Eléments d'optimisation pour les réseaux de neurones

Julien Velcin
M2 MALIA - Université Lumière Lyon 2

Note préliminaire

- Nombre d'exemples et d'illustrations sont tirés des livres de Goodfellow et al. (2016) et de Jurafsky et Martin (2019). Les références complètes sont données en fin de présentation.

De nombreux facteurs

De nombreux facteurs influent sur la réussite du processus d'optimisation :

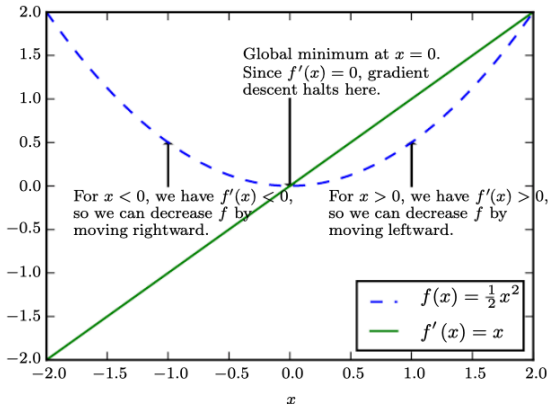
- initialisation (et contrôle) des poids
- calcul du gradient (batch, minibatch)
- pas d'apprentissage (fixe, adaptatif)
- au-delà du gradient : moments de 2nd ordre

Gradient stochastique

- L'approche par "mini-batch" garantit de suivre le gradient de l'erreur en généralisation… après une passe
- Effet de régularisation pour prévenir le sur-apprentissage
- Se parallélise (mais pas avec des mini-batchs trop petits)

Attention à bien penser à mélanger (une fois !) les exemples d'apprentissage

Descente du gradient



Descente simple (tirée de Goodfellow et al., 2016)

Observations sur la fonction d'activation

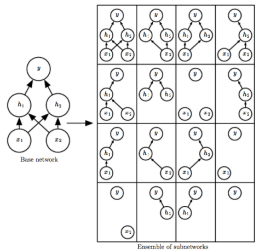
- Tendance à préférer des fonctions symétriques autour de l'origine (sigmoïde ou tanh) car elles fournissent une entrée centrée en 0 pour la couche suivante ; on a observé que tanh a de meilleurs propriétés de convergence. Cependant, la fonction **reLU** semble très employée ces derniers temps (bonnes propriétés héritées de la linéarité)
- Les poids initiaux doivent être petits et proches de 0 afin d'avoir des variations linéaires au démarrage
- pour tanh : $uniforme[-\frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}, \frac{\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}]$ où f_{in} et f_{out} sont le nombre de connexions entrantes et sortantes respectivement
- pour la sigmoïde : $uniforme[-\frac{4*\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}, \frac{4*\sqrt{6}}{\sqrt{fan_{in}+fan_{out}}}]$

Régularisation

- Comme pour les autres modèles d'apprentissage automatique, les réseaux de neurones sont menacés par le sur-apprentissage (*overfitting*).
- Une solution est de régulariser la fonction objectif : $f_{obj} = f_{err} + \lambda \,.\, \Omega(\theta)$, où $\lambda \in [0,\infty)$ est un hyper-paramètre à fixer.
- Des valeurs typiques à essayer pour λ sont 10^{-2}, 10^{-3}...
- Ω est souvent une norme, telle que la norme $L1$ (cf. LASSO), $L2$ (cf. *ridge regression*) ou les deux à la fois (cf. *elastic net*).
- Une autre manière de régulariser consiste à arrêter l'apprentissage à temps (*early stopping*).
- Le principe des mini-matchs est également une manière (indirecte) de régulariser

Dropout

- Le dropout est une forme de régularisation basée sur l'estimation d'un *ensemble* de réseaux calculés à partir du réseau initial :



Au-delà du gradient

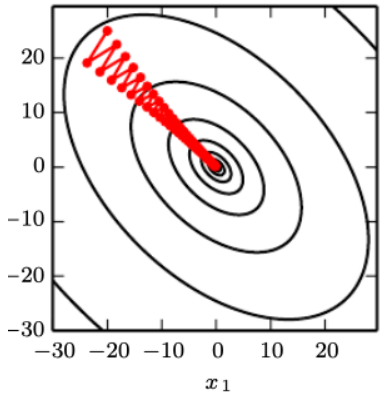
- Aller à l'opposée du gradient ne garantit pas toujours que l'on va minimiser la fonction :

$$f(x) \approx f(a) + \nabla f(a)(x-a) + \frac{1}{2}(x-a)^T Hf(a)(x-a) + \ldots$$

Gradient — Hessian

Expansion de Taylor

Limites du gradient

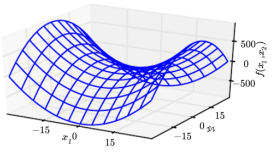


Paysages de recherche

- Espace des paramètres en (très) grande dimension

  L'étude des fonctions aléatoires montre que la probabilité que toutes les valeurs propres de H soient positives (minimum local) augmente dans les régions de faible coût (Dauphin et al., 2014)

-



Quelques heuristiques

- Décroissance du taux d'apprentissage
- Méthode du momentum
- AdaGrad
- RMSProp
- Adam

Méthode du momentum

- Basé sur la prise en compte d'une moyenne mobile (*moving average*) des gradients passés : $\theta_{t+1} = \theta_t + v$

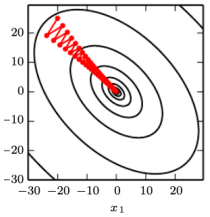
poids des gradients précédents

pas d'apprentissage

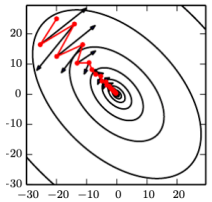
$$v = \alpha \,.\, v - \epsilon \,.\, \nabla_\theta$$

accumulation des gradients précédents

nouveau gradient



sans momentum



avec momentum

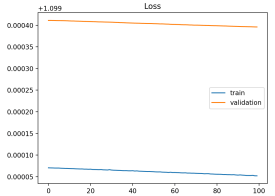
Approches adaptatives

- **AdaGrad** : adapte le pas d'apprentissage de manière inversement proportionnelle aux valeurs passées du gradient
- **RMSProp** : variante d'AdaGrad afin d'oublier les gradients passés (apparemment plus efficace en cas de forte non convexité)
- **Adam** : combinaison de RMSProp et du momentum avec quelques modifications

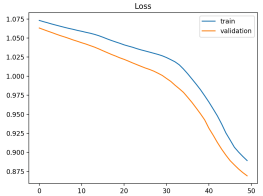
Autres heuristiques

- Méthodes de second ordre : Newton, gradients conjugués
- *Batch normalization* : reparamétrisation adaptative $H' = \frac{H - \mu}{\sigma}$
- Gradient clipping

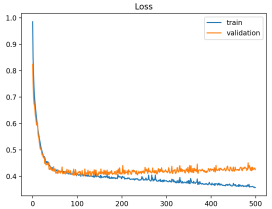
Surveiller les courbes (1)



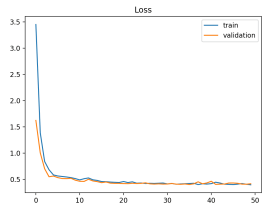
sous-apprentissage (1)



sous-apprentissage (2)

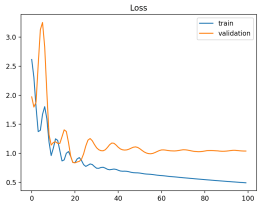


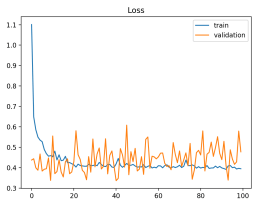
sur-apprentissage

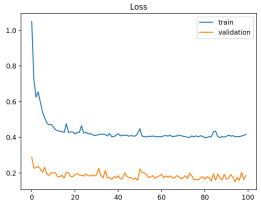


bon apprentissage

Surveiller les courbes (2)







données non représentatives

Références

- Deep Learning par Ian Goodfellow, Yoshua Bengio and Aaron Courville. MIT Press, 2016.
- Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft). October 16, 2019. https://web.stanford.edu/~jurafsky/slp3/
- Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convexoptimization. In NIPS'2014 .
- Deep Learning Tutorial, Release 0.1. LISA lab, University of Montreal, 2015. http://deeplearning.net/tutorial/deeplearning.pdf
- Blog J. Brownlee. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/