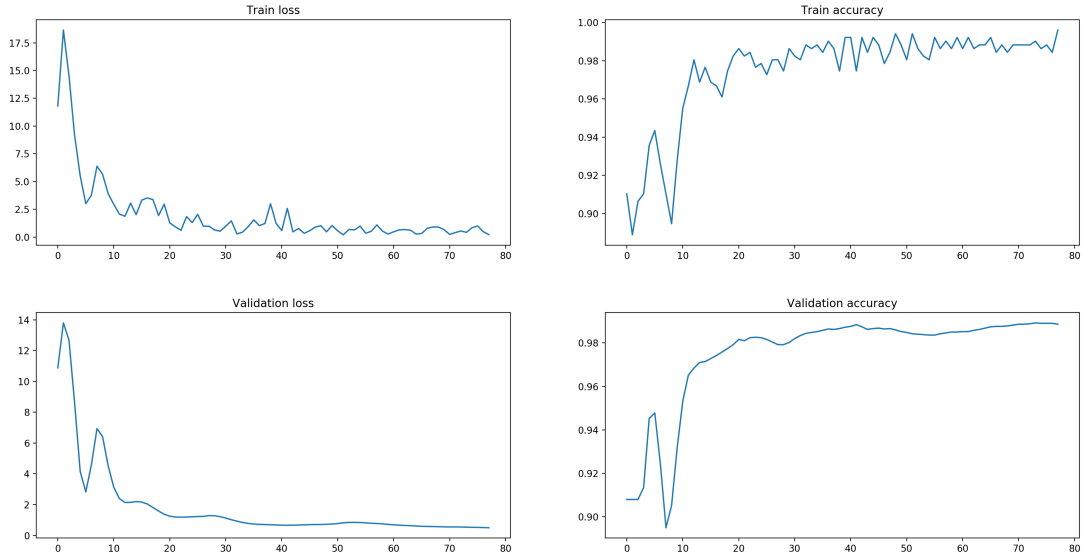

FIGURE 3 – Erreur et précision sur MNIST pour le jeu d'entraînement et le jeu de validation

4.3 Comparaison avec d'autres algorithmes