Introduction à la méthode de Descente du Gradient et application au Machine Learning

ONANA Damase Donald

Département d'Informatique, Université de Yaoundé I

15 décembre 2021

1 Introduction et Definition

L'algorithme de la descente du gradient est un algorithme d'optimisation continue différentiable (ou de premier ordre) . Il est particulièrement destiné au problème d'optimisation (P) sans contrainte ayant pour fonction objective une fonction $f : R^d \to R$ continue et totalement différentiable. Encore connue sous le nom d'algorithme de la plus forte pente ou de la plus profonde descente (steepest descent, en anglais), la descente du gradient est un algorithme itératif qui procède par amélioration successives et complètement basé sur le calcul du gradient $\nabla_\theta f$ par rapport à un paramètre θ de la fonction f à maximiser ou minimiser.
En effet une des formulations possible du problème d'optimisation (P) est celle de devoir optimiser un paramètre θ dont dépend la fonction f , c'est à dire que l'on cherche à converger vers un θ^* tel que :

$$\frac{\partial f(\theta^*)}{\partial \theta^*} = 0 \tag{1}$$

En d'autres termes , chercher à maximiser ou minimiser f revient à chercher un θ^* appelé point stationnaire tel que $f(\theta^*)$ soit le plus grand possible (pour un problème de maximisation) ou le plus petit possible (pour un problème de minimisation). Pour le faire l'algorithme de descente du gradient fonctionne comme suite :

1. On choisit une valeur initiale θ^0 du paramètre

2. On modifie la valeur courante du paramètre en se déplaçant dans la direction opposée au gradient (pour une minimisation) c'est à dire :

$$\theta^{t+1} = \theta^t - \eta \frac{\partial f(\theta^t)}{\partial \theta^t} \tag{2}$$

où η est le pas ou encore taux d'apprentissage qui détermine la vitesse de convergence. On en reparlera mieux aux sections suivantes .

3. L'on répète l'étape 2 jusqu'à la convergence .

Notons que nous avons juste considéré le cas d'une minimisation comme nous le ferons la plus part du temps aux sections suivantes , tout en se rappelant que pour maximiser une fonction il suffit de minimiser son opposé ou alors dans notre cas se déplacer dans la direction du gradient plutôt que dans la direction opposée comme nous l'avons fait à l'étape 2. Il est tout aussi important de noter que le point stationnaire θ^* trouvé est en fait soit un point de minimum local ou global (pour un problème de minimisation) , soit un point de maximum local ou global (pour un problème de maximisation) ou alors un point selle (qui est maximum et minimum). Si notre fonction f est une fonction convexe alors l'optimisation sera toujours parfaite et donc θ^* sera toujours un point de maximum ou de minimum global , dans le cas ou f est une fonction non convexe voir fortement non convexe l'on trouvera souvent voir toujours des points de maximum ou de minimum locaux.

Cet algorithme fait aujourd'hui parti des algorithmes d'optimisation les plus populaire et les plus utilisés notamment en informatique pour des problèmes d'apprentissage automatique (*machine learning*). En effet une formulation mathématique d'un problème d'apprentissage automatique nous renvoi à un problème d'optimisation avec ou sans contrainte , dans ce cas l'algorithme de descente du gradient associé aux réseaux de neurones est aujourd'hui jugée comme une des méthodes la plus satisfaisante pour résoudre ce genre de problème. Une des preuves de la popularité de cet algorithme est le fait qu'il est implémenté dans toutes bibliothèques (logiciel) traitant des problèmes d'apprentissage automatique profond (deep learning) . La descente du gradient du haut de sa popularité et de son efficacité mérite une attention particulière c'est à dire une étude du concept, des ses différentes variantes , de leurs défauts et qualités ainsi que sur les améliorations apportées .

2 Les variantes de la Descente de Gradient

L'on distingue principalement 03 variantes de l'algorithme :

- Le bacht ou standart gradient descent ;
- le stochastic ou online gradient descent ;
- le mini-bacht gradient descent

il serait difficile de dire qu'une variante est largement par rapport aux autres , tous dépend d'un certains nombres de paramètres comme par exemple le nombre de données disponible, l'environnement d'exécution ou la qualité du résultat voulue.

2.1 Bacht Gradient Descent

Le bacht gradient descent encore appelé standart gradient descent est comme son nom l'indique la version standard de l'algorithme. Ici, l'on calcul le gradient de la fonction objective par rapport à θ en considérant l'ensemble de données d'entrainement disponible .

$$\theta = \theta - \eta \nabla_\theta f(\theta) \tag{3}$$

Cette variante a la particularité d'être lente surtout lorsque le jeu de données d'entrainement est très grand, dans ce cas il est préférable d'utiliser l'une des deux autres variantes .
Le pseudo code suivant d'écrit plus en détail la variante bacht de la descente du gradient.

Algorithm 1 BGD($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta$)

```
Initialiser θ^0
E_0 = 0
for i = 1 to n_epoch do
  for all (x_d, y_d) in D do
    y_pred = get_prediction(θ^i, x_d)
    E_t = E_{t-1} + get_error(y_pred, y_d)
  end for
  caluler ∇E(θ)
  θ^i = θ^{i-1} − η∇E(θ)
end for
return θ^i
```

2.2 Stochastic Gradient Descent

L'algorithme stochastic de descente du gradient contrairement à la variante bacht, calcul le gradient de la fonction objective par rapport à θ pour chaque instance (x_i, y_i) des données d'entrainement .

$$\theta = \theta - \eta \nabla_\theta f(\theta, x_i, y_i) \tag{4}$$

Le fait que la variante stochastic calcul le gradient et mette directement à jour le paramètre pour chaque instance (x_i, y_i) des données d'entrainement, rend la convergence largement plus rapide qu'à la version bacht, mais par contre elle sera un peu moins précise.
Le pseudo code suivant d'écrit la variante stochastic de la descente du gradient.

Algorithm 2 SGD($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta$)

```
Initialiser θ^0
E_0 = 0
for i = 1 to n_epoch do
  shuffle(D)
  for all (x_d, y_d) in D do
    y_pred = get_prediction(θ^i, x_d)
    E_t = E_{t-1} + get_error(y_pred, y_d)
    caluler ∇E(θ)
    θ^i = θ^{i-1} − η∇E(θ)
  end for
end for
return θ^i
```

2.3 Mini-Bacht Gradient Descent

La version mini-bacht de la descente du gradient essaie de prendre le meilleur de la version bacth et stochastic . Ici , l'on calcul le gradient et on met à jour le paramètre pour chaque mini lots de l'ensemble de données d'entrainement.

$$\theta = \theta - \eta \nabla_\theta f(\theta, x_{i:i+n}, y_{i:i+n}) \tag{5}$$

La véritable difficulté ici est de trouver la valeur de l'hyperparamètre n qui définit la taille de chaque lot (Bacht size en anglais) . Comme le dit Sebastian Ruder dans [1] l'on prend communément une valeur comprise entre 50 et 256 pour cet hyperparamètre cela dépend vraiment de la tache qu'on effectue et de l'objectif à atteindre. Mais une fois qu'aprés plusieurs expérience et observation l'on parvient à trouver la valeur adéquate , la version mini-bacht offre des avantages tels que la diminution de la fréquence de mise à jour ce qui à pour conséquence de rendre la convergence plus stable, ou alors prise en compte du calcul vectoriel possible qui vas rendre le calcul du gradient beaucoup plus rapide et simple.
Le pseudo code suivant d'écrit la variante mini-bacht de la descente du gradient.

Algorithm 3 mini_bacht($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta, bacht\_size$)

```
Initialiser θ^0
E_0 = 0
for i = 1 to n_epoch do
  shuffle(D)
  for bacht in get_bacht(D, bacht_size) do
    for all (x_d, y_d) in bacht do
      y_pred = get_prediction(θ^i, x_d)
      E_t = E_{t-1} + get_error(y_pred, y_d)
      caluler ∇E(θ)
      θ^i = θ^{i-1} − η∇E(θ)
    end for
  end for
end for
return θ^i
```

3 Application à un Problème de machine Learning

Dans cette section nous allons appliquer ce qui a été vue au section précédente pour essayer de résoudre un problème de machine learning plus précisément le problème de régression linéaire . Nous commençons d'abord par présenté la formulation du problème , ensuite nous utiliserons successivement les 3 variantes de l'algorithme de descente du gradient pour le résoudre. Et à la section suivante , nous effectuerons une étude comparative des résultats et des performances de chacune des variantes.

3.1 Le probléme de regression linéaire

Comme définit dans [2], l'analyse pas régression est à l'initial une technique statistique permettant d'estimer une possible relation de causes à effet entre des variables. De manière un peu plus explicite dans [3] l'on définit l'analyse par régression comme étant une méthode pour découvrir la relation entre une ou plusieurs variables de réponses (également appelées variables dépendantes, variables expliquées, variables prédites ou régressandes, généralement désignées par y) et des variables prédicteurs (également appelés variables indépendantes, variables explicatives, variables de contrôle ou régresseurs, généralement désignés par $x_1, x_2, ..., x_p$).

Dans la littérature, il existe 03 type de regression, à savoir :

– La régression linéaire simple ou univarié : où on essai de modéliser une relation linéaire entre deux variables, l'une d'elle étant la variable dépendante (y) et l'autre la variable indépendante (x). Elle est très souvent formalisée par l'équation suivante

$$y = \beta_1 x + \beta_0 + \epsilon \tag{6}$$

– La régression linéaire multiple : où l'on cherche à modéliser une relation relation linéaire entre une variable dependente et plusieurs variables indépendante. Formellement on a

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \epsilon \tag{7}$$

– La régression non linéaire : l'on suppose que la relation entre la variable dépendante et les variables indépendantes est non linéaire . Un exemple de modèle de régression non linéaire (modèle de croissance) peut s'écrire :

$$y = \frac{\alpha}{1 + e^{\beta t}} + \epsilon \tag{8}$$

où y est la croissance d'un organisme particulier en fonction du temps t, β et α sont les paramètres du modèle, et ϵ est l'erreur aléatoire.

Les objectifs de l'analyse de régression sont triples :

1. Pouvoir établir une relation entre la variable dépendante et les variables indépendante ;

2. Pouvoir prédire la variables dépendante à partir des variables indépendante ;

3. Observer les variables indépendantes et dire lesquelles sont importantes pour expliquer la variables dépendante

Dans notre étude , nous ferons face à un problème de régression linéaire simple et nous supposerons qu'il existe une relation linéaire entre les deux variables, notre défis sera donc de pouvoir prédire les variables dépendante à partir des variables indépendantes .

3.2 Résolution par la méthode Bacht

L'on rappelle que notre objectif est de pouvoir trouver β_0^* et β_1^* de manière à minimiser au tant que possible la fonction d'erreur $E = (y_i - y_{pred}^i)$ avec $y_{pred}^i = \beta_1 x_i + \beta_0$. On applique donc l'algorithme 1 et on trouve finalement $\beta_0 = 1.15$, $\beta_1 = 71.35$. La figure suivante nous montre la représentation du nuage de points et la droite (D) d'équation $y = \beta_1 x_i^* + \beta_0^*$ (pour i de 1 à n).

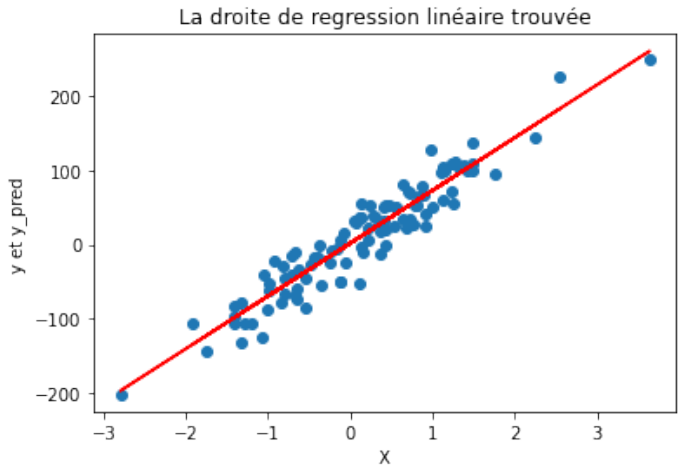

FIGURE 1 – Le nuage de point et la droite de régression trouvée avec la méthode bacht

La droite (D) représenté en rouge est donc appelé droite de régression de y en x , elle représente la relation linéaire entre les variables y_i et x_i. On observe ensuite à l'image suivante l'évolution de l'erreur E au cour des itérations.

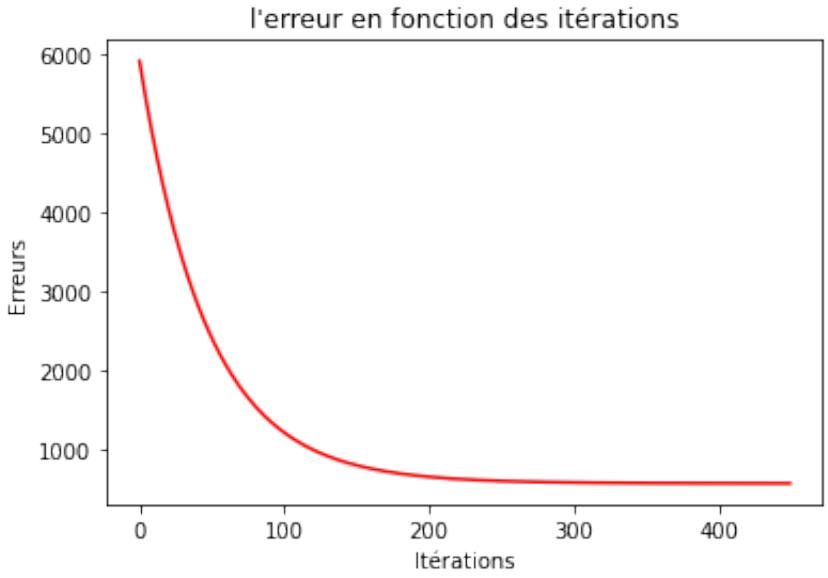

FIGURE 2 – Evolution de l'erreur en fonction des itérations (méthode Bacht)

l'on voix bien qu'à la première itération, on a une erreur assez grande et au cour de l'exécution de l'algorithme on optimise les paramètres β_0 et β_1 pour finalement obtenir à la dernière itération une erreur plus petite (jugée la plus petite possible) .

3.3 Résolution par la méthode Stochastic

Comme nous l'avons fait pour la variante bacht, on implémente aussi la variante stochastic présenté à l'algorithme 2 . Et on trouve $\beta_0 = 1.316$, $\beta_1 = 71.294$. La figure suivant nous montre le nuage de point et la droite de régression trouvé .

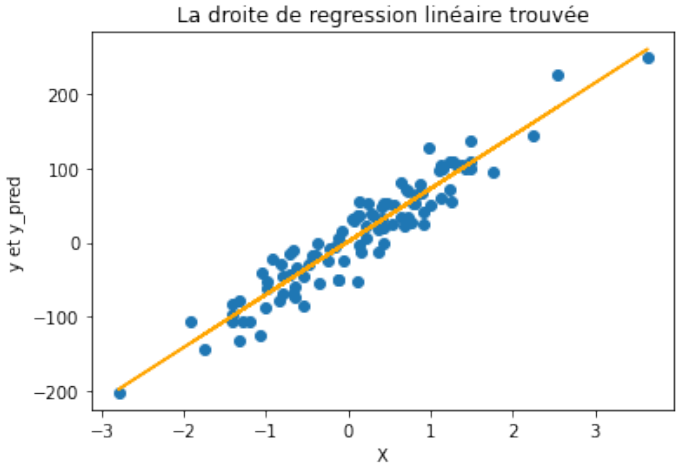

FIGURE 3 – Le nuage de point et la droite de régression trouvée avec la méthode Stochastic

On observe ensuite à l'image suivante l'évolution de l'erreur E au cour des itérations pour

la version stochastic .

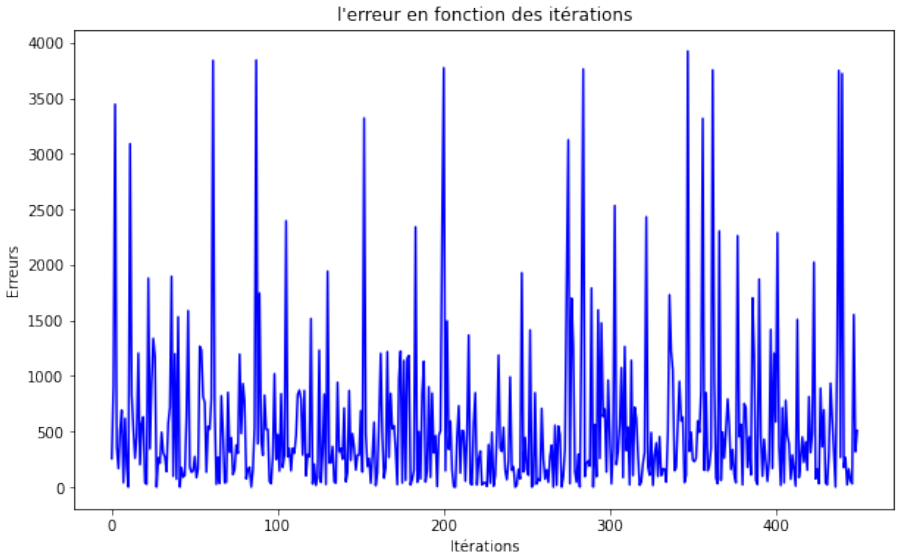

FIGURE 4 – Evolution de l'erreur en fonction des itérations (méthode stochastic)

Cette figure confirme bel et bien nos propos précédent qui disait que , avec la méthode stochactic on observe beaucoup de fluctuation de la fonction objective (dans ce cas la fonction d'erreur) ce qui rend convergence moins stable.

3.4 Résolution par la méthode Mini Bacht

Apres avoir implémenter la version bacht pour un bacht_size de 25 pour un jeu de données de taille 100 , on trouve $\beta_0 = 1.1362$, $\beta_1 = 71.8050$. La figure suivant nous montre le nuage de point et la droite de régression trouvé .

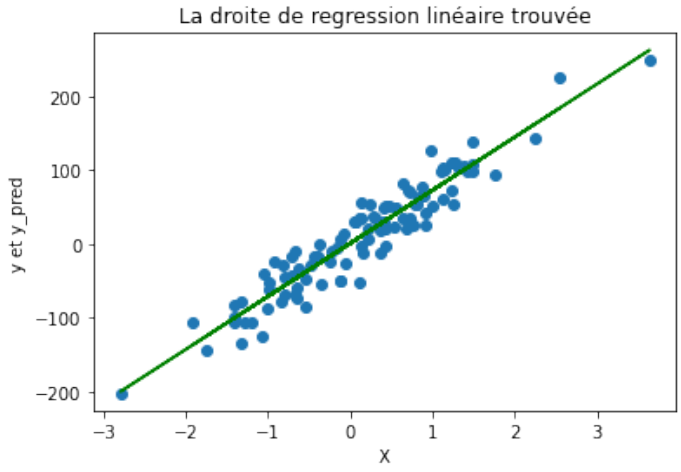

FIGURE 5 – Le nuage de point et la droite de régression trouvée avec la méthode Stochastic

On observe ensuite à l'image suivante l'évolution de l'erreur E au cour des itérations pour la version mini_bacht

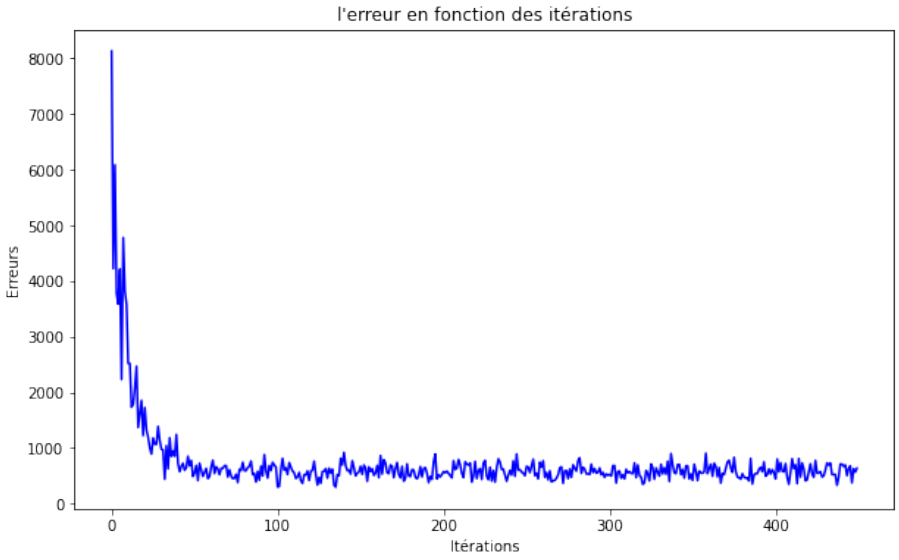

FIGURE 6 – Evolution de l'erreur en fonction des itérations (méthode stochastic)

4 Etude Comparative

Dans cette section nous analyserons la qualités des résultats aux sections précédentes ainsi que la manière dont ils ont été obtenus pour chacune des variantes et faire des comparaison .

Nous commençons pas présenter un tableau récapitulatif des résultats obtenus .

	Bacht	stochastic	Mini-Bacht
MSE	583.924	348.879	688.295
Nombre d'itérations (fixés)	450	450	450
Nombres de mise à jour	450	45000	1800
β_1	71.0745	71.6831	71.8122
β_0	1.5600	1.1194	1.1287

Le premier constat est que, bien que pour cette expérience le MSE de la variante stochastique est inférieur aux autres elle est par contre beaucoup moins stable dû au très grand nombres de mise à jour . Avec un Bacht size de 25 , la convergence version mini bacht est un peu plus stable et si on augmente la valeur du bacht size , on se rapprocherait du comportement exactement de la variante bacht . Pareillement si l'on diminue le bacht size on se rapproche de la variante stochastique avec une convergence moins stable , la figure suivante illustre un peu tout cela .

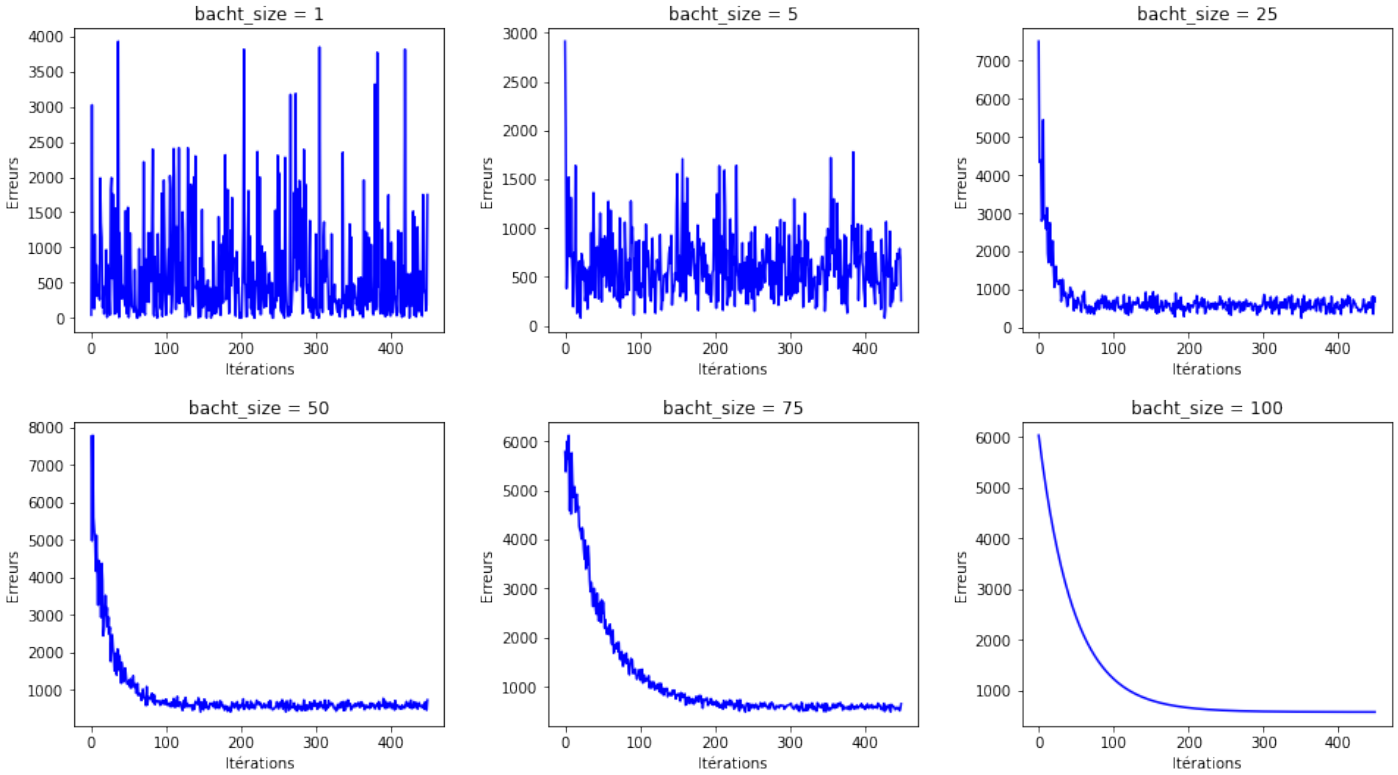

FIGURE 7 – La méthode mini-bacht avec différents bacht size

L'on constate qu'avec un bach size égale à 1 , la version mini bacht et la version stochastique sont semblable pareil lorsque le bach size grimpe jusqu'à 100 la version bacht et la version mini bacht deviennent semblable.

Ensuite pour apprécier encore plus la qualité et la vitesse de convergence de chacune des variantes , nous devons ajouter une condition d'arrêt outre que celle sur les itérations . Il s'agit de la condition d'arrêt sur l'amélioration insuffisante , formellement l'algorithme s'arrêtera lorsque la condition suivante sera respecté :

Références

[1] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv :1609.04747*, 2016.

[2] Gülden Kaya Uyanık and Neşe Güler. A study on multiple linear regression analysis. *Procedia-Social and Behavioral Sciences*, 106 :234–240, 2013.

[3] Xin Yan and Xiaogang Su. *Linear regression analysis : theory and computing.* World Scientific, 2009.