

Pense-bête VIP : Apprentissage profond

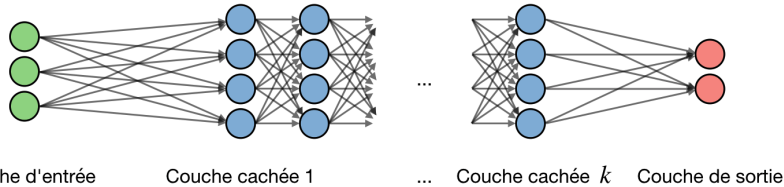
Afshine AMIDI et Shervine AMIDI

6 octobre 2018

Réseau de neurones

Les réseaux de neurones (en anglais *neural networks*) sont une classe de modèles qui sont construits à l'aide de couches de neurones. Les réseaux de neurones convolutionnels (en anglais *convolutional neural networks*) ainsi que les réseaux de neurones récurrents (en anglais *recurrent neural networks*) font parti des principaux types de réseaux de neurones.

□ **Architecture** – Le vocabulaire autour des architectures des réseaux de neurones est décrit dans la figure ci-dessous :



En notant i la $i^{\text{ème}}$ couche du réseau et j la $j^{\text{ème}}$ unité de couche cachée, on a :

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

où l'on note w , b , z le coefficient, le biais ainsi que la variable sortie respectivement.

□ **Fonction d'activation** – Les fonctions d'activation sont utilisées à la fin d'une unité de couche cachée pour introduire des complexités non linéaires au modèle. En voici les plus fréquentes :

Sigmoïde	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$

□ **Cross-entropy loss** – Dans le contexte des réseaux de neurones, la fonction objectif de cross-entropy $L(z, y)$ est communément utilisée et est définie de la manière suivante :

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

□ **Taux d'apprentissage** – Le taux d'apprentissage (appelé en anglais *learning rate*), souvent noté α ou parfois η , indique la vitesse à laquelle les coefficients évoluent. Cette quantité peut être fixe ou variable. L'une des méthodes les plus populaires à l'heure actuelle s'appelle Adam, qui a un taux d'apprentissage qui s'adapte au fil du temps.

□ **Rétropropagation du gradient** – La rétropropagation du gradient (en anglais *backpropagation*) est une méthode destinée à mettre à jour les coefficients d'un réseau de neurones en comparant la sortie obtenue et la sortie désirée. La dérivée par rapport au coefficient w est calculée à l'aide du théorème de dérivation des fonctions composées, et s'écrit de la manière suivante :

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Ainsi, le coefficient est actualisé de la manière suivante :

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

□ **Actualiser les coefficients** – Dans un réseau de neurones, les coefficients sont actualisés comme suit :

- Étape 1 : Prendre un groupe d'observations appartenant aux données du training set.
- Étape 2 : Réaliser la propagation avant pour obtenir le loss correspondant.
- Étape 3 : Effectuer une rétropropagation du loss pour obtenir les gradients.
- Étape 4 : Utiliser les gradients pour actualiser les coefficients du réseau.

□ **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. En pratique, les neurones sont soit abandonnés avec une probabilité p ou gardés avec une probabilité $1 - p$.

Réseaux de neurones convolutionnels

□ **Pré-requis de la couche convolutionnelle** – Si l'on note W la taille du volume d'entrée, F la taille de la couche de neurones convolutionnelle, P la quantité de zero padding, alors le nombre de neurones N qui tient dans un volume donné est tel que :

$$N = \frac{W - F + 2P}{S} + 1$$

□ **Normalisation de batch** – C'est une étape possédant les paramètres γ, β qui normalise le batch $\{x_i\}$. En notant μ_B, σ_B^2 la moyenne et la variance de ce que l'on veut corriger au batch, ceci est fait de la manière suivante :

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

Cela est normalement effectué après une couche fully-connected/couche convolutionnelle et avant une couche de non-linéarité et a pour but de permettre un taux d'apprentissage plus grand et de réduire une dépendance trop forte à l'initialisation.

Réseaux de neurones récurrents

□ **Types de porte** – Voici les différents types de porte que l'on rencontre dans un réseau de neurones récurrent typique :

Porte d'entrée	Porte d'oubli	Porte de sortie	Porte
Écrire ?	Supprimer ?	A quel point révéler ?	Combien écrire ?

□ **LSTM** – Un réseau de long court terme (en anglais *long sort-term memory*, LSTM) est un type de modèle RNN qui empêche le phénomène de *vanishing gradient* en ajoutant des portes d'oubli.

Reinforcement Learning

Le but du reinforcement learning est pour un agent d'apprendre comment évoluer dans un environnement.

□ **Processus de décision markovien** – Un processus de décision markovien (MDP) est décrite par 5 quantités $(S, \mathcal{A}, \{P_{sa}\}, \gamma, R)$, où :

- S est l'ensemble des états
- \mathcal{A} est l'ensemble des actions
- $\{P_{sa}\}$ sont les probabilités d'états de transition pour $s \in S$ et $a \in \mathcal{A}$
- $\gamma \in [0, 1[$ est le taux d'actualisation (en anglais *discount factor*)
- $R : S \times \mathcal{A} \rightarrow \mathbb{R}$ ou $R : S \rightarrow \mathbb{R}$ est la fonction de récompense que l'algorithme veut maximiser

□ **Politique** – Une politique π est une fonction $\pi : S \rightarrow \mathcal{A}$ qui lie les états aux actions.

Remarque : on dit que l'on effectue une politique donnée π si étant donné un état s , on prend l'action $a = \pi(s)$.

□ **Fonction de valeurs** – Pour une politique donnée π et un état donné s , on définit la fonction de valeurs V^π comme suit :

$$V^\pi(s) = E \left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi \right]$$

□ **Équation de Bellman** – Les équations de Bellman optimales caractérisent la fonction de valeurs V^{π^*} de la politique optimale π^* :

$$V^{\pi^*}(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in S} P_{sa}(s') V^{\pi^*}(s')$$

Remarque : on note que la politique optimale π^ pour un état donné s est tel que :*

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

□ **Algorithme d'itération sur la valeur** – L'algorithme d'itération sur la valeur est faite de deux étapes :

— On initialise la valeur :

$$V_0(s) = 0$$

— On itère la valeur en se basant sur les valeurs précédentes :

$$V_{i+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left[\sum_{s' \in S} \gamma P_{sa}(s') V_i(s') \right]$$

□ **Maximum de vraisemblance** – Les estimations du maximum de vraisemblance pour les transitions de probabilité d'état sont comme suit :

$$P_{sa}(s') = \frac{\text{\#fois où l'action } a \text{ dans l'état } s \text{ est prise pour arriver à l'état } s'}{\text{\#fois où l'action } a \text{ dans l'état } s \text{ est prise}}$$

□ **Q-learning** – Le Q-learning est une estimation non-paramétrique de Q , qui est faite de la manière suivante :

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$