Méthodes fiables et robustes pour l'apprentissage machine.

Rodolphe Turpault (Bordeaux-INP/IMB)

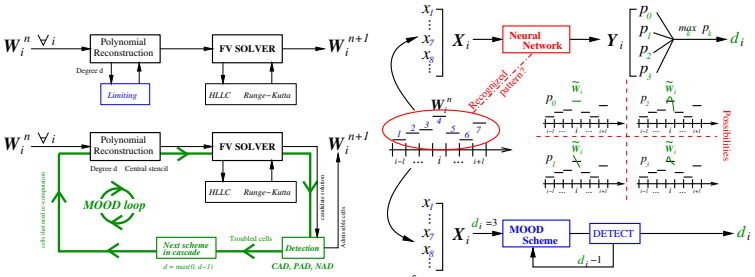
avec Bilel Bensaid et Gael Poette (CEA/Bordeaux-INP/IMB)

3 décembre 2024

1. Deux exemples pratiques
2. Amélioration de l'apprentissage
3. Apprentissage par lots
4. Mise en perspective: quelques écueils à éviter
5. Conclusion et perspectives

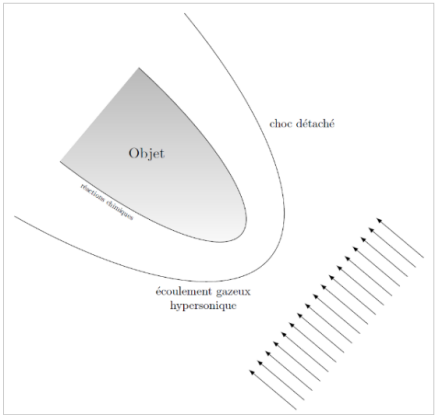
Amélioration d'un schéma numérique

A. Bourriaud, R. Loubère, R. Turpault [1]



Un problème de rentrée atmosphérique

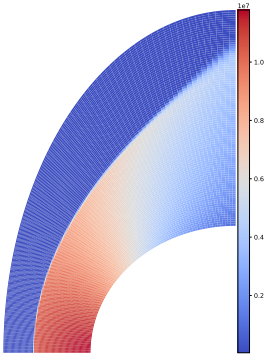
P. Novello, G. Poette, D. Lugato, S. Peluchon, P. Congedo [2]



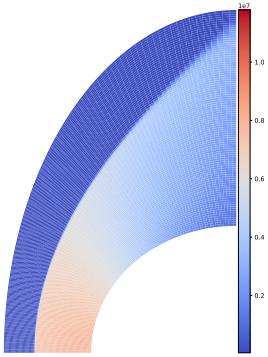
Enjeu: conception d'engins entrants dans l'atmosphère

Modèle PG vs. Modèle M_{++} vs. modèle hybride

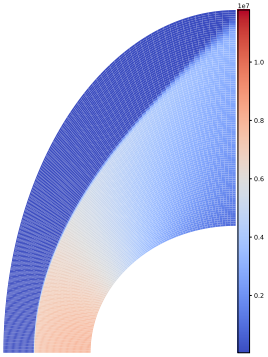
(cartes de pression pour plusieurs modèles de réactions chimiques)



PG: 80s.



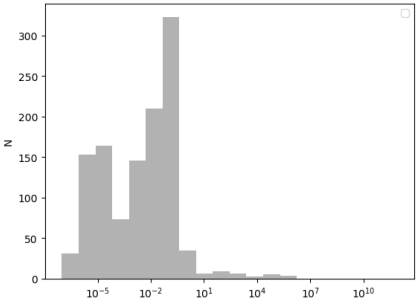
M_{++}: 4090s.



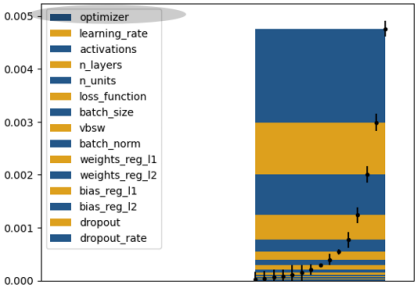
NN: 136s. (gain $\times 30$)

Contexte et motivations

Construire des réseaux de neurones efficaces n'est pas toujours facile [3]



Distrib. des erreurs



Sensibilité aux hyperparamètres

- Des erreurs allant de 10^{-6} à 10^{7}... Beaucoup d'apprentissages instables (NaN)...
- Construire un réseau efficace nécessite beaucoup de temps "ingénieur"...
- L'algorithme d'apprentissage est l'hyperparamètre le plus important...

Principe de l'apprentissage

Cas supervisé: on se donne une base de données (d'entraînement):

$$\left(X_k, \bar{Y}_k\right)_{k=1..P},$$

et on cherche les paramètres optimaux du modèle:

$$(w^\star, b^\star) = \text{argmin}_{(w,b)}\ \mathcal{R}(w, b),$$

où $\mathcal{R}$ est une fonction de coût, par exemple (MSE):

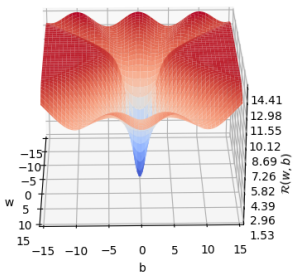
$$\mathcal{R}(w, b) = \frac{1}{2P} \sum_{k=1}^{P} \|u_{(w,b)}(X_k) - \bar{Y}_k\|^2$$

$\rightarrow$ Pb d'optimisation (non-convexe) en grande dimension!

Exemples

Des fonctions non-convexes en deux dimensions

$$\mathcal{R}(w,b) = \sum_{i=1}^{50} \|\sigma(wx_i + b) - (0.5 + 0.25\sin(3\pi x_i))\|^2 \text{ for } x_i = -1 + i\frac{2}{50}.$$

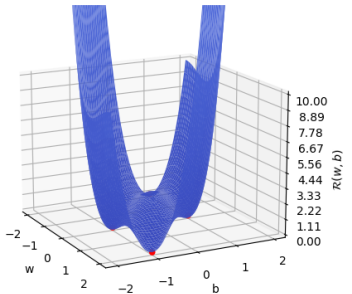


$\Longrightarrow$ plusieurs minima locaux, des endroits plats, des points-selles, des maxima...

Exemples

Des fonctions non-convexes en deux dimensions

$$\mathcal{R}(w, b) = g(w + b)^2 + g(b)^2, \; g(z) = z^2 - 1 \text{ (Benchmark \#1)}.$$



$\Longrightarrow$ plusieurs minima locaux **raides**, (des points selles,) des maxima...

Comment trouver un minimum?

On utilise une méthode numérique itérative (optimiseur):

Partant d'une initialisation $\theta_0 = (w_0, b_0)$, on construit une suite $\theta_n = (w_n, b_n)$ dont on espère qu'elle converge vers le minimum.

Exemple: descente de gradient (GD):

$$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{R}(\theta_n),$$

où $\eta > 0$ est le pas de descente (ou *learning rate* dans ce contexte).

Comment trouver un minimum?

Problèmes pratiques:

1. Comment calculer le gradient de $\mathcal{R}$?
2. Comment choisir η?
3. Quel critère d'arrêt?
4. Quelle initialisation?

Qq optimiseurs “classiques”

(4 sont présentés ici mais déjà autant de paramètres à régler)

Gradient Descent (GD) de paramètre η

$$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{R}(\theta_n),\ 0 < \eta \leq 1(?).$$

Momentum de paramètres η et β_1 [4]

$$\begin{cases} v_{n+1} = (1-\beta_1) v_n + \beta_1 \nabla \mathcal{R}(\theta_n),\ 0 < \beta_1 \leq 1, \\ \theta_{n+1} = \theta_n - \eta v_{n+1}. \end{cases}$$

Adam without bias (AWB) de paramètres η, β_1 ϵ_a et β_2 [5]

$$\begin{cases} v_{n+1} = (1-\beta_1) v_n + \beta_1 \nabla \mathcal{R}(\theta_n), \\ S_{n+1} = (1-\beta_2) S_n + \beta_2 \nabla \mathcal{R}(\theta_n)^2,\ 0 < \beta_2 \leq 1 \\ \theta_{n+1} = \theta_n - \eta \frac{v_{n+1}}{\sqrt{S_{n+1}+\epsilon_a}}. \end{cases}$$

Adam de paramètres η, β_1 ϵ_a et β_2 [5]

$$\begin{cases} v_{n+1} = (1-\beta_1) v_n + \beta_1 \nabla \mathcal{R}(\theta_n), \\ \hat{v}_{n+1} = \frac{v_{n+1}}{1-(1-\beta_1)^n}, \\ S_{n+1} = (1-\beta_2) S_n + \beta_2 \nabla \mathcal{R}(\theta_n)^2, \\ \hat{S}_{n+1} = \frac{S_{n+1}}{1-(1-\beta_2)^n}, \\ \theta_{n+1} = \theta_n - \eta \frac{\hat{v}_{n+1}}{\sqrt{\hat{S}_{n+1}+\epsilon_a}}. \end{cases}$$

Comment les optimiseurs se comportent-ils en pratique?

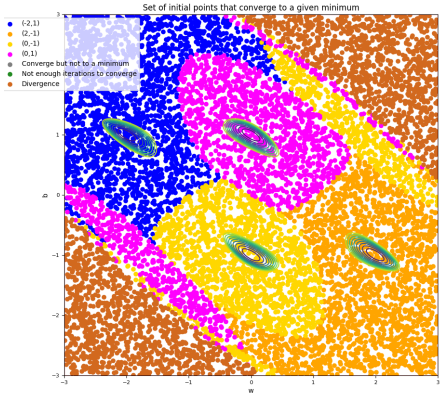


Figure: Sensibilité à la condition initiale, benchmark #1 pour GD

Comment les optimiseurs se comportent-ils en pratique?

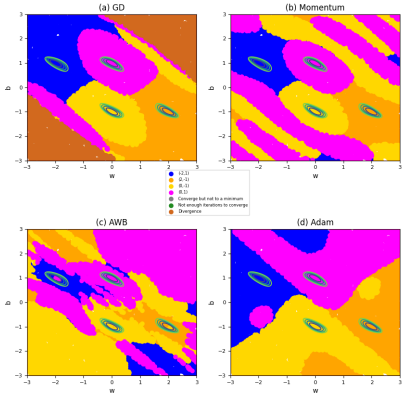


Figure: Sensibilité à la condition initiale, benchmark #1 pour 4 optimiseurs

Comment les optimiseurs se comportent-ils en pratique?

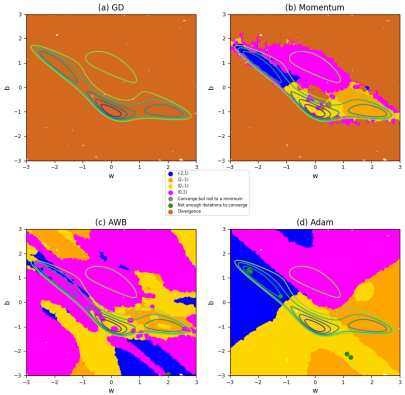


Figure: Sensibilité à la condition initiale, benchmark #2 pour 4 optimiseurs

Comment les optimiseurs se comportent-ils en pratique?

Pourquoi ne sommes nous pas satisfaits des comportements précédents:

- Le comportement divergent est problématique
  (pour certains entrainements, plus de 70% de NaN!)
- Les instabilités ruinent toutes les connaissances *a priori* sur θ que l'on pourrait avoir
  (Initialisations Xavier/Bengio/He ou régularisation L1/L2 = *a priori* sur la position des minima)
  (PINNS par exemple, construisent une régularisation basée sur la physique)
  (Le *Transfer Learning* est basé sur l'utilisation de réseaux pré-entrainés)
- Le réglage des paramètres des optimiseurs est très pénible pour un utilisateur.
  (... éviter les comportements précédents est très difficile, voire impossible ...)
  (... beaucoup d'essais/erreurs ...)
  (... cela prend beaucoup de temps "ingénieur" - "machine learning" ou "user learning"?)

Idée-clé

Les optimiseurs classiques peuvent être vus comme des discrétisations d'EDO.
Cas de GD:

$$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{R}(\theta_n),$$

Schéma d'Euler explicite consistant avec:

$$\theta^{'}(t) = -\nabla \mathcal{R}(\theta(t)).$$

On peut utiliser la théorie sur les EDO. Ici,

$$V(\theta) := \mathcal{R}(\theta)(-\mathcal{R}(\theta^{\star}))$$

est une fonction de Lyapunov (énergie/entropie):

$$\dot{V}(\theta) = -\|\nabla \mathcal{R}(\theta)\|^2.$$

Que se passe-t-il en pratique?

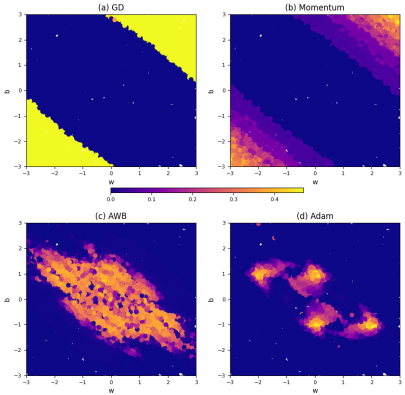


Figure: Décroissance de V (proportion), benchmark #1 pour 4 optimiseurs

LCD

On peut la forcer grâce à un pas adaptatif. Pour corriger GD:

1. η_n est choisi tq $\mathcal{R}\Big(\theta_n + \eta_n \nabla\mathcal{R}(\theta_n)\Big) - \mathcal{R}(\theta_n) < -\lambda\eta_n \|\nabla\mathcal{R}(\theta_n)\|^2$,[1]
2. $\theta_{n+1} = \theta_n - \eta_n \nabla\mathcal{R}(\theta_n)$,

En pratique l'étape 1 s'effectue faisant une dichotomie (de facteur $f_1 > 1$), puis avant chaque itération, on multiplie η_n par un facteur $f_2 > 1$.

Typiquement, $\lambda = 0.5$, $f_1 = 30$ et $f_2 = 10000$.

[1] Pour GD, on retrouve ici la condition d'Armijo.

Retour sur l'exemple précédent

benchmark #2

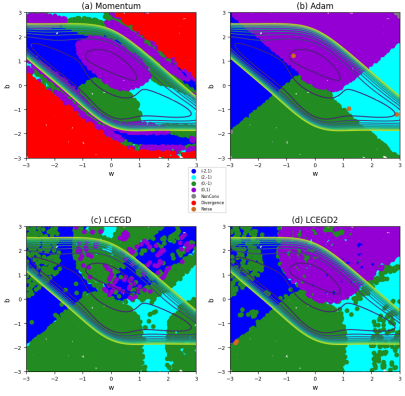


Figure: Les optimiseurs à pas adaptatif n'ont plus d'instabilité de Lyapunov.

Autre exemple plus difficile

benchmark #3

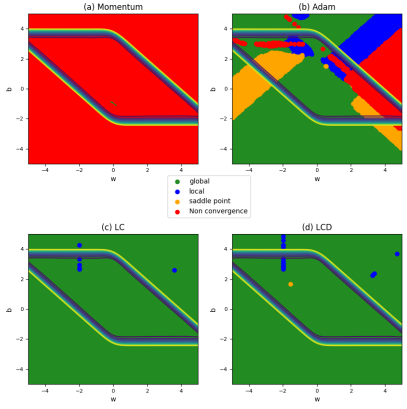


Figure: Pas de divergences, stabilité de Lyapunov (et pas de pt-selle capté)

Quid des nouveaux hyperparamètres?

On montre qu'ils ne sont pas sensibles. Par exemple (hypothèses omises):

Theorem

Le nombre d'évaluations moyen de la fonction de Lyapunov par LCD vérifie:

$$\frac{C_n}{n} \leq \left[1 + \frac{\log(f_2)}{\log(f_1)}\right] + \frac{1}{n}\frac{\log\left(\frac{f_1^2 \eta_0}{\eta_\star}\right)}{\log(f_1)}$$

Theorem (Taux de convergence au voisinage d'un point critique $\theta^\star$)

Si on note $S_n := \sum_{k=0}^{n-1} \eta_k$ *et* α *le coefficient de Łojasiewicz:*

$$\|\theta_n - \theta^\star\| = \begin{cases} \mathcal{O}\Big(\exp\big(-\lambda\alpha c^2 S_n\big)\Big), \ \textit{si } \alpha = \frac{1}{2}, \\ \mathcal{O}\Big(\big[cte + \lambda c^2(1-2\alpha)S_n\big]\Big)^{-\frac{\alpha}{1-2\alpha}}, \ \textit{si } \alpha \in]0, \frac{1}{2}[. \end{cases}$$

Quelques exemples plus classiques en ML

Boston Housing - [13(tanh)15(tanh)15(lin)1]

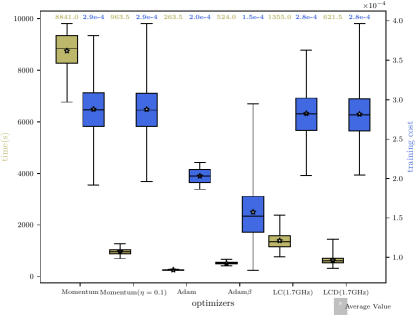


Figure: Performances (Boston Housing)

Quelques exemples plus classiques en ML

Boston Housing - [13(tanh)15(tanh)15(lin)1]

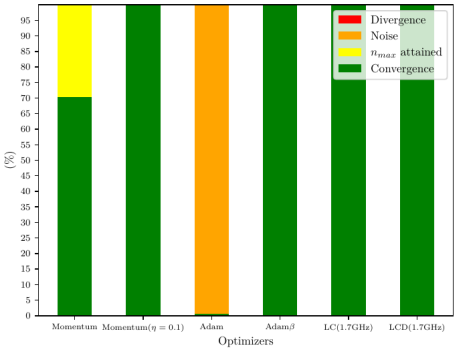


Figure: Convergence des trajectoires (Boston Housing)

Quelques exemples plus classiques en ML

Sonar - [60(GELU)30(sigmoid)1]

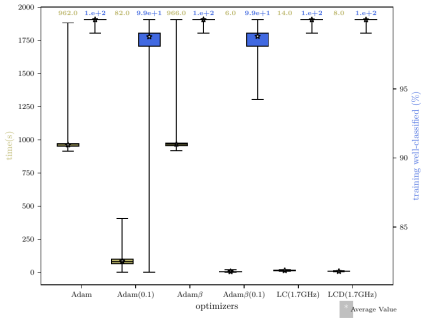


Figure: Performances (Sonar)

Quelques exemples plus classiques en ML

Sonar - [60(GELU)30(sigmoid)1]

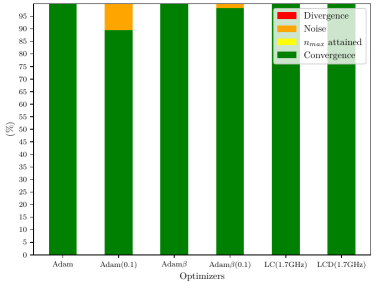


Figure: Convergence des trajectoires (Sonar)

Quelques exemples plus classiques en ML

MNIST - [754(tanh)24(softmax)10]

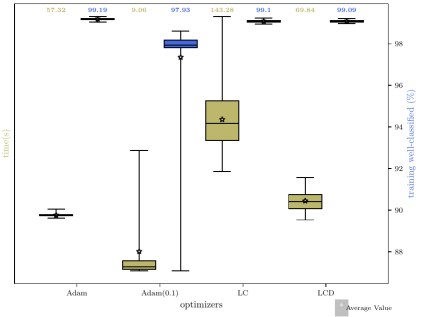


Figure: Performances (MNIST)

Quelques exemples plus classiques en ML

FashionMNIST - LeNet1

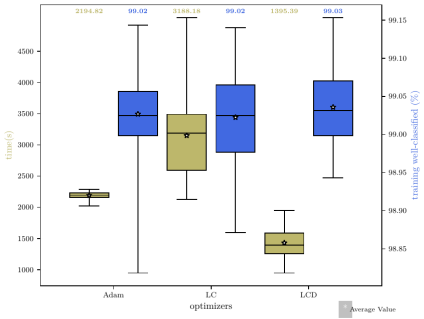


Figure: Performances (Fashion MNIST)

Un bref résumé des résultats de la thèse de Bilel Bensaid

Les résultats en bref sur cette partie ([7]):

- Les nouveaux optimiseurs sont stables, attractifs, convergents
  (plusieurs théorèmes en témoignent $\Longrightarrow$ garanties)
- Les entrainements sont plus rapides $\Longrightarrow$ frugalité
- Pas de réglages nécessaires par les utilisateurs $\Longrightarrow$ pas de pb d'explicabilité du comportement
- Les nouveaux optimiseurs atteignent parfois de meilleures précisions que les classiques
  (parfois, des architectures moins lourdes sont obtenues pour la même précision $\Longrightarrow$ frugalité)

Position du problème

On peut découper les données en m lots (*mini-batches*):

$$\mathcal{R}(\theta) = \sum_{i=1}^{m} \mathcal{R}_i(\theta).$$

Dans ce cas, à chaque itération, on n'a accès qu'à la i^e partie des données.
Un intérêt pratique: moins de demande mémoire.

Remarque: les méthodes stochastiques sont basées sur ce principe (pour d'autres raisons).

Cas pratique

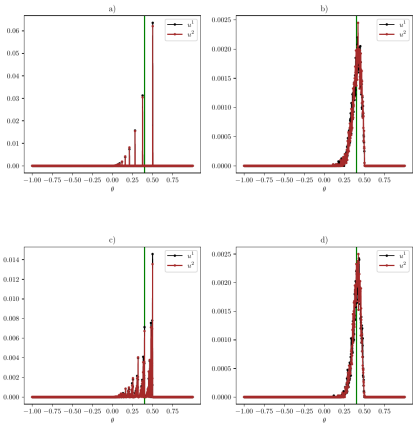


Figure: Benchmark par lots #1 pour SGD avec différents pas de temps $\eta =$ a)$(\max(L_1, L_2))^{-1}$, b) 1/3 a), c) $(L_1 + L_2)^{-1}$, d) 1/3 c)

Explication

Dans le cas déterministe avec 2 lots, on résout:

$$\begin{aligned}\theta_{n+1/2} &= \theta_n - \eta \nabla \mathcal{R}_1(\theta_n),\\ \theta_{n+1} &= \theta_{n+1/2} - \eta \nabla \mathcal{R}_2(\theta_{n+1/2}).\end{aligned}$$

L'état stationnaire est alors:

$$\theta_\infty = \frac{2}{5-8\eta} \neq \theta^\star = \frac{2}{5}.$$

Le problème vient du fait que l'on utilise une méthode de *splitting* non équilibrée.

Splitting équilibré

Schéma de Speth [8]:

$$\begin{aligned}\theta_{n+1/2} &= \theta_n - \eta\nabla\mathcal{R}_1(\theta_n) + \eta c_n,\\ \theta_{n+1} &= \theta_{n+1/2} - \eta\nabla\mathcal{R}_2(\theta_{n+1/2}) - \eta c_n,\\ c_{n+1} &= c_n + \frac{1}{2\eta}\big(\theta_{n+1} + \theta_n - 2\theta_{n+1/2}\big).\end{aligned}$$

On obtient à la limite: $\nabla\mathcal{R}(\theta_\infty) = 0$.

Cette idée peut être (astucieusement) adaptée à LCD pour obtenir un optimiseur par lots équilibré: RAG.

Retour sur le cas pratique

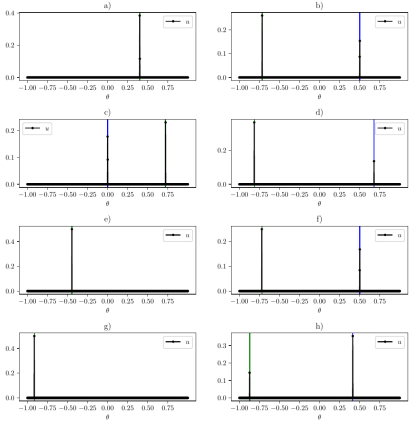


Figure: Résultats de RAG pour les 8 benchmarks par lots.

Résultats numériques

Benchmark # 3

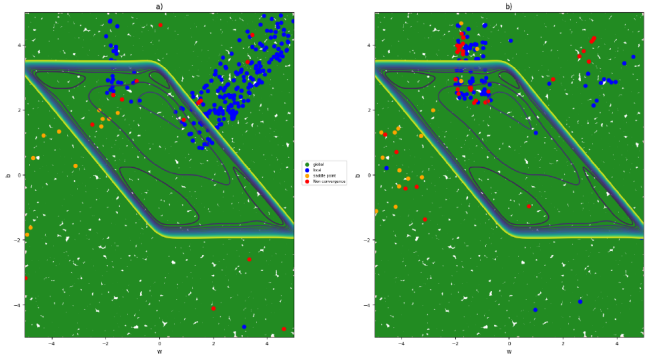


Figure: Résultats de RAG et RAGL sur le benchmark # 3.

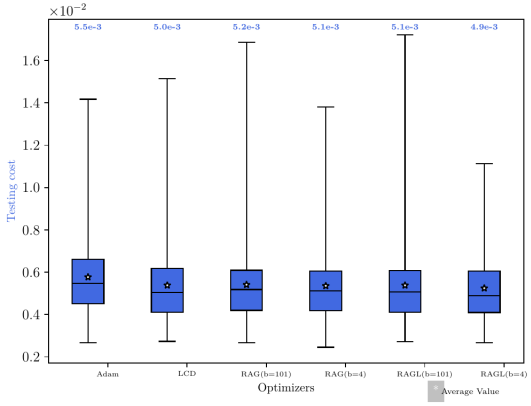


Figure: Performances (Boston Housing).

Résultats numériques

Boston Housing

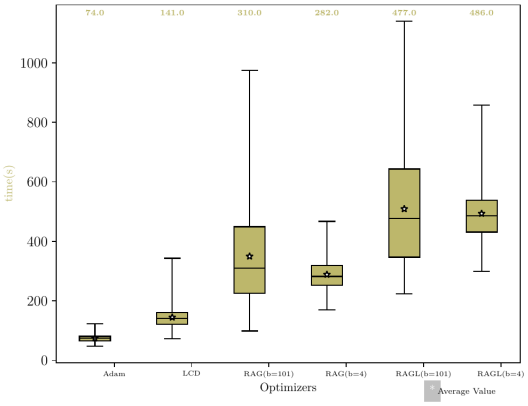


Figure: Temps de calcul (Boston Housing).

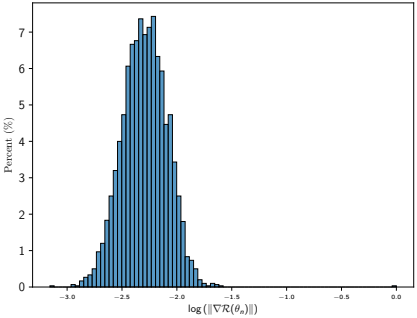


Figure: RRA - distribution des gradients avec $b = 4$

Quelques écueils à éviter

- Attention aux légendes urbaines!
- Bigger is better.
- Ne pas correctement valider.
- Surinterpréter.

Attention aux légendes urbaines!

(un exemple parmi d'autres)

Fait communément accepté dans la littérature[2]:

"... seule la stochasticité permet de s'échapper des points selles"

[2] traduction d'une phrase trouvée dans une publication de référence

Attention aux légendes urbaines!

(un exemple parmi d'autres)

Fait communément accepté dans la littérature[2]:

"... seule la stochasticité permet de s'échapper des points selles"

C'est faux! En tout cas ce n'est pas la seule solution ...

[2] traduction d'une phrase trouvée dans une publication de référence

Attention aux légendes urbaines!

(un exemple parmi d'autres)

Fait communément accepté dans la littérature[2]:

"... seule la stochasticité permet de s'échapper des points selles"

3 des optimiseurs déterministes sont asymptotiquement instables aux voisinages de points selles

[2] traduction d'une phrase trouvée dans une publication de référence

Bigger is better (?)

2 résultats issus de la thèse de Bilel Bensaid.

- MNIST:

Réseau	DDL	Max	Médiane
$784 \times 24 \times 10$	19 k	96.3%	95.9%
$784 \times 300 \times 10$	239 k	95.3%	
$784 \times 1000 \times 10$	795 k	95.5%	
$784 \times 300 \times 100 \times 10$	267 k	96.9%	

- Fashion MNIST:

Réseau	DDL	Max	Médiane
LeNet-1	3.2 k	99.1%	98.8%
Fine-DARTS	3.2 M	96.9%	
WRN28-10	36 M	96.4%	
Resnet110	1.6 M	96.0%	
VGG8B	7.3 M	95.5%	

Ne pas valider correctement

- L'algorithme de Momentum tel qu'implémenté dans diverses bibliothèques de ML:

$$\begin{array}{rcl|rcl} v_{n+1} & = & (1-\beta_1)v_n + \beta_1 \nabla \mathcal{R}(\theta_n), & v_{n+1} & = & (1-\beta_1)v_n - \eta \nabla \mathcal{R}(\theta_n), \\ \theta_{n+1} & = & \theta_n - \eta v_{n+1} & \theta_{n+1} & = & \theta_n + v_{n+1} \\ \hline v_{n+1} & = & (1-\beta_1)v_n + \eta \nabla \mathcal{R}(\theta_n), & v_{n+1} & = & (1-\beta_1)v_{n+1} + \nabla \mathcal{R}(\theta_n), \\ \theta_{n+1} & = & \theta_n - \eta v_{n+1} & \theta_{n+1} & = & \theta_n - \eta v_{n+1} \end{array}$$

- Comparer les résultats sur des cas complexes n'est pas suffisant.

Ne pas surinterpréter

Que mesure-t-on au final?

Que dire des tableaux présentés il y a 2 transparents?

Conclusion et perspectives

- Des méthodes pour fiabiliser les optimiseurs.
- Des résultats théoriques garantissant les propriétés.
- Des résultats pratiques à la hauteur.
- On peut encore améliorer les résultats dans certains cas particulièrement raides.
- Il reste encore beaucoup de choses à comprendre: initialisations, architecture, etc.

Conclusion et perspectives

Merci de votre attention!

Bibliographie

1. A. Bourriaud, R. Loubère, R. Turpault, A Priori Neural Networks Versus A Posteriori MOOD Loop: A High Accurate 1D FV Scheme Testing Bed, Journal of Scientific Computing, 2020,
2. P. Novello, G. Poette, D. Lugato, S. Peluchon, P. Congedo, Goal oriented sensitivity analysis of hyperparamaters in Deep Learning, Journal of Scientific Computing, 2023,
3. P. Novello, Combining supervised deep learning and scientific computing : some contributions and application to computational fluid dynamics, 2022,
4. B. T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Computational Mathematics and Mathematical Physics, 1964,
5. D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, International Conference on Learning Representations, 2014,
6. B. Bensaid, G. Poette, R. Turpault, Deterministic Neural Networks Optimization from a Continuous and Energy Point of View, 2023,
7. B. Bensaid, Analyse et développement de nouveaux optimiseurs en Machine Learning, 2024,
8. R.L. Speth, W.H. Green, S. Macmanara, and G. Strang. Balanced Splitting AND Rebalanced Splitting. SIAM Journal on Numerical Analysis, 2013.