



**ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTEMES**

**MASTER RECHERCHE SCIENCES DE DONNÉES ET BIG DATA**

**MODULE : TECHNIQUES D'OPTIMISATION**

# RAPPORT : PROJET DE FIN DE MODULE

Encadré par : Pr. A. ELAFIA

Réalisé par : Rashid Haffadi

**ANNÉE UNIVERSITAIRE 2018/2019**

# Introduction et motivation

L'optimisation par les méthodes du gradient est d'une importance pratique fondamentale dans de nombreux domaines de la science, de la technologie et de l'ingénierie.

De nombreux problèmes dans ces domaines peuvent être exprimés comme l'optimisation d'une fonction objective qui nécessite une maximisation ou une minimisation par rapport à ses paramètres.

Si la fonction est différentiable, la méthode du gradient est une optimisation relativement efficace, car le calcul des dérivées partielles du premier ordre ont la même complexité de calcule que l'évaluation de la fonction. Souvent, les fonctions objectives sont stochastiques.

Par exemple, de nombreuses fonctions objectives sont composées d'une somme de sous-fonctions évaluées à différents niveaux dans ce cas l'optimisation peut être rendue plus efficace en prenant des gradients des sous-fonctions individuelles, à savoir descente de gradient stochastique (SGD).

SGD a fait ses preuves comme une méthode d'optimisation efficace et centrale qui a joué un rôle central dans le succès de beaucoup d'apprentissage automatique.

Notre présentation ici va ce limité aux méthodes du premier ordre.

# Sommaire

# I-Problème

## 1-Gradient stochastique

Soit P le problème de minimisation suivant : (1)

$$\min_x f(x)$$

$$\text{s. à. } x \in \mathrm{R}^d, \qquad d \in N$$

Où $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ , $f_i$ i = 1, 2, ... n sont des fonctions différentiables.

Puisque $f_i$ est différentiable alors on peut noter $\nabla f_i(x)$ le gradient de la fonction f dans le timestep i.

On note $\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$ le gradient de la fonction objective f. (2)

Dans la méthode du gradient de descente nous avons utilisé le gradient de la fonction objective pour résoudre le problème (P), la méthode du gradient converge linéairement avec $\frac{R(T)}{T} = O(\frac{1}{T})$.

Quand $f_i$ est une fonction de perte basée sur l'instance de données d'apprentissage indexée par i. Il est important de souligner que le coût de calcul par itération en descente de gradient est linéaire avec la taille de l'ensemble de données d'apprentissage n $(O(n))$. Par conséquent, lorsque n est énorme, le coût de calcul par descente de la descente de gradient est très élevé.

De ce fait, la descente de gradient stochastique offre une solution plus légère. À chaque itération, plutôt que de calculer le gradient $\nabla f(x)$, la descente de gradient stochastique échantillonne de manière aléatoire (i) uniformément et calcule $\nabla f_i(x)$. En effet la descente de gradient stochastique utilise $\nabla f_i(x)$ comme estimateur sans biais de f (x) puisque :

$$\mathrm{E}\left(\nabla f_i(x)\right) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x) \qquad (3)$$

Dans la pratique c'est préférable d'utiliser un échantillon de taille N, à la place d'utiliser 1 seule,

$$\mathrm{E}\left(\nabla f_N(x)\right) = \frac{N}{n}\sum_{i=1}^{\frac{n}{N}} \nabla f_N(x) = \nabla f(x)$$

$$= \frac{N}{n}\sum_{i=1}^{\frac{n}{N}}\sum \nabla f_i(x)$$

$$= \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x) \qquad (4)$$

On dit que $\nabla f_N(x)$ est un estimateur sans biais de $\nabla f(x)$.

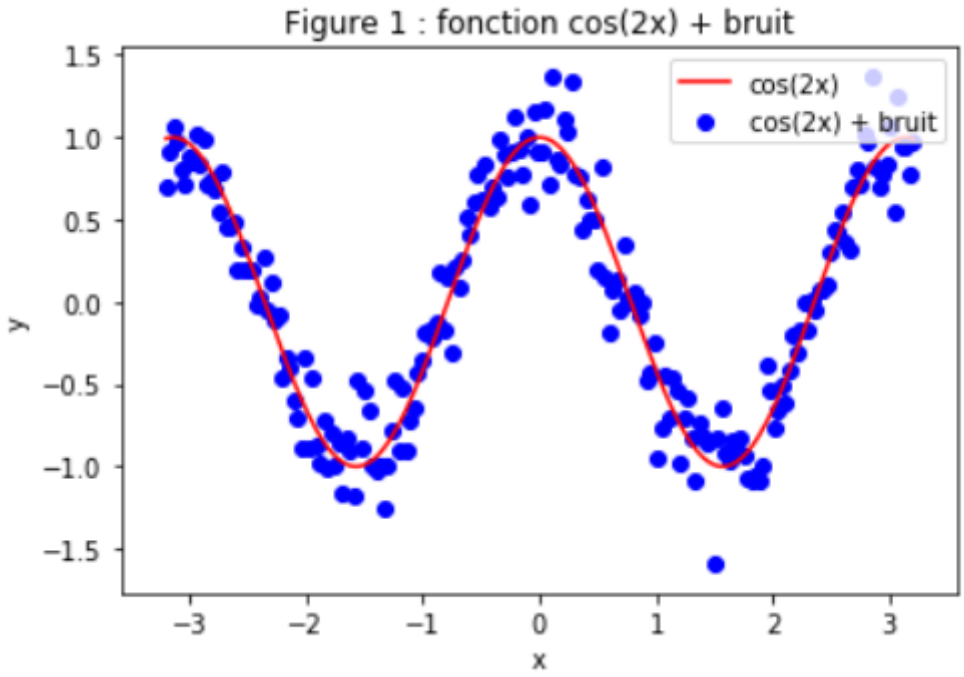
Plus N est grand plus l'estimation est plus précis.

Dans la suite on va introduire trois algorithmes qui utilisent le gradient stochastique pour la résolution de (P).

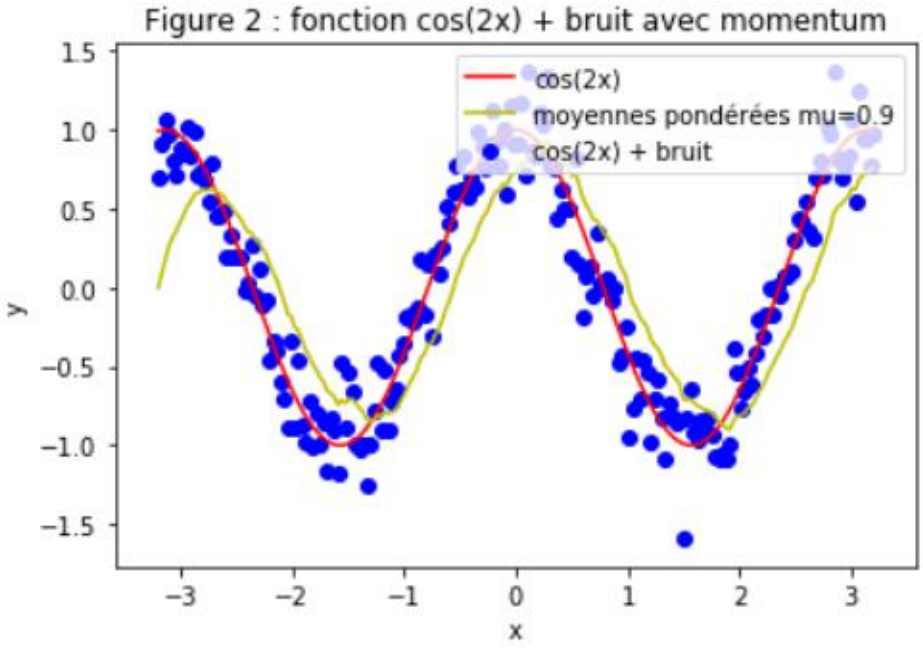
## 2- Les moyennes mobiles (Moving averages)

Supposons que nous ayons une séquence S de nombres qui soit bruyante. Pour cet exemple, on va tracer la fonction cosinus et ajouté du bruit. Cela ressemble à ceci :

$S = \{x, y, b \in R \,/\, y = \cos(x) + b\}$ telle que b est un bruit aléatoire.




Ce que nous voulons faire avec ces données, c'est, au lieu de les utiliser, nous voulons une sorte de «moyenne pondérées» qui «nuirait» les données et les rapprocherait de la fonction d'origine. Les moyennes mobiles de manière exponentielle peuvent nous donner un résultat qui ressemble à ceci :




Comme vous pouvez constater, c'est un très bon résultat. Au lieu d'avoir des données avec beaucoup de bruit, nous avons une ligne beaucoup plus fluide, ce

qui est plus proche de la fonction d'origine que les données dont nous disposions. Les moyennes mobiles de manière exponentielle définissent une nouvelle séquence V avec l'équation suivante :

$$\begin{aligned} V_t &= \mu V_{t-1} + (1-\mu)S_t \ , \mu \in [0,1] \\ &= \mu(\mu V_{t-2} + (1-\mu)S_{t-1}) + (1-\mu)S_t \\ &= \mu^{t-1}V_0 + (1-\mu)\sum_{i=1}^{t} \mu^i S_{t-i} \end{aligned}$$

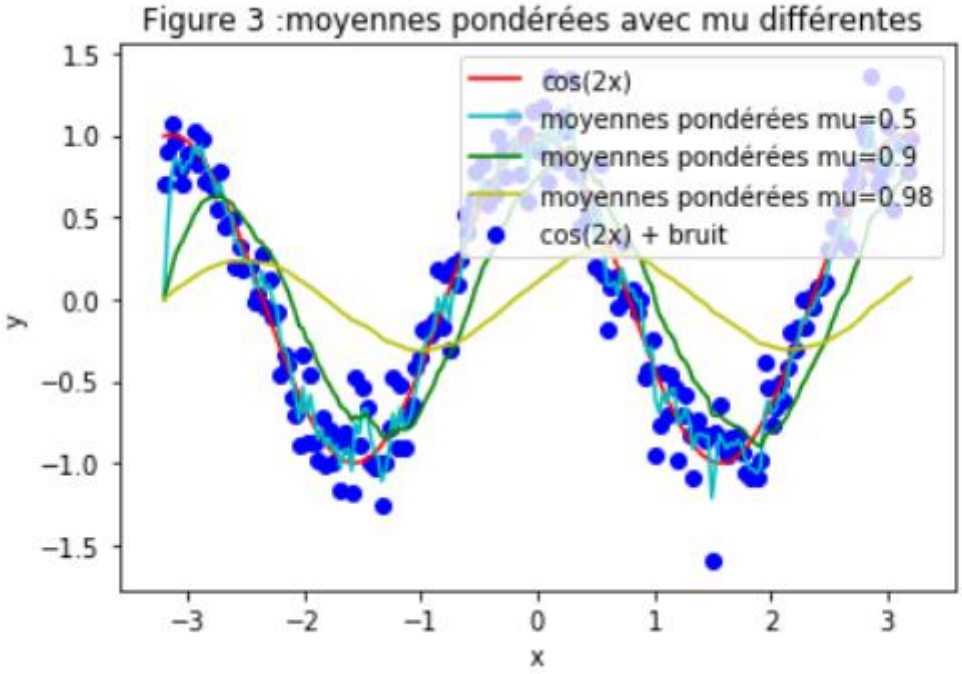
Si $V_0 = 0$ alors :

$$V_t = (1-\mu)\sum_{i=1}^{t} \mu^i S_{t-i} \qquad (5)$$

Donc quand t devient important $\mu^i \to 0$ $car$ $\mu \in [0,1]$.

Cette séquence V est celle représentée en jaune ci-dessus. Le μ est un autre hyper-paramètre qui prend des valeurs entre 0 et 1. On a utilisé μ= 0.9 ci-dessus (Figure 2). Intuitivement, vous pouvez penser μ comme suit, nous faisons une moyenne approximative sur les $\frac{1}{1-\mu}$ derniers points de la séquence. Voyons comment le choix de μ affecte notre nouvelle séquence V.

Figure 3 :moyennes pondérées avec mu différentes



Comme vous pouvez le constater, avec un nombre réduit de μ, la nouvelle séquence fluctue beaucoup, car nous calculons la moyenne sur un plus petit nombre d'exemples donc «plus proches» des données bruitées. Avec des valeurs de μ plus grandes, comme μ=0,98, nous obtenons une courbe beaucoup plus étouffée, mais elle est légèrement décalée vers la droite, car nous faisons la moyenne sur un plus grand nombre d'exemples (environ 50 pour μ = 0,98). μ = 0,9 fournit un bon équilibre entre ces deux extrêmes.

C'est pour cela qu'on va prendre μ = 0,9 pour tous nos tests dans la section des tests.

### 3- Les moyennes mobiles à correction de biais

La dernière chose à noter est que les deux premières itérations fourniront une très mauvaise moyenne car nous n'avons pas encore suffisamment de valeurs pour faire la moyenne. Au lieu d'utiliser V, la solution consiste à utiliser ce que l'on appelle la version de V à correction de biais.

Calculons $E(V_t)$ En utilisant le résultat de (5).

On aura :

$$E(V_t) = E((1-\mu)\sum_{i=1}^{t}\mu^{t-i}S_i)$$

$$= (1-\mu)E(\sum_{i=1}^{t}\mu^{t-i}S_i)$$

$$= E(S_i)(1-\mu)\sum_{i=0}^{t}\mu^{t-i} + \varepsilon$$

$$= E(S_i)(1-\mu^t) + \varepsilon \qquad (6)$$

Nous devons maintenant corriger l'estimateur afin que la valeur attendue soit celle que nous voulons. Cette étape est généralement appelée correction de biais. La formule finale pour notre estimateur sera la suivante :

$$\widehat{V_t} = \frac{V_t}{1 - \mu^t} \qquad (7)$$

Avec de grandes valeurs de t, la puissance de t sera très proche de zéro, ne modifiant donc pas nos valeurs de V du tout. Mais pour de petites valeurs de t, cette équation produira des résultats un peu meilleurs.

Ce résultat est très important pour comprendre le fonctionnement des algorithmes qu'on va présenter après.

## II-Adam

### 1-pseudo code

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
**while** $\theta_t$ not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
**end while**
**return** $\theta_t$ (Resulting parameters)

### 2-Algorithme

Soit la fonction objective $J(x) = E(f(x)) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$

Nous sommes intéressés à minimiser la fonction J respectivement à x et $x \in R^d$ .

On note $f_i$ i = 1, 2, ... n les réalisations de la fonction stochastique $f(x)$ et $g_t = \nabla f_i(x)$.

L'algorithme met à jour les moyennes mobiles du gradient $(m_t)$ et du gradient carré $(\widehat{V}_t)$ où les paramètres β1, β2 ∈ [0, 1) contrôlent les vitesses de décroissance exponentielles de ces déplacements moyennes. Les moyennes mobiles elles-mêmes sont des estimations du premier ordre du moment (la moyenne) et du 2ème ordre du moment brut (la variance non centrée) du gradient. Cependant, ces moyennes mobiles sont initialisé avec 0 $(\in R^d)$, conduisant à des estimations de moment biaisées vers zéro, en particulier pendant les pas de temps initiaux, et en particulier lorsque les taux de décroissance sont faibles (c'est-à-dire que les β sont proches de 1).

Alors Adam peut être considéré comme une combinaison de RMSprop et de GSD avec moment. Il utilise les gradients au carré pour mettre à l'échelle le taux d'apprentissage comme RMSprop et la moyenne mobile du gradient au lieu du gradient lui-même comme GSD avec moment.

Parfois, la valeur de $(\widehat{V_t})$ pourrait être très proche de 0. La valeur absolue de nos poids pourrait devenir très grande, pour éviter cela nous incluons un paramètre epsilon $(\varepsilon)$ dans le dénominateur, qui prend valeur très faible.

## 3-Analyse de convergence

### Lemme 3.1 : inégalité du gradient.

Soit $D_f$ $de$ $R^d$ un ensemble convexe, $f \epsilon$ $\mathcal{C}^1$ $R^d \rightarrow R$

F est convexe sur $D_f$ ssi :

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \qquad \forall x, y \epsilon D_f$$

### Conjecture 3.2 : malheureusement ce n'est pas encore démontré.

Soit $g_t = \nabla f_t(x)$, $g_{t,i}$ le ième élément de $g_t$

$$g_{1:t,i} = (g_{1,i}, g_{2,i}, g_{3,i}, \dots, g_{t,i})^T \epsilon R^t$$

$$\gamma = \frac{{\beta_1}^2}{\sqrt{\beta_2}} \quad \beta_1, \beta_2 \epsilon (0,1) \quad \gamma < 1.$$

Et Soit $\|g_t\|_2 \leq G$ et $\|g_t\|_\infty \leq G_\infty$ alors :

$$\sum_{t=1}^{n} \frac{\widehat{m_{t,\iota}}}{\sqrt{t\widehat{\vartheta_{t,\iota}}}} \leq \frac{2}{1-\gamma} * \frac{1}{\sqrt{1-\beta_2}} \left\| g_{1:n,i} \right\|_2$$

### Définition 3.3 : Somme des erreurs

Soit $x^* = \min_{x \epsilon X} \sum_{t=1}^{n} f_t(x)$

$$\mathrm{R(n)} = \sum_{t=1}^{n} (f_t(x_t) - f_t(x^*))$$

: Démonstrations

Théorème 3.4:

Soit $f_t$ des fonctions différentiables et convexes pout tout t=1, 2, …, n

$$g_t = \nabla f_t(x)$$

Soit $\|g_t\|_2 \leq G$ , $\|g_t\|_\infty \leq G_\infty \ \ \forall t\epsilon\{1,2,\dots,n\}$

Supposons que :

$$\|x_i - x_j\|_2 \leq D \ et \ \|x_i - x_j\|_\infty \leq D_\infty \ \ \forall \ i,j \ \epsilon\{1,2,\dots,n\}$$

$$\gamma = \frac{{\beta_1}^2}{\sqrt{\beta_2}} \quad \beta_1, \beta_2 \epsilon (0,1) \quad \gamma < 1 \quad , \alpha_t = \frac{\alpha}{\sqrt{t}}$$

$$\beta_{1,t} = \beta_{1,t}\tau^{t-1}, \tau\epsilon(0,1)$$

$$R(n) \leq \frac{D_\infty}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{n\widehat{\vartheta_{n,\iota}}} + \frac{d{D_\infty}^2 G_\infty}{2\alpha(1-\beta_1)(1-\tau)^2} + \frac{\alpha(1+\beta_1)}{(1-\beta_1)(1-\tau)\sqrt{1-\beta_2}} \sum_{i=1}^{d} \|g_{1:n,i}\|_2$$

Corollaire 3.5 :

Soit $f_t$ des fonctions différentiables et convexes pout tout t=1, 2, …, n

$$g_t = \nabla f_t(x)$$

Soit $\|g_t\|_2 \leq G$ , $\|g_t\|_\infty \leq G_\infty \ \ \forall t\epsilon\{1,2,\dots,n\}$

Supposons que :

$$\|x_i - x_j\|_2 \leq D \ et \ \|x_i - x_j\|_\infty \leq D_\infty \ \ \forall \ i,j \ \epsilon\{1,2,\dots,n\}$$

Alors l'estimation de convergence suivant est vérifiée :

$$\frac{R(n)}{n} = O(\frac{1}{\sqrt{n}})$$

## III-AdaMax

### 1-pseudo code

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$u_0 \leftarrow 0$ (Initialize the exponentially weighted infinity norm)
$t \leftarrow 0$ (Initialize timestep)
**while** $\theta_t$ not converged **do**
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ (Update biased first moment estimate)
$u_t \leftarrow \max(\beta_2 \cdot u_{t-1}, |g_t|)$ (Update the exponentially weighted infinity norm)
$\theta_t \leftarrow \theta_{t-1} - (\alpha/(1-\beta_1^t)) \cdot m_t/u_t$ (Update parameters)
**end while**
**return** $\theta_t$ (Resulting parameters)

### 2-algorithme

L'idée avec Adamax est de considérer la valeur $v_t$ comme la norme L2 des gradients (ou bien une estimation de $\nabla f(x)^2$). On peut généraliser le résultat pour les normes d'ordre p, mais cela devient assez instable pour les grandes valeurs de p. Mais si nous utilisons le cas particulier de la norme L∞ ($p \to \infty$), il en résulte un algorithme étonnamment stable et performant.

$$\vartheta_t = {\beta_2}^p v_{t-1} + (1 - {\beta_2}^p)|g_t|^p$$

$$= (1 - {\beta_2}^p) \sum_{i=1}^{t} {\beta_2}^{p(t-i)} |g_i|^p$$

Quand $p \to \infty$ on aura :

$$u_t = \lim_{p\to\infty} (\vartheta_t)^{\frac{1}{p}}$$

$$= \lim_{p\to\infty} ({\beta_2}^p v_{t-1} + (1 - {\beta_2}^p)|g_t|^p)^{1/p}$$

$$= \lim_{p\to\infty}(1-\ {\beta_2}^p)^{\frac{1}{p}}.(\sum_{i=1}^{t}{\beta_2}^{p(t-i)}|g_i|^p)^{\frac{1}{p}}$$

$$= \lim_{p\to\infty}(\sum_{i=1}^{t}({\beta_2}^{(t-i)}|g_i|)^p)^{\frac{1}{p}}$$

$$= \max({\beta_2}^{t-1}|g_i|, {\beta_2}^{t-2}|g_2|, \dots, \beta_2|g_{t-1}|, |g_t|)$$

Ce qui correspond à la formule récursive suivante :

$$u_t = \max(\beta_2 u_{t-1}, |g_t|)$$

## 3- Avantages de AdaMax :

### a- pas de mise à jour des paramètres :

La taille de pas réelle $(d_t)$ prise par AdaMax à chaque itération est inférieur à l'hyper-paramètre $\alpha$ $(|d_t| \leq 1)$.

### b- Taille de mise à jour invariant:

La taille de pas de la règle de mise à jour d'Adam est invariante par rapport à la magnitude du gradient, ce qui aide beaucoup lorsque on fait le parcoure des zones avec des gradients très petits.

### c- Les mêmes avantages de Adam :

Adam a été conçu pour combiner les avantages d'Adagrad, qui fonctionne bien avec des gradients clairsemés (sparse gradients), et de RMSprop, qui fonctionne bien dans l'optimisation en ligne. Avoir ces deux éléments nous permet d'utiliser Adam pour une gamme plus large de tâches. Adam peut également être considéré comme la combinaison de RMSprop et GSD avec moment.

### d- Le même taux de convergence qu'Adam

L'algorithme AdaMax converge avec le même taux de convergence qu'Adam $O(1/\sqrt{n})$.

## 4- Problèmes de AdaMax :

Plusieurs articles ont mentionné des problèmes de convergence dans adam et adamax.

Par exemple dans l'article **[2]** ont démontré qui a des fautes dans la démonstration de convergence dans l'article mère de Adamax et Adam **[1]** et ils ont donné une démonstration plus fiable.

Mais avec cela un autre article **[3]** récent publié en 2018 a démontré qu'il ne converge pas dans plusieurs cas par exemple Soit $f_t(x)$ une fonction définit sur $[-1,1]$ :

$$f_t(x) = \begin{cases} Cx \ si \ t \ \% \ 3 = 1 \\ -x \ sinon \end{cases}$$

Note :% = modulo

Quand C > 2 c'est évident que x=-1 donne le minimum de la somme des erreurs R(n), mais pour $\beta_1 = 0 \ et \ \beta_2 = \frac{1}{1+C^2}$ Adam diverge vers x=+1.

Un autre algorithme (AMSGrad) est proposé pour résoudre ce problème sans augmenter la complexité.

**Algorithm 2** AMSGRAD

**Input:** $x_1 \in \mathcal{F}$, step size $\{\alpha_t\}_{t=1}^{T}$, $\{\beta_{1t}\}_{t=1}^{T}$, $\beta_2$
Set $m_0 = 0$, $v_0 = 0$ and $\hat{v}_0 = 0$
**for** $t = 1$ **to** $T$ **do**
  $g_t = \nabla f_t(x_t)$
  $m_t = \beta_{1t} m_{t-1} + (1 - \beta_{1t}) g_t$
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$ and $\hat{V}_t = \mathrm{diag}(\hat{v}_t)$
  $x_{t+1} = \Pi_{\mathcal{F}, \sqrt{\hat{V}_t}}(x_t - \alpha_t m_t / \sqrt{\hat{v}_t})$
**end for**

## IV-Nesterov-accelerated Adaptive Moment Estimation (Nadam)

### 1-pseudo code

**Algorithm 2** Nesterov-accelerated Adaptive Moment Estimation (Nadam)

**Require:** $\alpha_0, \ldots, \alpha_T$; $\mu_0, \ldots, \mu_T$; $\nu$; $\epsilon$: Hyperparameters

$\mathbf{m}_0; \mathbf{n}_0 \leftarrow 0$ (first/second moment vectors)

**while** $\theta_t$ not converged **do**

$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$

$\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + (1-\mu_t)\mathbf{g}_t$

$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$

$\hat{\mathbf{m}} \leftarrow (\mu_{t+1}\mathbf{m}_t / (1 - \prod_{i=1}^{t+1} \mu_i)) + ((1-\mu_t)\mathbf{g}_t / (1 - \prod_{i=1}^{t} \mu_i))$

$\hat{\mathbf{n}} \leftarrow \nu \mathbf{n}_t / (1-\nu^t)$

$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{\mathbf{n}}_t + \epsilon}} \hat{\mathbf{m}}_t$

**end while**

**return** $\theta_t$

### 2-Algorithme

Nous pouvons considérer les gradients accélérés de Nesterov comme facteur de correction pour les méthodes des moments. Prenons le cas où la vélocité ajoutée aux paramètres vous donne une perte élevée immédiate non désirée, par exemple le cas du gradient très grand. Dans ce cas, les méthodes des moments peuvent être très lentes car le chemin d'optimisation emprunté présente de fortes oscillations. Dans le cas du gradient accéléré de Nesterov, vous pouvez l'afficher comme si vous fouilliez à travers les paramètres intermédiaires lorsque la vélocité ajoutée mènerait les paramètres. Si la mise à jour de la vitesse conduit à une mauvaise perte, les gradients la dirigeront vers $x_t$. Cela aide Nesterov Accelerated Gradient (NAG) à éviter les oscillations.

Nadam (Estimation du moment adaptatif accéléré par Nesterov) combine Adam et NAG.

Pour incorporer NAG dans Adam, nous devons modifier le terme du moment $(m_t)$ :

$$g_t = \nabla f_t(x_t)$$

$$m_t = \mu_t m_{t-1} + \alpha_t g_t$$

$$x_t = x_{t-1} - (\mu_{t+1} m_t + \alpha_t g_t)$$

On appliquant ce résultat sur Adam on aura :

$$x_t = x_{t-1} - \alpha_t \left(\frac{\mu_t m_{t-1}}{1 - \prod_{i=1}^{t} \mu_i} + \frac{(1-\mu_t) g_t}{1 - \prod_{i=1}^{t} \mu_i}\right)$$

$$x_t = x_{t-1} - \alpha_t \left(\frac{\mu_{t+1} m_t}{1 - \prod_{i=1}^{t+1} \mu_i} + \frac{(1-\mu_t) g_t}{1 - \prod_{i=1}^{t} \mu_i}\right)$$

Cela est faisable aussi avec AdaMax aussi.

## V- recherche linéaire non monotone (F-rule)

### 1-Définitions

**Notation 1 :**

On note $L = \{x \epsilon R^d \big| f(x) \leq f(x_0)\}$

Supposons f(x) est borné dans L et f (x) est continue sur un ensemble ouvert qui contient L.

**Définition 1 :** La fonction $\sigma : [0, +\infty] \rightarrow [0, +\infty]$ est une fonction de forçage (fonction F) si pour toute séquence $\{t_i\}\ dans\ [0, +\infty]$

$$\lim_{i\to\infty} \boldsymbol{\sigma(t_i) = 0} \Rightarrow \lim_{i\to\infty} \boldsymbol{t_i = 0}$$

**Définition 2 :** Soit $\boldsymbol{\rho = sup\{\ \|g(x) - g(y)\|\ x, y\ \epsilon L\} > 0}$

$\boldsymbol{\delta}: [\mathbf{0}, +\infty] \rightarrow [\mathbf{0}, +\infty]$une fonction définit par :

$$\boldsymbol{\delta(t)} = \begin{cases} \boldsymbol{inf\{\|x - y\| \| \|g(x) - g(y)\| \geq t\}\ si\ t \epsilon [0, \rho)} \\ \boldsymbol{\lim_{s\to\rho^-} \delta(s)\ \ si\ t \epsilon [\rho, +\infty]} \end{cases}$$

### 2-Algorithme

Entrées : $x_k$, M, $\gamma \epsilon (0,1), \sigma \epsilon (0,1), k\ (l'itération\ actuel)$,

Sortie : $\alpha_k$

Début

$\alpha_k$ := 1

si k=0 :

m(k) :=1

sinon :

m(k) := min(k+1, M)

$\varphi(\alpha) = f_k(x_k + \alpha d_k)$

$\lambda_k \epsilon (0,1]$ Prendre $\lambda_{kr} \geq \lambda_k\ r = 0, 1, \dots, m(k) - 1, \sum_{r=0}^{m(k)-1} \lambda_{kr} = 1$

Tant que$(\varphi(\alpha) > \max\left\{f_k(x_k),\ \sum_{r=0}^{m(k)-1} \lambda_{kr} f_k(x_{k-r})\right\} + \gamma\alpha {g_k}^T d_k)$ :

$\alpha_k := \sigma\alpha_k$

Fin tant que

Return $\alpha_k$

Fin.

## 3-Convergence globale

**Règle de mise à jour :**

$$f(x_k + \alpha_k d_k) \leqslant \max\left[f(x_k), \sum_{r=0}^{m(k)-1} \lambda_{kr} f(x_{k-r})\right] - \sigma(t_k),$$

$$x_{k+1} = x_k + \alpha_k d_k.$$

Lemme 3.1 :

Si :

$$f(x_k + \alpha_k d_k) \leqslant \max\left[f(x_k), \sum_{r=0}^{m(k)-1} \lambda_{kr} f(x_{k-r})\right] - \sigma(t_k),$$

Et : $x_{k+1} = x_k + \alpha_k d_k.$

Alors

$$f(x_k) \leqslant f(x_0) - \lambda \sum_{r=0}^{k-2} \sigma(t_r) - \sigma(t_{k-1}) \leqslant f(x_0) - \lambda \sum_{r=0}^{k-1} \sigma(t_r).$$

Théorème 3.2 :

Sous notation 1 si :

$$\left|\frac{-g_k^{\mathrm{T}} d_k}{\|d_k\|}\right| \geqslant \sigma(\|g_k\|), \quad k = 1, 2, \ldots$$

Alors $\{x_k\} \subseteq \mathscr{L}$ et $\lim_{k\to\infty} \|g_k\| = 0$

Notation 3.3 :

f(x) est deux fois différentiable alors : il existe c3, c4 telle que :

$$c_3\|z\|^2 \leqslant z^{\mathrm{T}} G(x) z \leqslant c_4\|z\|^2,$$

$$G(x) = \nabla^2 f(x).$$

Lemme 3.4 :

Si notation 2 est vérifié alors il existe un réel positive telle que :

$$\frac{\|y_k\|^2}{y_k^{\mathrm{T}} s_k} \leqslant c_5, \quad k = 1, 2, \ldots.$$

Théorème 3.5 :

Soit f(x) une fonction qui satisfait les notations 1 et 2,
$\forall x_0 \epsilon R^n, B_0 \epsilon R^{n\times n}$ *une matrice symetrique définit positive, soit* $\{x_k\}$ *une* suite infinie généré par Quasi-Newton avec la règle de mise à jour au-dessus alors :

$$\lim_{k\to\infty} \|g_k\| = 0.$$

## VI- Tests Numériques

### 1 - Regression Lineaire

Soit $D = \{(x_i, y_i) \mid x_i \epsilon R^d, y_i \epsilon R\}$ un ensemble de données numérique.

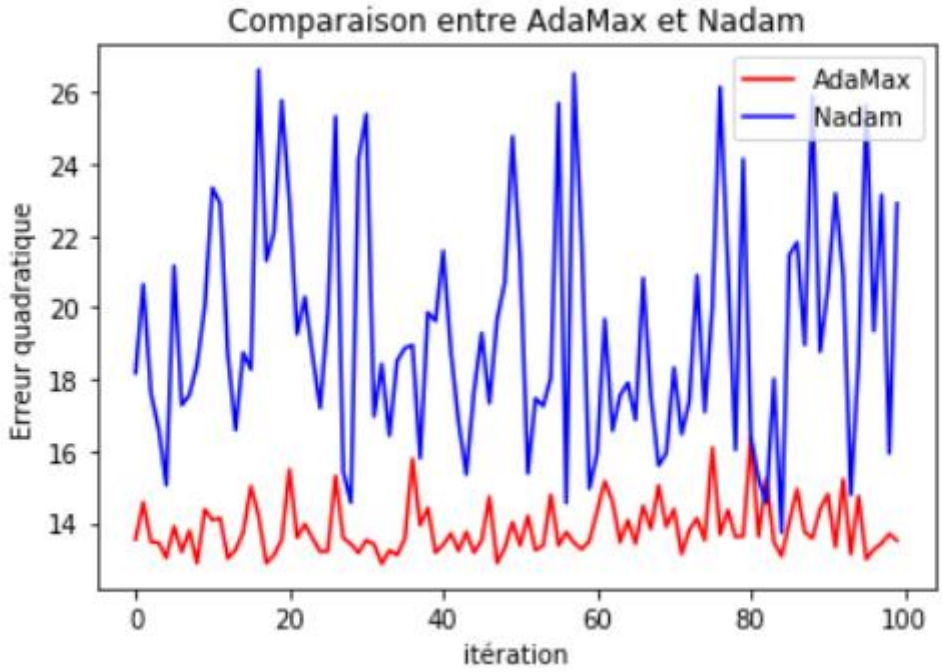
Soit la fonction objective $J(\omega) = \frac{1}{2m}\sum_{i=1}^{i=m}(\omega^T x_i - y_i)^2 \quad , \omega \epsilon R^d$.

La fonction J est l'erreur quadratique de D réspectivement $\omega$.

Soit $\rho_i(\omega) = \frac{1}{2}(\omega^T x_i - y_i)^2$

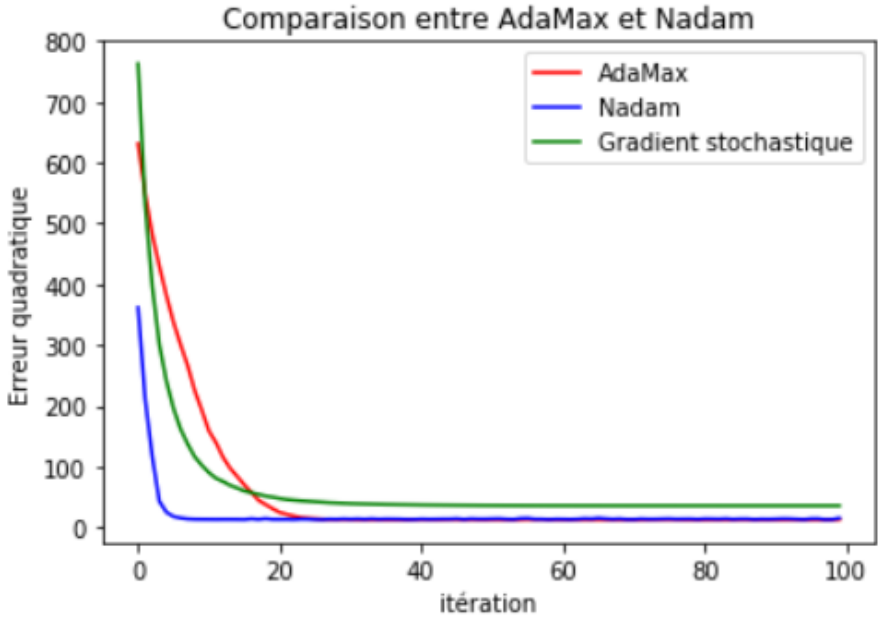
$$g_i = \frac{\partial \rho_i(\omega)}{\partial w} = (\omega^T x_i - y_i)x_i$$

Le test est fait avec AdaMax et Nadam sur les paramétres : m = 10000 (taille des données), $\alpha$ =0.05 (pas d'apprentissage), β1=0.9, β2=0.999, d=10 (la dimension de données).




alors pour $\alpha$ =0.05 AdaMax et Nadam ne sont pas stable méme que AdaMax donne des résultat approché.

On vas tester avec un pas plus grand α=0.01 pour AdaMax et Nadam et α=2*1e-5 pour l'algorithme du gadient stochastique.




## 2 - ($f_{\alpha,\beta}(x) = \alpha_t {x_1}^2 + \beta_t {x_2}^2$) Convexe

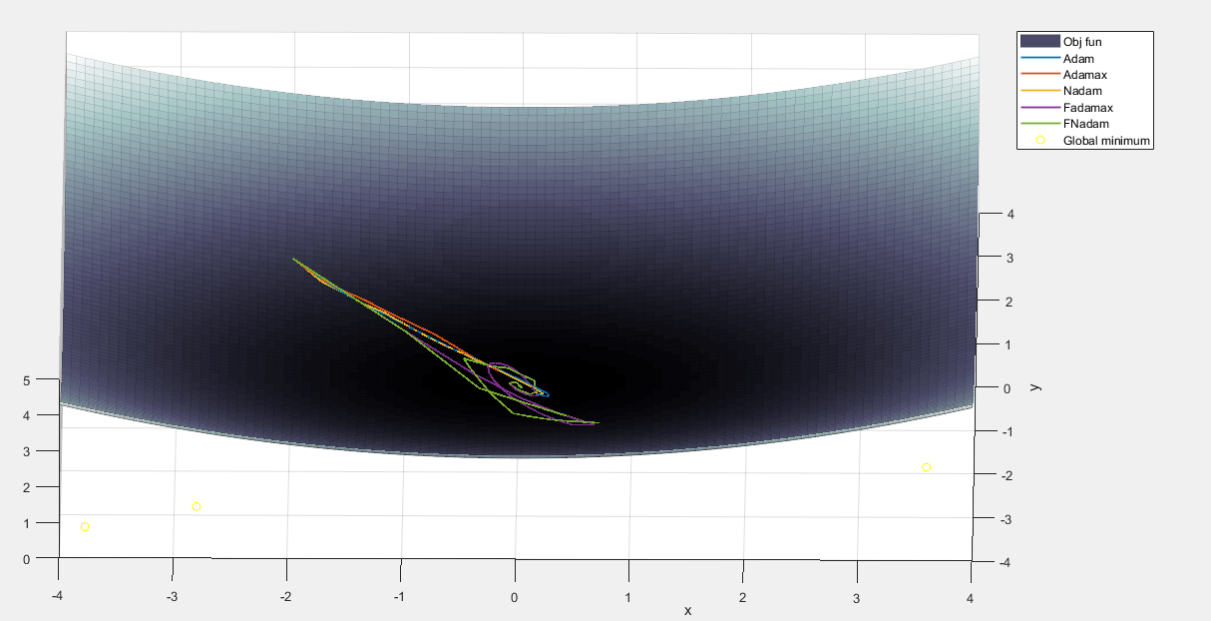
Pout chaque itération t on choisie aléatoirement $\alpha_t, \beta_t$ (méthode PEAR) telle que :

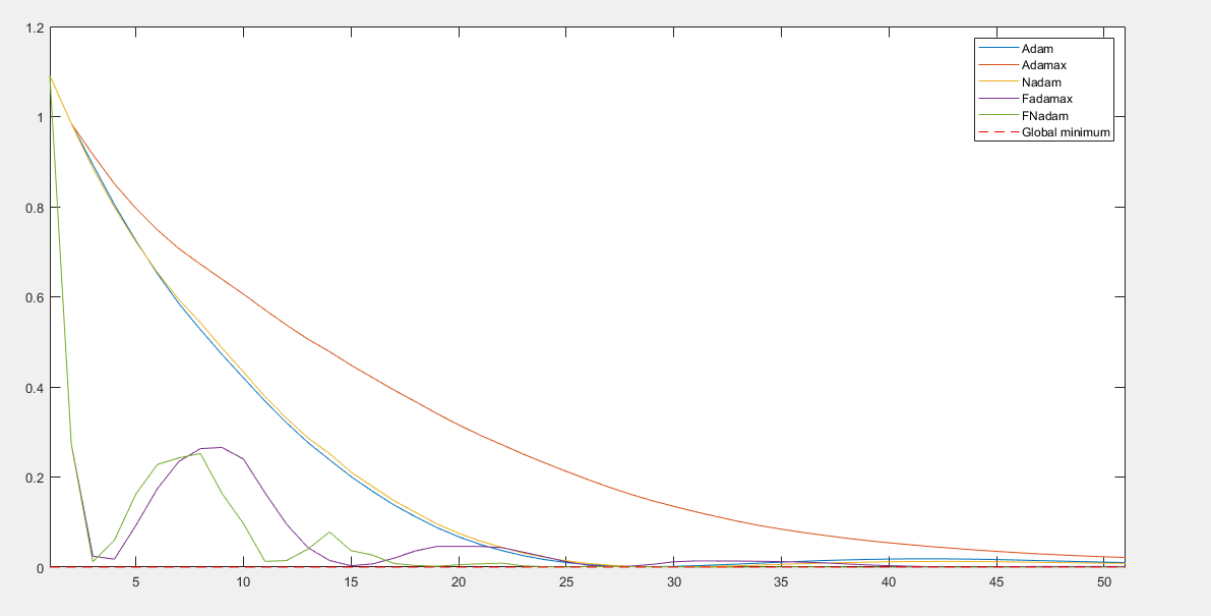
$$\alpha_t, \beta_t \epsilon \{\left(\frac{1}{2}, \frac{1}{2}\right), (1,1), (2,2), (3,3), (4,4), (2,1), (1,2), (3,1), (1,3), (4,1), (1,4)\}$$

$$J(x) = \frac{1}{T} \sum_{t=1}^{T} f_{\alpha_t,\beta_t}(x)$$

On veut minimiser la fonction J s.à x :

Pour $\alpha = 0.1, \beta_1 = 0.9, \beta_2 = 0.999, x_0 = (-2, 2), N = 50, \gamma = 0.5, \varepsilon = 10^{-5}$, $M = 10$.



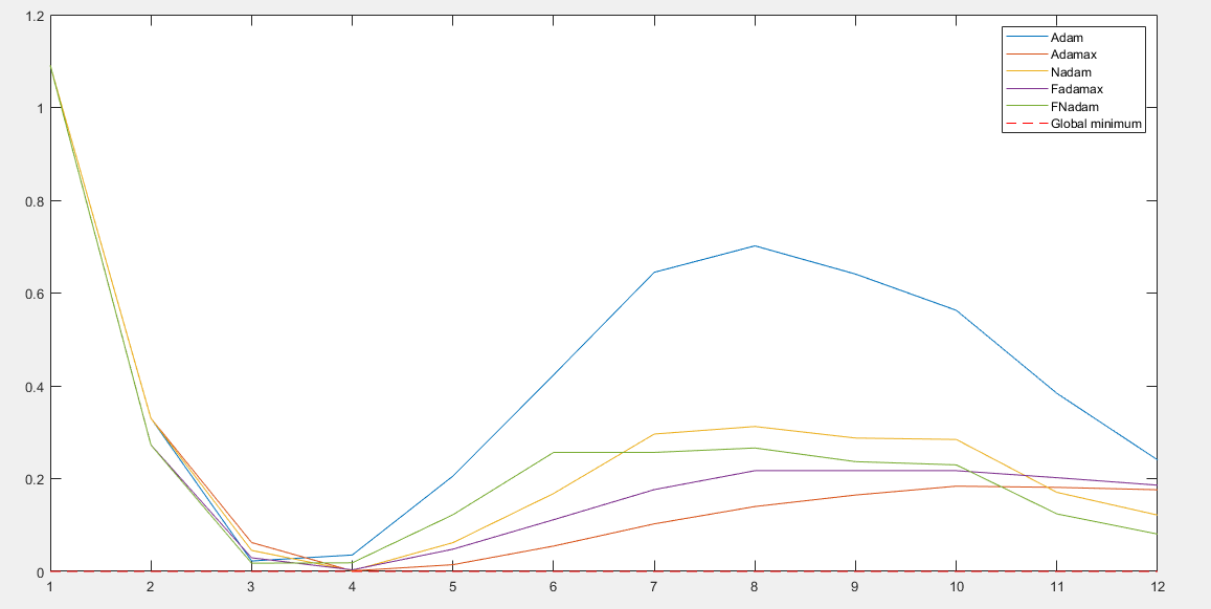

Fadamax : Adamax avec le pas approché.

FNadam : Nadam avec le pas approché.

Une premiére remarque qu'on peut faire sur les deux figure est que le pas approché a beucoup réduit le taux de convergence, pour Nadam par exemple il converge en 28 itérations mais avec le pas approché il converge en 3 itérations seulement .

Une deuxiéme remarque est que après l'algorithme converge une premiére fois, il diverge mais il converge encore une fois, c'est bien le fait du moment.

Avec $\alpha$ =0.9 pour Adam, AdaMax et Nadam on aura:




Tous les algorithms converge en 4 itérations, mais avec une supériorité de FNadam et Nadam en terme du précision.

## 3 - Beale Function

Soit la fonction objective $\mathbf{J(x,y)} = ((\frac{3}{2} - \mathbf{x} + \mathbf{xy})^2 + (\frac{9}{4} - \mathbf{x} + \mathbf{xy^2})^2 + (\frac{21}{8} - \mathbf{x} + \mathbf{xy^3})^2)/\mathbf{3}, \quad \forall \boldsymbol{x, y \epsilon R}.$

J est différentiable, multimodale et non-convex.

On vas étudier la fonction sur [-4.5 4.5]×[-4.5 4.5] :

$$= \nabla J(x,y)$$

$$= 2\begin{pmatrix} \left(\frac{3}{2} - \mathbf{x} + \mathbf{xy}\right)(y-1) + \left(\frac{9}{4} - \mathbf{x} + \mathbf{xy^2}\right)(\boldsymbol{y^2} - \mathbf{1}) + (\frac{21}{8} - \mathbf{x} + \mathbf{xy^3})(\boldsymbol{y^3} - \mathbf{1}) \\ x\left(\frac{3}{2} - \mathbf{x} + \mathbf{xy}\right) + 2xy\left(\frac{9}{4} - \mathbf{x} + \mathbf{xy^2}\right) + \mathbf{3}\boldsymbol{xy^2}(\frac{21}{8} - \mathbf{x} + \mathbf{xy^3}) \end{pmatrix}$$

Prenons $J_1(x,y) = = (\frac{3}{2} - \mathbf{x} + \mathbf{xy})^2$ , $J_2(x,y) = (\frac{9}{4} - \mathbf{x} + \mathbf{xy^2})^2$ , $J_3(x,y) = (\frac{21}{8} - \mathbf{x} + \mathbf{xy^3})^2$.

Alors :

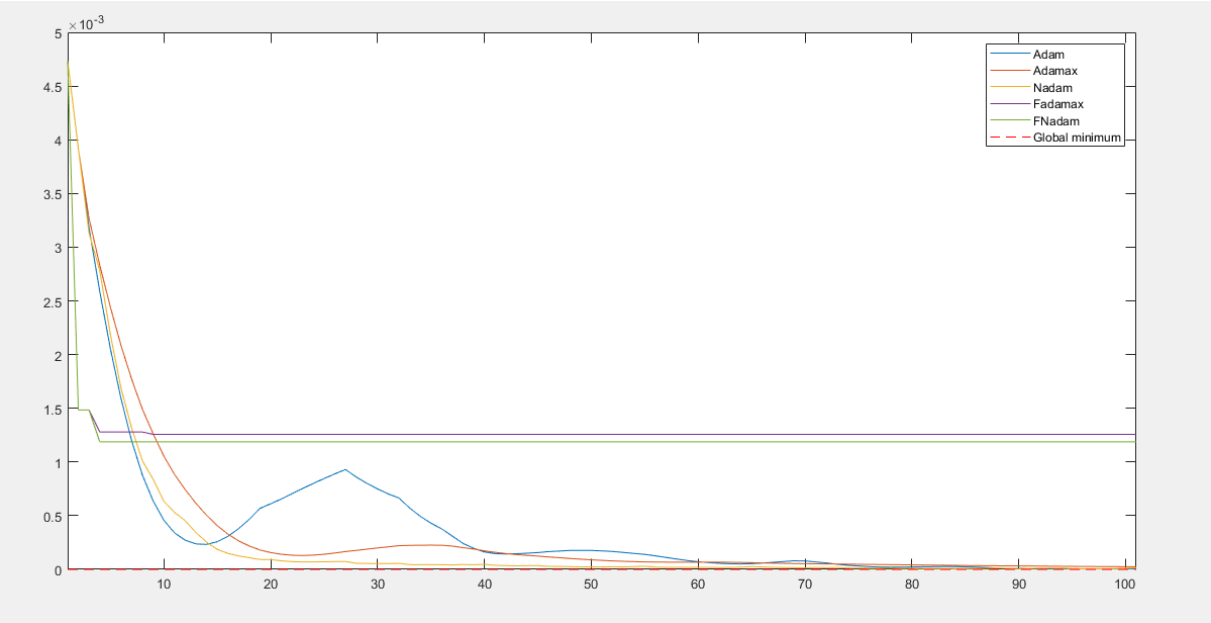
$$g_1 = \nabla J_1(x,y) = 2\begin{pmatrix} \left(\frac{3}{2} - \mathbf{x} + \mathbf{xy}\right)(y-1) \\ x\left(\frac{3}{2} - \mathbf{x} + \mathbf{xy}\right) \end{pmatrix},$$

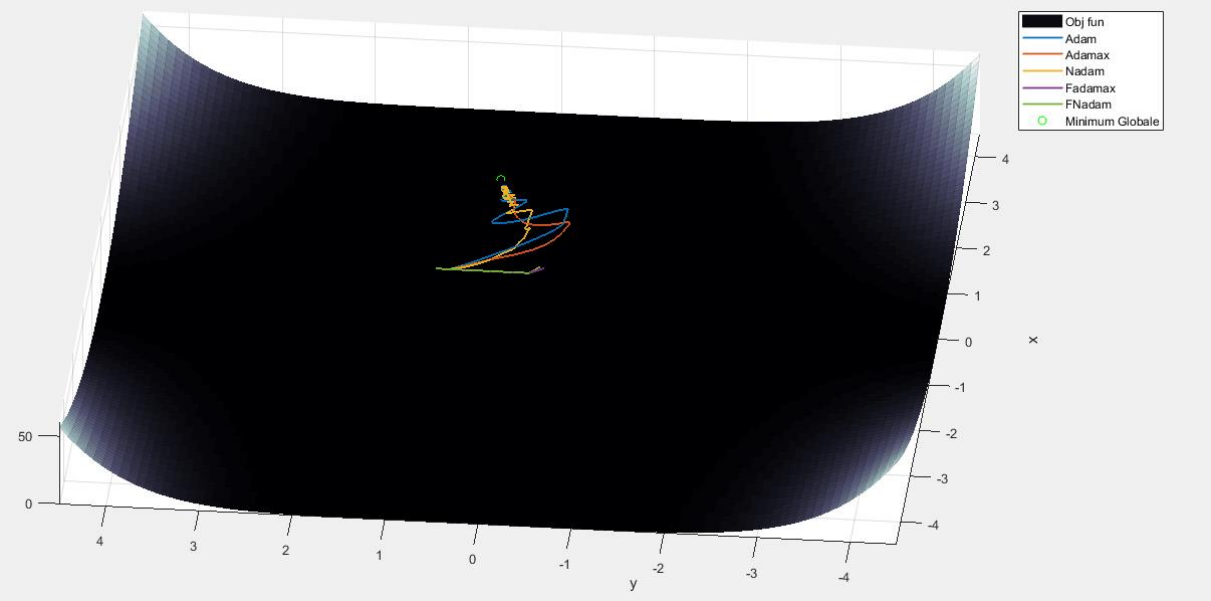
$$g_2 = \nabla J_2(x,y) = 2\begin{pmatrix} \left(\frac{9}{4} - x + xy^2\right)(y^2 - 1) \\ 2xy\left(\frac{9}{4} - x + xy^2\right) \end{pmatrix},$$

$$g_3 = \nabla J_3(x,y) = 2\begin{pmatrix} \left(\frac{21}{8} - x + xy^3\right)(y^3 - 1) \\ 3xy^2\left(\frac{21}{8} - x + xy^3\right) \end{pmatrix}$$

On vas prendre d'une maniére aleatoire avec remise pour chaque itération t : $g_t = g_1 \; ou \; g_2 ou \; g_3$.

Les tests sont faites avec les paramétres : $\alpha$ = 0.1, x0 = [1 1], N=100, β1=0.9, β2=0.999.



Ce qu'on peut tirer des deux figure ci-dessus est que les algorithmes Adam, Nadam et Adamax converge vers un minimum globale ou bien au moins vers un minimum locale.

Les deux algorithms avec le pas approché sont stocker dans un minimum locale.

## VII-Conclusion

Dans ce rapport on a pu à étudier quatre algorithmes d'optimisation qui sont les plus utiliser dans les problèmes de machine/deep Learning avec des données de grands dimensions.

Pour récapituler, on a dit que Adam et AdaMax combine entre les avantages de deux méthodes populaires, AdaGrad pour les gradients 'sparse' et RMSProp pour les fonctions objectives non stationnaires.

Nadam est une méthode plus robuste, car c'est une amélioration de Adam utilisant le moment de Nesterov.

Comme ces méthodes sont très bonnes pour la résolution des problèmes convexes, ces algorithmes sont aussi bons pour la résolution de beaucoup de problèmes non convexe qu'on peut rencontrer en Machine/Deep Learning.

## Annex : Démonstrations

### Théorème 3.4

*Proof.* With lemma 4.1 we can write for a convex differentiable function $e(w)$:

$$\begin{aligned} e_t\left(\vec{w}^*\right) &\geq e_t\left(\vec{w}_t\right) + g_t^T\left(\vec{w}^* - \vec{w}_t\right) \\ \Leftrightarrow e_t(\vec{w}_t) - e_t(\vec{w}^*) &\leq g_t^T\left(\vec{w}_t - \vec{w}^*\right) \end{aligned}$$

With the update rule from the ADAM-Optimizer:

$$\begin{aligned} \vec{w}_{t+1} &= \vec{w}_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \\ &= \vec{w}_t - \frac{\eta_t}{1-\beta_1^t}\left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_t}} m_{t-1} + \frac{(1-\beta_{1,t})}{\sqrt{\hat{v}_t}} g_t\right) \end{aligned}$$

$$\vec{w}_{t+1,i} - \vec{w}^*_{,i} = \vec{w}_{t,i} - \vec{w}^*_{,i} - \eta_t \frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}$$

$$(\vec{w}_{t+1,i} - \vec{w}^*_{,i})^2 = (\vec{w}_{t,i} - \vec{w}^*_{,i})^2 - \frac{2\eta_t \hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}} + \eta_t^2 \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}\right)^2$$

$$g_{t,i}\left(\vec{w}_{t,i} - \vec{w}^*_{,i}\right) = \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\eta_t(1-\beta_{1,t})}\left((\vec{w}_{t,i} - \vec{w}^*_{,i})^2 - (\vec{w}_{t+1} - \vec{w}^*_{,i})^2\right)$$

$$
\begin{aligned}
&\underbrace{- \frac{\beta_{1,t}}{(1-\beta_{1,t})} m_{t-1,i} \left(\vec{w}_{t,i} - \vec{w}^{*}_{,i}\right)}_{(*)} \\
&+ \frac{\eta_t \left(1-\beta_1^t\right) \sqrt{\hat{v}_{t,i}}}{2\left(1-\beta_{1,t}\right)} \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}\right)^2
\end{aligned}
$$

In $(*)$ we multiply with $1 = \frac{\hat{v}_{t-1}^{\frac{1}{4}} \sqrt{\eta_{t-1}}}{\hat{v}_{t-1}^{\frac{1}{4}} \sqrt{\eta_{t-1}}}$ and use the binomial equation to simplify:

$$
\begin{aligned}
&\frac{\beta_{1,t}}{1-\beta_{1,t}} \left(\vec{w}^{*}_{,i} - \vec{w}_{t,i}\right) \frac{\hat{v}_{t-1}^{\frac{1}{4}} \sqrt{\eta_{t-1}}}{\hat{v}_{t-1}^{\frac{1}{4}} \sqrt{\eta_{t-1}}} = \\
&= \frac{\beta_{1,t}}{1-\beta_{1,t}} \left( \frac{\hat{v}_{t-1,i}^{\frac{1}{4}}}{\sqrt{\eta_{t-1}}} \left(\vec{w}^{*}_{,i} - \vec{w}_{t,i}\right) \sqrt{\eta_{t-1}} \frac{m_{t-1,i}}{\hat{v}_{t-1,i}^{\frac{1}{4}}} \right) \\
&\leq \underbrace{\frac{\beta_{1,t}}{1-\beta_{1,t}}}_{\leq \frac{\beta_1}{1-\beta_1}} \left( \frac{\sqrt{\hat{v}_{t-1},i} \left(\vec{w}^{*}_{,i} - \vec{w}_{t,i}\right)^2}{2\eta_{t-1}} + \frac{\eta_{t-1} m_{t-1,i}}{2\sqrt{\hat{v}_{t-1,i}}} \right)
\end{aligned}
$$

If we put all these together we reach the following inequality. We separate it in five terms. Each of them will be handled on their own.

$$\underbrace{g_{t,i}\left(\overrightarrow{w}_{t,i}-\overrightarrow{w}^{*}_{,i}\right)}_{(1)} \leq \underbrace{\frac{\left(\left(\overrightarrow{w}_{t,i}-\overrightarrow{w}^{*}_{,i}\right)^{2}-\left(\overrightarrow{w}_{t+1,i}-\overrightarrow{w}^{*}_{,i}\right)^{2}\right)\sqrt{\hat{v}_{t,i}}}{2\eta_{t}\left(1-\beta_{1}\right)}}_{(2)}$$
$$+\underbrace{\frac{\beta_{1,t}}{2\eta_{t-1}\left(1-\beta_{1,t}\right)}\left(\overrightarrow{w}^{*}_{,i}-\overrightarrow{w}_{t,i}\right)^{2}\sqrt{\hat{v}_{t-1,i}}}_{(3)}$$
$$+\underbrace{\frac{\beta_{1}\eta_{t-1}m^{2}_{t-1,i}}{2\left(1-\beta_{1}\right)\sqrt{\hat{v}_{t-1,i}}}}_{(4)}+\underbrace{\frac{\eta_{t}\hat{m}^{2}_{t,i}}{2\left(1-\beta_{1}\right)\sqrt{\hat{v}_{t,i}}}}_{(5)}$$

To get the link to the error sum, we sum over the elements of the gradient $i \in 1, \cdots, d$ and the time stamps $t \in 1, \cdots, T$. Then term (1) looks like:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{d} g_{t,i}\left(\overrightarrow{w}_{t,i}-\overrightarrow{w}^{*}_{,i}\right) &= \sum_{t=1}^{T} g_{t}^{T}\left(\overrightarrow{w}_{t}-\overrightarrow{w}^{*}\right)\\ &\geq \sum_{t=1}^{T}\left(e_{t}\left(\overrightarrow{w}_{t}\right)-e_{t}\left(\overrightarrow{w}^{*}\right)\right)\\ &= R(T)\end{aligned}$$

Now we look at term (2).

$$\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{\left(\left(\vec{w}_{t,i}-\vec{w}^{*}_{,i}\right)^{2}-\left(\vec{w}_{t+1,i}-\vec{w}^{*}_{,i}\right)^{2}\right)\sqrt{\hat{v}_{t,i}}}{2\eta_t\left(1-\beta_1\right)}$$
$$\sum_{i=1}^{d}\frac{1}{2\eta_1\left(1-\beta_1\right)}\left(\vec{w}_{1,i}-\vec{w}^{*}_{,i}\right)^{2}\sqrt{\hat{v}_{1,i}}$$
$$+\sum_{i=1}^{d}\sum_{t=2}^{T}\frac{1}{2\eta_t\left(1-\beta_1\right)}\left(\vec{w}_{1,i}-\vec{w}^{*}_{,i}\right)^{2}\sqrt{\hat{v}_{1,i}}$$
$$\underbrace{-\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{1}{2\eta_t}\left(\vec{w}_{t+1,i}-\vec{w}^{*}_{,i}\right)^{2}\sqrt{\hat{v}_{t,i}}}_{(2a)}$$

We can rewrite (2a):

$$(2a)=\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{1}{2\eta_{t-1}\left(1-\beta_1\right)}\left(\vec{w}_{t,i}-\vec{w}^{*}_{,i}\right)^{2}\sqrt{\hat{v}_{t-1,i}}$$
$$+\sum_{i=1}^{d}\frac{1}{2\eta_T\left(1-\beta_1\right)}\left(\vec{w}_{T+1,i}-\vec{w}^{*}_{,i}\right)^{2}\sqrt{\hat{v}_{T,i}}$$

$$
\begin{aligned}
\textcircled{2} &= \sum_{i=1}^{d} \frac{1}{2\eta_1 (1-\beta_1)} \underbrace{\left(\overrightarrow{w}_{1,i} - \overrightarrow{w}^*_{,i}\right)^2}_{\leq D_\infty^2} \sqrt{\hat{v}_{1,i}} \\
&\quad + \sum_{i=1}^{d} \sum_{t=2}^{T} \frac{1}{2(1-\beta_1)} \underbrace{\left(\overrightarrow{w}_{t,i} - \overrightarrow{w}^*_{,i}\right)^2}_{\leq D_\infty^2} \left( \frac{\sqrt{\hat{v}_{t,i}}}{\eta_t} - \frac{\sqrt{\hat{v}_{t-1,i}}}{\eta_{t-1}} \right) \\
&\quad \underbrace{- \sum_{i=1}^{d} \frac{1}{2\eta_T (1-\beta_1)} \left(\overrightarrow{w}_{T+1,i} - \overrightarrow{w}^*_{,i}\right)^2 \sqrt{\hat{v}_{T,i}}}_{\leq 0} \\
&\quad \leq \frac{D_\infty^2}{2\eta(1-\beta_1)} \left( \sum_{i=1}^{d} \sqrt{\hat{v}_{1,i}} + \sum_{i=1}^{d} \sum_{t=2}^{T} \left( \sqrt{t\hat{v}_{t,i}} - \sqrt{(t-1)\hat{v}_{t-1,i}} \right) \right) \\
&\quad = \frac{D_\infty^2}{2\eta (1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}}
\end{aligned}
$$

Now we look at term $\textcircled{3}$.

$$
\begin{aligned}
\textcircled{3} &\leq \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\beta_{1,t}}{2\eta_t (1-\beta_{1,t})} \left(\overrightarrow{w}^*_{,i} - \overrightarrow{w_{t,i}}\right)^2 \sqrt{\hat{v}_{t-1,i}} \\
&= \frac{1}{2\eta} \sum_{t=1}^{T} \sum_{i=1}^{d} \underbrace{\left(\overrightarrow{w}^*_{,i} - \overrightarrow{w}_{t,i}\right)^2}_{\leq D_\infty^2} \frac{\beta_{1,t}}{1-\beta_{1,t}} \sqrt{t\hat{v}_{t-1,i}} \\
&\leq \frac{D_\infty^2}{2\eta} \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\beta_{1,t}}{(1-\beta_{1,t})} \sqrt{t\hat{v}_{t-1,i}}
\end{aligned}
$$

$$
\begin{aligned}
\sqrt{\hat{v}_{t-1,i}} &= \sqrt{1-\beta_2}\sqrt{\frac{\sum_{j=1}^{t-1} g_{j,i}^2 \beta_2^{t-1-j}}{1-\beta_2^{t-1}}} \\
&\leq \sqrt{1-\beta_2} G_\infty \sqrt{\frac{\sum_{j=1}^{t-1} \beta_2^{t-1-j}}{1-\beta_2^{t-1}}} \\
&\leq \sqrt{1-\beta_2} G_\infty \sqrt{\frac{\sum_{j=1}^{t-1} \beta_2^{j}}{1-\beta_2^{t-1}}} \\
&\leq \sqrt{1-\beta_2} G_\infty \sqrt{\frac{1-\beta_2^{t-1}}{\left(1-\beta_2^{t-1}\right)\left(1-\beta_2\right)}} \\
&\leq G_\infty
\end{aligned}
$$

follows

$$
\textcircled{3} \leq \frac{D_\infty^2 G_\infty}{2\eta} \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\beta_{1,t}}{1-\beta_{1,t}} \sqrt{t}
$$

For $\sum_{t=1}^{T} \frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t}$ we can estimate:

$$
\begin{aligned}
\sum_{t=1}^{T} \frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t} &\leq \sum_{t=1}^{T} \frac{\beta_1 \lambda^{t-1}}{(1-\beta_1)}\sqrt{t} \\
&\leq \sum_{t=1}^{T} \frac{\lambda^{t-1}}{(1-\beta_1)} t \\
&= \frac{1}{1-\beta_1} \sum_{t=0}^{T-1} \lambda^t (t+1) \\
&= \frac{1}{1-\beta_1} \left( \sum_{t=0}^{T-1} \lambda^t t + \sum_{t=0}^{T-1} \lambda^t \right) \\
&= \frac{\left( \frac{(T-1)\lambda^{T+1} - T\lambda^T + \lambda}{(\lambda-1)^2} + \frac{1-\lambda^T}{1-\lambda} \right)}{1-\beta_1} \\
&= \frac{\left( 1 - \underbrace{T\left(\lambda^T - \lambda^{T+1}\right)}_{\geq 0} - \underbrace{\lambda T}_{\geq 0} \right)}{(1-\beta_1)(\lambda-1)^2} \\
&\leq \frac{1}{(1-\beta_1)(\lambda-1)^2}
\end{aligned}
$$

Then (3) results in:

$$
(3) \leq \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty}{2\eta(1-\beta_1)(1-\lambda)^2} = \frac{d D_\infty^2 G_\infty}{2\eta(1-\beta_1)(1-\lambda)^2}
$$

For term (4) we estimate:

$$
\begin{aligned}
(4) &= \frac{\beta_1 \eta}{2\,(1-\beta_1)} \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\hat{m}_{t-1,i}^2}{\sqrt{(t-1)\,\hat{v}_{t-1,i}}} \\
&= \frac{\beta_1 \eta}{2\,(1-\beta_1)} \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\hat{m}_{t-1,i}^2}{\sqrt{(t-1)\,\hat{v}_{t-1,i}}} \underbrace{\left(1-\beta_1^{t-1}\right)^2}_{\leq 1} \\
&\leq \frac{\beta_1 \eta}{2\,(1-\beta_1)} \sum_{i=1}^{d} \frac{2}{(1-\gamma)\sqrt{1-\beta_2}} ||g_{1:T,i}||_2 \\
&= \frac{\beta_1 \eta}{(1-\beta_1)\sqrt{1-\beta_2}\,(1-\gamma)} \sum_{i=1}^{d} ||g_{1:t,i}||_2
\end{aligned}
$$

Analogously to (4), for (5):

$$
\begin{aligned}
\sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\eta_t}{2\,(1-\beta_1)} \frac{\hat{m}_{t,i}^2}{\sqrt{\hat{v}_{t,i}}} &= \frac{\eta}{2\,(1-\beta_1)} \sum_{i=1}^{d} \sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v}_{t,i}}} \\
&\leq \frac{\eta}{2\,(1-\beta_1)} \sum_{i=1}^{d} \frac{2||g_{1:T,i}||_2}{(1-\gamma)\sqrt{1-\beta_2}}
\end{aligned}
$$

$$= \frac{\eta \sum_{i=1}^{d} ||g_{1:T,i}||_2}{(1-\beta_1)\sqrt{1-\beta_2}\,(1-\gamma)}$$

Both in (4) and in (5) we use conjecture 4.2. Now we can combine both.

$$(4) + (5) = \frac{\eta\,(1+\beta_1)}{(1-\beta_1)\sqrt{1-\beta_2}\,(1-\gamma)} \sum_{i=1}^{d} ||g_{1:T,i}||_2$$

If we combine all terms, we get our assertion and the proof is finished.

$$R(T) \leq \frac{D_\infty^2}{2\eta\,(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{dD_\infty^2 G_\infty}{2\eta\,(1-\beta_1)\,(1-\lambda)^2} + \frac{\eta\,(1+\beta_1)}{(1-\beta_1)\sqrt{1-\beta_2}\,(1-\gamma)} \sum_{i=1}^{d} ||g_{1:T,i}||_2$$

Corollaire 3.5 :

*Proof.* The same requirements apply as above. Then the inequality from theorem 4.4 applies and because of $T > 0$ we can divide by $T$.

$$\frac{R(T)}{T} \leq \frac{D_\infty^2}{2\eta\,(1-\beta_1)} \sum_{i=1}^{d} \frac{\sqrt{\hat{v}_{T,i}}}{\sqrt{T}} + \frac{dD_\infty^2 G_\infty}{T2\eta\,(1-\beta_1)\,(1-\lambda)^2} + \frac{\eta\,(1+\beta_1)}{T\,(1-\beta_1)\sqrt{1-\beta_2}\,(1-\gamma)} \sum_{i=1}^{d} ||g_{1:T,i}||_2$$

With

$$\begin{aligned}\sum_{i=1}^{d} ||g_{1:T,i}||_2 &= \sum_{i=1}^{d} \sqrt{g_{1,i}^2 + g_{2,i}^2 + \cdots + g_{T,i}^2} \\ &\leq \sum_{i=1}^{d} \sqrt{G_\infty^2 + G_\infty^2 + \cdots + G_\infty^2} \\ &= \sum_{i=1}^{d} \sqrt{T} G_\infty \\ &= d G_\infty \sqrt{T}\end{aligned}$$

and

$$\begin{aligned}\sum_{i=1}^{d} \sqrt{T \hat{v}_{T,i}} &\leq \sum_{i=1}^{d} \sqrt{T} G_\infty \\ &\leq d G_\infty \sqrt{T}\end{aligned}$$

we can estimate:

$$\lim_{T \to \infty} \frac{R(T)}{T} \leq \lim_{T \to \infty} \left( \frac{1}{\sqrt{T}} + \frac{1}{\sqrt{T}} + \frac{1}{T} \right) = 0$$

This proves the convergence speed $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ of the ADAM-Method. □

## Lemme 3.1 :

**Proof.** We prove (14) by induction.

If $k = 1$, from (12, 13), we have from $\lambda \leqslant 1$ that

$$f(x_1) \leqslant f(x_0) - \sigma(t_0) \leqslant f(x_0) - \lambda\sigma(t_0).$$

Assume (14) holds for $1, 2, \ldots, k$, we consider two cases:

*Case 1*: $\max[f(x_k), \sum_{r=0}^{m(k)-1} \lambda_{kr} f(x_{k-r})] = f(x_k)$, from (12, 13), we have:

$$\begin{aligned} f(x_{k+1}) = f(x_k + \alpha_k d_k) &\leqslant f(x_k) - \sigma(t_k) \\ &\leqslant f(x_0) - \lambda \sum_{r=0}^{k-1} \sigma(t_r) - \sigma(t_k) \\ &\leqslant f(x_0) - \lambda \sum_{r=0}^{k} \sigma(t_r). \end{aligned}$$

*Case 2*: $\max[f(x_k), \sum_{r=0}^{m(k)-1} \lambda_{kr} f(x_{k-r})] = \sum_{r=0}^{m(k)-1} \lambda_{kr} f(x_{k-r})$, let $q = \min[k, M-1]$, again from (12, 13), we have

$$\begin{aligned} f(x_{k+1}) = f(x_k + \alpha_k d_k) &\leqslant \sum_{p=0}^{q} \lambda_{kp} f(x_{k-p}) - \sigma(t_k) \\ &\leqslant \sum_{p=0}^{q} \lambda_{kp} (f(x_0) - \lambda \sum_{r=0}^{k-p-2} \sigma(t_r) - \sigma(t_{k-p-1})) - \sigma(t_k). \end{aligned}$$

Using $(0, 1, 2, \ldots, q) \times (0, 1, 2, \ldots, k-q-2) \subset \{(p, r); 0 \leqslant p \leqslant q, 0 \leqslant r \leqslant k-q-2\}$. $\sum_{p=0}^{q} \lambda_{kp} = 1$, $\lambda_{kp} \geqslant \lambda$, we have

$$\begin{aligned} f(x_{k+1}) &\leqslant f(x_0) - \lambda \sum_{r=0}^{k-q-2} \left( \sum_{p=0}^{q} \lambda_{kp} \right) \sigma(t_r) - \sum_{p=0}^{q} \lambda_{kp} \sigma(t_{k-p-1}) - \sigma(t_k) \\ &\leqslant f(x_0) - \lambda \sum_{r=0}^{k-q-2} \sigma(t_r) - \lambda \sum_{r=k-q-1}^{k-1} \sigma(t_r) - \sigma(t_k) \\ &= f(x_0) - \lambda \sum_{r=0}^{k-1} \sigma(t_r) - \sigma(t_k) \\ &\leqslant f(x_0) - \lambda \sum_{r=0}^{k} \sigma(t_r). \end{aligned}$$

Théorème 3.2 :

**Proof.** From Lemma 1 we know that $x_k \in \mathscr{L}$ for all $k$. Since $f(x)$ is bounded below on $\mathscr{L}$, hence Lemma 1 means

$$\lambda \sum_{r=0}^{k} \sigma(t_r) \leqslant f(x_0) - f(x_{k+1}),$$

let $k \to \infty$, we have

$$\lambda \sum_{r=0}^{\infty} \sigma(t_r) < \infty.$$

Hence, we have

$$\lim_{k\to\infty} \sigma(t_k) = 0,$$

which means from Definition 1 that

$$\lim_{k\to\infty} t_k = \lim_{k\to\infty} \frac{-g_k^{\mathrm{T}} d_k}{\|d_k\|} = 0.$$

Using condition (3) we deduce

$$\lim_{k\to\infty} \sigma(\|g_k\|) = 0,$$

Théoréme 3.5 :

**Proof.** From Lemma 1, we have

$$f(x_k) \leqslant f(x_0) - \lambda \sum_{r=0}^{k-1} \sigma(t_r),$$

let $k \to \infty$, from Assumption 1, we have

$$\lim_{k\to\infty} \sigma(t_k) = 0.$$

Hence, according to Definition 1, we have

$$\lim_{k\to\infty} \frac{g_k^{\mathrm{T}} H_k g_k}{\|d_k\|} = \lim_{k\to\infty} t_k = 0.$$

By the update formula of $B_k$ and $H_k$, we have

$$\mathrm{tr}(B_{k+1}) = n \frac{\|y_k\|^2}{y_k^{\mathrm{T}} s_k}$$

and

$$\mathrm{tr}(H_{k+1}) = (n-2)\frac{y_k^{\mathrm{T}} s_k}{\|y_k\|^2} + 2\frac{\|s_k\|^2}{y_k^{\mathrm{T}} s_k}$$

From Assumption 2, we have

$$y_k^{\mathrm{T}} s_k = \int_0^1 s_k^{\mathrm{T}} G(x_k + t s_k) s_k \, \mathrm{d}t \geqslant c_3 \|s_k\|^2,$$

from which and the Cauchy–Schwartz inequality, we have

$$\frac{y_k^{\mathrm{T}} s_k}{\|y_k\|^2} \leqslant \frac{\|s_k\|^2}{y_k^{\mathrm{T}} s_k} \leqslant \frac{1}{c_3}.$$

$$\lim_{k\to\infty} \frac{g_k^{\mathrm{T}} H_k g_k}{\|d_k\|} = \lim_{k\to\infty} t_k = 0.$$

Alors :

$$\lim_{k\to\infty} \|g_k\| = 0.$$

## Références


[1] – Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9

[2] – Sebastian Bock, Josef Goppold, Martin Weiß. An improvement of the convergence proof of the ADAM-Optimizer. 2018. arXiv:1804.10587v1

[3] – Sashank J. Reddi, Satyen Kale, Sanjiv Kumar. On the Convergence of Adam and Beyond. 2018.

[4] – Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. ICLR Workshop, (1), 2013–2016.

[5] – S. Ruder, "An overview of gradient descent optimization algorithms," cite arxiv:1609.04747Comment: 12 pages, 6 figures. [Online]. Available: http://arxiv.org/abs/1609.04747