





# Pense-bête VIP : Petites astuces





Afshine AMIDI et Shervine AMIDI

6 janvier 2019

## Traitement des données

□ **Augmentation des données** – Les modèles d'apprentissage profond ont typiquement besoin de beaucoup de données afin d'être entraînés convenablement. Il est souvent utile de générer plus de données à partir de celles déjà existantes à l'aide de techniques d'augmentation de données. Celles les plus souvent utilisées sont résumées dans le tableau ci-dessous. À partir d'une image, voici les techniques que l'on peut utiliser :

Original	Symétrie axiale	Rotation	Recadrage aléat.
			
- Image sans aucune modif	- Symétrie par rapport à un axe pour lequel le sens de l'image est conservé	- Rotation avec un petit angle - Reproduit une calibration imparfaite de l'horizon	- Concentration aléatoire sur une partie de l'image - Plusieurs rognements aléatoires peuvent être faits à la suite

Ch. de couleur	Addition de bruit	Perte d'info.	Ch. de contraste
			
- Nuances de RGB sont légèrement changées - Capture le bruit qui peut survenir avec de l'exposition lumineuse	- Addition de bruit - Plus de tolérance envers la variation de la qualité de l'entrée	- Parties de l'image ignorées - Imite des pertes potentielles de parties de l'image	- Changement de luminosité - Contrôle la différence de l'exposition dû à l'heure de la journée

□ **Normalisation de lot** – La normalisation de lot (en anglais *batch normalization*) est une étape qui normalise le lot  $\{x_i\}$  avec un choix de paramètres  $\gamma, \beta$ . En notant  $\mu_B, \sigma_B^2$  la moyenne et la variance de ce que l'on veut corriger du lot, on a :

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Ceci est couramment fait après un fully connected/couche de convolution et avant une couche non-linéaire. Elle vise à permettre d'avoir de plus grands taux d'apprentissages et de réduire la dépendance à l'initialisation.

## Entraîner un réseau de neurones

□ **Epoch** – Dans le contexte de l'entraînement d'un modèle, l'*epoch* est un terme utilisé pour référer à une itération où le modèle voit tout le training set pour mettre à jour ses coefficients.

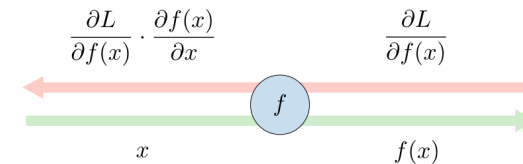
□ **Gradient descent sur mini-lots** – Durant la phase d'entraînement, la mise à jour des coefficients n'est souvent basée ni sur tout le training set d'un coup à cause de temps de calculs coûteux, ni sur un seul point à cause de bruits potentiels. À la place de cela, l'étape de mise à jour est faite sur des mini-lots, où le nombre de points dans un lot est un paramètre que l'on peut régler.

□ **Fonction de loss** – Pour pouvoir quantifier la performance d'un modèle donné, la fonction de loss (en anglais *loss function*)  $L$  est utilisée pour évaluer la mesure dans laquelle les sorties vraies  $y$  sont correctement prédites par les prédictions du modèle  $z$ .

□ **Entropie croisée** – Dans le contexte de la classification binaire d'un réseau de neurones, l'entropie croisée (en anglais *cross-entropy loss*)  $L(z, y)$  est couramment utilisée et est définie par :

$$L(z, y) = - \left[ y \log(z) + (1 - y) \log(1 - z) \right]$$

□ **Backpropagation** – La backpropagation est une méthode de mise à jour des coefficients d'un réseau de neurones en prenant en compte les sorties vraies et désirées. La dérivée par rapport à chaque coefficient  $w$  est calculée en utilisant la règle de la chaîne.



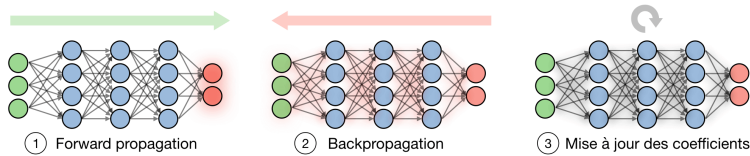
En utilisant cette méthode, chaque coefficient est mis à jour par :

$$w \leftarrow w - \alpha \frac{\partial L(z, y)}{\partial w}$$

□ **Mise à jour les coefficients** – Dans un réseau de neurones, les coefficients sont mis à jour par :

- Étape 1 : Prendre un lot de training data et effectuer une forward propagation pour calculer le loss.
- Étape 2 : Backpropager le loss pour obtenir le gradient du loss par rapport à chaque coefficient.

— Étape 3 : Utiliser les gradients pour mettre à jour les coefficients du réseau.



## Réglage des paramètres

□ **Initialisation de Xavier** – Au lieu de laisser les coefficients s’initialiser de manière purement aléatoire, l’initialisation de Xavier permet d’avoir des coefficients initiaux qui prennent en compte les caractéristiques uniques de l’architecture.

□ **Apprentissage par transfert** – Entraîner un modèle d’apprentissage profond requière beaucoup de données et beaucoup de temps. Il est souvent utile de profiter de coefficients pré-entraînés sur des données énormes qui ont pris des jours/semaines pour être entraînés, et profiter de cela pour notre cas. Selon la quantité de données que l’on a sous la main, voici différentes manières d’utiliser cette méthode :

Taille du training	Illustration	Explication
Petite		Gèle toutes les couches, entraîne les coefficients du softmax
Moyenne		Gèle la plupart des couches, entraîne les coefficients des dernières couches et du softmax
Grande		Entraîne les coefficients des couches et du softmax en initialisant les coefficients sur ceux qui ont été pré-entraînés

□ **Taux d’apprentissage** – Le taux d’apprentissage (en anglais *learning rate*), souvent noté  $\alpha$  ou  $\eta$ , indique la vitesse à laquelle les coefficients sont mis à jour. Il peut être fixe ou variable. La méthode actuelle la plus populaire est appelée Adam, qui est une méthode faisant varier le taux d’apprentissage.

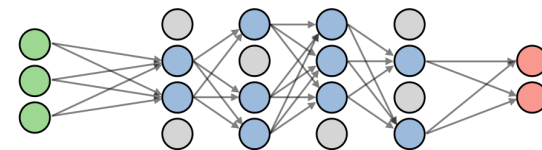
□ **Taux d’apprentissage adaptatifs** – Laisser le taux d’apprentissage varier pendant la phase d’entraînement du modèle peut réduire le temps d’entraînement et améliorer la qualité de la solution numérique optimale. Bien que la méthode d’Adam est la plus utilisée, d’autres peuvent aussi être utiles. Les différentes méthodes sont récapitulées dans le tableau ci-dessous :

Méthode	Explication	Mise à jour de $w$	Mise à jour de $b$
Momentum	<ul style="list-style-type: none"> <li>- Amortit les oscillations</li> <li>- Amélioration par rapport à la méthode SGD</li> <li>- 2 paramètres à régler</li> </ul>	$w - \alpha v_{dw}$	$b - \alpha v_{db}$
RMSprop	<ul style="list-style-type: none"> <li>- Root Mean Square propagation</li> <li>- Accélère l’algorithme d’apprentissage en contrôlant les oscillations</li> </ul>	$w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \leftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	<ul style="list-style-type: none"> <li>- Adaptive Moment estimation</li> <li>- Méthode la plus populaire</li> <li>- 4 paramètres à régler</li> </ul>	$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \leftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

*Remarque : parmi les autres méthodes existantes, on trouve Adadelta, Adagrad et SGD.*

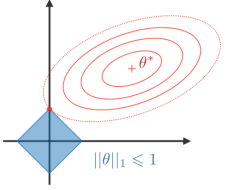
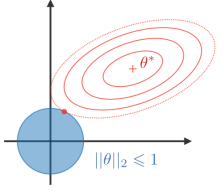
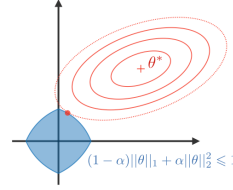
## Régularisation

□ **Dropout** – Le dropout est une technique qui est destinée à empêcher le sur-ajustement sur les données de training en abandonnant des unités dans un réseau de neurones avec une probabilité  $p > 0$ . Cela force le modèle à éviter de trop s’appuyer sur un ensemble particulier de features.



*Remarque : la plupart des frameworks d’apprentissage profond paramètrent le dropout à travers le paramètre ‘garder’  $1 - p$ .*

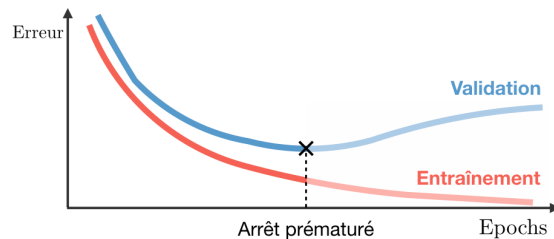
□ **Régularisation de coefficient** – Pour s’assurer que les coefficients ne sont pas trop grands et que le modèle ne sur-ajuste pas sur le training set, on utilise des techniques de régularisation sur les coefficients du modèle. Les techniques principales sont résumées dans le tableau suivant :

LASSO	Ridge	Elastic Net
<ul style="list-style-type: none"> <li>- Réduit les coefficients à 0</li> <li>- Bon pour la sélection de variables</li> </ul>	Rend les coefficients plus petits	Compromis entre la sélection de variables et la réduction de la taille des coefficients
		
$\dots + \lambda   \theta  _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda   \theta  _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda \left[ (1 - \alpha)   \theta  _1 + \alpha   \theta  _2^2 \right]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

	Gradient numérique	Gradient analytique
Formule	$\frac{df}{dx}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$	$\frac{df}{dx}(x) = f'(x)$
Commentaires	<ul style="list-style-type: none"> <li>- Coûteux ; le loss doit être calculé deux fois par dimension</li> <li>- Utilisé pour vérifier l'exactitude d'une implémentation analytique</li> <li>- Compromis dans le choix de h entre pas trop petit (instabilité numérique) et pas trop grand (estimation du gradient approximative)</li> </ul>	<ul style="list-style-type: none"> <li>- Résultat 'exact'</li> <li>- Calcul direct</li> <li>- Utilisé dans l'implémentation finale</li> </ul>

★ ★ ★

□ **Arrêt prématuré** – L'arrêt prématuré (en anglais *early stopping*) est une technique de régularisation qui consiste à stopper l'étape d'entraînement dès que le loss de validation atteint un plateau ou commence à augmenter.



## Bonnes pratiques

□ **Surapprentissage d'un mini-lot** – Lorsque l'on débogue un modèle, il est souvent utile de faire de petits tests pour voir s'il y a un gros souci avec l'architecture du modèle lui-même. En particulier, pour s'assurer que le modèle peut être entraîné correctement, un mini-lot est passé dans le réseau pour voir s'il peut sur-ajuster sur lui. Si le modèle ne peut pas le faire, cela signifie que le modèle est soit trop complexe ou pas assez complexe pour être sur-ajusté sur un mini-lot.

□ **Vérification de gradient** – La méthode de la vérification de gradient (en anglais *gradient checking*) est utilisée durant l'implémentation d'un backward pass d'un réseau de neurones. Elle compare la valeur du gradient analytique par rapport au gradient numérique au niveau de certains points et joue un rôle de vérification élémentaire.