Théorie de l'Apprentissage par Réseaux de Neurones

Ibrahim Merad encadré par Stéphane Gaïffas & Emmanuel Bacry

October 1, 2020

1 Introduction

Faisant partie des méthodes de machine learning ou apprentissage automatique, les réseaux de neurones sont des objets qui ont récemment reçu beaucoup d'attention de la part de la communauté scientifique [27,36,38]. Initialement introduits dans les années 60 [48], l'intérêt pour leur utilisation a été renouvelé plus récemment avec l'élaboration de protocoles permettant de les entraîner [39]. De plus, des outils matériels et logiciels ont été expressément développés afin de faciliter leur utilisation. Il s'agit des cartes graphiques (calcul parallèle) ainsi que de librairies informatiques [1,18,45](Torch, Tensorflow, Keras ...).

Le domaine de l'apprentissage par réseau de neurones est en évolution rapide, autant sur les aspects méthodiques et applicatifs que sur la théorie visant à expliquer leurs performances constatées sur une variété de tâches. L'une des principales applications étant la classification. Cette tâche est effectuée comme suit : étant donné un ensemble prédéfini de classes possibles, par exemple $\{1, 2 \ldots, N_{\mathcal{C}}\}$, un classifieur associe à toute entrée représentée par un vecteur $x \in \mathbb{R}^{d_x}$ le vecteur des probabilités $\mathbb{P}\left(class(x) = j\right)_{j=1,2\ldots,N_{\mathcal{C}}}$, par suite, la classe de plus grande probabilité est désignée comme la prédiction du classifieur. Un exemple très courant est la reconnaissance d'objets sur des images [36].

De nombreuses autres applications existent et il serait difficile de les énumérer exhaustivement mais, parmi les plus courantes, on peut mentionner :

- **Traitement du langage naturel** : reconnaissance vocale [5], analyse de sentiment [58], traduction [9], complétion ou résumé de texte [11] ...
- **Apprentissage par renforcement** : contrôle de robots [43], décision dans un marcher financier, recommandations publicitaires, jeu d'échec ou de Go [51] ...
- **Médical** : Segmentation de clichés et détection d'anomalies, suggestion de traitements, suivi de patients selon données d'historique [14, 50]...

Malgré leur succès dans plusieurs applications, l'explication des performances d'un réseau de neurones reste un défi. De plus, il a même été démontré qu'ils sont parfois loin d'être fiables, par exemple, une perturbation visuellement imperceptible d'une image introduite dans un classifieur suffirait à fausser sa prédiction [55].

Dans ce document, nous présenterons quelques travaux importants effectués dans le but d'améliorer notre compréhension des réseaux de neurones. Cette présentation ne se veut certainement pas exhaustive en raison de l'abondance de recherche pertinente et de l'espace limité.

2 Qu'est ce qu'un réseau de neurones ?

Dans cette section, nous présenterons rapidement comment un réseau de neurones est construit et une méthode générique pour l'entraîner.

2.1 Structure

On peut penser aux réseaux de neurones comme une famille de fonctions formées par la composition alternée d'applications linéaires (ou affines) et de fonctions non linéaires. Par exemple, soient d_x et d_y les dimensions de l'entrée et de la sortie respectivement. Un réseau $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ est paramétré par $\theta = (W_1, b_1, \ldots, W_L, b_L)$ avec $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ et $b_i \in \mathbb{R}^{d_i}$ pour $i = 1, \ldots, L$. Le nombre L est la *profondeur* du réseau et on a $d_0 = d_x$, $d_L = d_y$. Les entiers d_i représentent le nombre de *neurones* dans chaque couche i du réseau. On appelle *largeur* du réseau le nombre maximal de neurones dans une de ses couches.

Pour une entrée $x \in \mathbb{R}^{d_x}$, la sortie correspondante $f_\theta(x) = h_L$ est obtenue en définissant les activations des *couches* du réseau comme suit :

$$h_0 = x \quad , \quad h_i = \sigma\left(W_i h_{i-1} + b_i\right) \text{ pour } i = 1, \ldots, L-1 \text{ et } h_L = W_L h_{L-1} + b_L$$

où σ désigne une *fonction d'activation* non linéaire appliquée individuellement aux coordonnées d'un vecteur. Plusieurs choix sont possibles pour cette dernière, la plus courante est ReLU (Rectified Linear Unit [2]) qui s'exprime simplement comme $\sigma(x) = \max(0, x)$, d'autres choix populaires sont, par exemple, la fonction sigmoïde $\sigma(x) = \frac{1}{1+e^{-x}}$, la tangente hyperbolique tanh ou bien la fonction arctan.

Ainsi défini, f_θ représente un Multi Layer Perceptron (MLP). La Figure 1 illustre la structure d'un tel modèle. Il s'agit d'une définition basique et plusieurs variantes sont possibles.

2.2 Entraînement

Nous nous intéressons principalement à l'entraînement supervisé. Étant donné un jeu de données (*dataset*) constitué de n paires $\{(x_i, y_i)\}_{i=1}^n$ avec $(x_i, y_i) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ et une *fonction de perte* $\ell : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$, l'entraînement d'un réseau vise à trouver les paramètres θ minimisant l'objectif :

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \ell\left(f_\theta(x_i), y_i\right). \tag{1}$$

Dans le cas d'une tâche de classification, les labels y_i seront des vecteurs de la base canonique de $\mathbb{R}^{N_\mathcal{C}}$ avec $N_\mathcal{C}$ le nombre de classes. La coordonnée non nulle de y_i correspond à la classe de x_i et on utilise généralement l'entropie croisée discrète comme fonction de perte ℓ. En revanche, dans une tâche de régression, les y_i seront des vecteurs sans structure particulière et ℓ pourra être, par exemple, la perte quadratique i.e. $\ell(y, y') = \|y - y'\|_2^2$. L'objectif (1) représente le risque empirique qui est un substitut du risque moyen :

$$\widetilde{\mathcal{L}}(\theta) = \mathbb{E}_{(X,Y)} \ell\left(f_\theta(X), Y\right) \tag{2}$$

où l'espérance est prise par rapport à la paire (X, Y) suivant une distribution inconnue dont les données sont issues. En pratique, il n'est presque jamais possible d'optimiser le risque moyen (2), on utilise plutôt un échantillon de données d'entraînement $\{(x_i, y_i)\}_{i=1}^n$ pour définir le risque

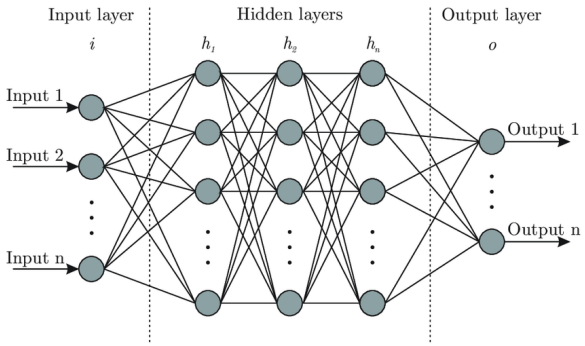

Figure 1: Illustration d'un MLP. Les neurones alignés verticalement représentent les coordonnées individuelles des vecteurs d'activation h_i. Les arêtes les reliant représentent les éléments des matrices W_i utilisées pour les calculer. Les biais b_i ne sont pas représentés ici.

empirique (1). Après avoir optimisé ce dernier, on évalue généralement le modèle entraîné sur un autre échantillon de *test* afin de vérifier que l'apprentissage *généralise* pour de nouvelles données. Une partie de la littérature étudie l'*erreur de généralisation* $\mathcal{L}(\theta)/n - \widetilde{\mathcal{L}}(\theta)$ dûe à l'utilisation d'un échantillon de données [33, 40] mais nous ne nous y intéresserons pas ici.

Vu la complexité des réseaux de neurones, l'objectif (1) vu comme fonction de θ n'est généralement pas convexe et son optimisation est effectuée grâce à des méthodes de premier ordre, typiquement une descente de gradient i.e. on calcule, à partir d'un θ_0 initial aléatoire une séquence définie par :

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{3}$$

avec $\eta > 0$ un pas de gradient. Encore une fois, il y a une littérature riche étudiant les divers algorithmes qu'il est possible d'utiliser pour cette optimisation. Il s'agit en général de variantes de la descente de gradient ayant une meilleure vitesse de convergence parfois avec une adaptation dynamique du pas η. La plus simple variante, appelée *stochastic gradient descent* (SGD) consiste à estimer, à chaque itération, le gradient $\nabla_\theta \mathcal{L}$ en utilisant seulement un sous-échantillon (*batch*) tiré aléatoirement dans le jeu de données. Cela a pour but de réduire le temps de calcul nécessaire pour un seul pas d'optimisation lorsque le jeu de données utilisé est de grande taille, ce qui est souvent le cas. Ce type d'entraînement permet également au réseau d'avoir de meilleures propriétés de généralisation [10]. L'utilisation d'un batch est parfois inévitable, par exemple dans le cas où l'objectif est défini par une intégrale sur une variable aléatoire de loi continue (c'est le cas, par exemple des autoencodeurs variationnels [34]). Dans ce cas, il est nécessaire de l'estimer avec une méthode de type Monte Carlo.

Remarquons que cette procédure d'entraînement apparemment simple a longtemps posé des difficultés numériques et algorithmiques. Par exemple, un choix complètement arbitraire du θ_0 initial peut provoquer une explosion exponentielle des activations dans le réseau ou lancer l'optimisation à partir d'un "plateau" de la fonction objectif sur lequel elle aura beaucoup de mal à progresser. De plus, le calcul du gradient pour une architecture arbitraire n'est pas évident, surtout lorsque le réseau est défini avec d'autres opérations que nous n'avons pas introduites ici. Cette fonction est remplie par l'algorithme Autograd qui permet d'obtenir efficacement le gradient pour une architecture donnée [39]. Ce dernier est implémenté dans les librairies informatiques de machine learning et la différentiation automatique se fait de manière transparente pour l'utilisateur, ce qui

a grandement contribué à la popularisation de ces méthodes d'apprentissage.

Avant d'aborder les propriétés des réseaux de neurones, mentionnons deux aspects d'architecture très populaires :

Réseaux convolutionnels : Pour les réseaux prenant des images en entrée, les opérations linéaires encodent généralement des convolutions, on parle alors de réseau convolutionnel. Leur avantage est d'exploiter l'invariance par translation pour les tâches de détection d'objet. De plus, cela permet d'exploiter l'information de voisinage dans une image pour utiliser des matrices "éparses" (à faible nombre de coefficients non nuls) et ainsi réduire drastiquement les coûts de calcul.

Réseaux résiduels : Un réseau est dit "résiduel" si certaines de ses couches sont calculées comme (par exemple) :

$$h_i = \sigma\left(W_i h_{i-1} + b_i + h_j\right) \qquad \text{avec } j < i-1$$

c'est à dire en utilisant également une couche antérieure à celle qui la précède immédiatement. Les réseaux définis de la sorte se sont révélés être plus performants [27]. Ils sont en fait plus faciles à entraîner car leurs connections résiduelles atténuent le problème du "vanishing gradient" dont souffrent les réseaux *profonds* i.e. possédant un grand nombre de couches. Il s'agit d'un phénomène faisant que le gradient est très petit, surtout pour les couches superficielles, empêchant l'optimisation de progresser.

Nous présentons quelques propriétés d'approximation avant d'aborder la question de convergence de l'entraînement.

3 Propriétés d'approximation universelle

Le résultat suivant, dénommé théorème d'approximation universel, implique que l'ensemble des fonctions définies par un réseau de neurones est dense dans l'espace des fonctions continues sur un compact pour la topologie uniforme.

Le théorème suivant, tiré de [46], étend les résultats classiques de [19, 29].

Théorème 1. *Soit $\sigma : \mathbb{R} \to \mathbb{R}$ une fonction d'activation continue non polynomiale et soient d_x, d_y des entiers. Pour toute fonction continue $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, tout ensemble compact $K \subset \mathbb{R}^{d_x}$ et tout $\epsilon > 0$ il existe un réseau de neurones f de profondeur $L = 2$ (une couche cachée) avec σ comme fonction d'activation, tel que :*

$$\sup_{x \in K} \|g(x) - f(x)\| \leq \epsilon.$$

Ainsi, les réseaux à une couche cachée sont déjà des approximateurs universels. Une approximation plus fine nécessite un réseau avec un plus grand nombre de neurones. Ce résultat s'étend facilement aux réseaux à plus de deux couches, c'est-à-dire les réseaux profonds. De plus, la profondeur d'un réseau lui donne un pouvoir d'approximation supérieur pour un moindre nombre de neurones. Ceci est exprimé par le théorème suivant, dû à [47], qui dit que les réseaux profonds sont exponentiellement plus efficaces que ceux à une seule couche cachée pour approcher des monômes.

Théorème 2. *Soit $p(x) : \mathbb{R}^n \to \mathbb{R}$ un monôme s'écrivant $p(x) = x_1^{r_1} \dots x_n^{r_n}$ et notons $d = \sum_{i=1}^n r_i$. On suppose que la fonction d'activation σ a des coefficients de Taylor non nuls jusqu'au degré $2d$. On note $m_k(p)$ le nombre minimal de neurones nécessaires à un réseau de profondeur k pour approcher p à ϵ-près pour la norme uniforme sur un compact et $m(p) = \min_{k \geq 1} m_k(p)$. Alors on a :*

- $m_1(p) = \prod_{i=1}^{n}(r_i + 1)$.
- $m(p) = \sum_{i=1}^{n}(7\lceil \log_2(r_i)\rceil + 4)$.

L'expressivité des réseaux profonds est donc nettement supérieure dans le sens où le nombre de neurones nécessaires est bien moindre.

Ces résultats sont intéressants, mais ils n'adressent qu'une partie du problème. On peut les voir comme des résultats d'existence. Étant donnée une fonction à approcher, pour un nombre suffisant de neurones, il existe un ensemble de paramètres tels que l'erreur soit inférieure à ϵ.

La seconde partie du problème est de pouvoir trouver ces paramètres. Comme mentionné précédemment, les paramètres d'un réseau sont entraînés par une descente de gradient d'une fonction objectif reflétant la performance du réseau à la tâche effectuée (voir équation (1)). Cette optimisation se fait en grande dimension et sur un objectif satisfaisant peu de propriétés utiles à l'optimisation (non convexe, continu mais pas $\mathcal{C}^1$ en général), par conséquent, elle risque de se retrouver bloquée dans des minima locaux, d'autres points critiques comme des points selles ou même sur des plateaux à faible gradient. Cependant, le constat empirique est que l'optimisation suit bien un chemin décroissant et atteint généralement un point critique de même valeur. Ceci indépendamment de l'aléa sur l'initialisation ou sur le gradient stochastique.

4 Connexité des sous niveaux de l'objectif

Une explication géométrique de la facilité d'entraînement des réseaux de neurones est possible. Cette piste a été initialement suggérée par un constat empirique de [25] et [21]. Après avoir effectué un entraînement à partir de deux initialisations différentes et ainsi obtenu deux optima, il est possible de les relier par un chemin dans l'espace des paramètres le long duquel la valeur de l'objectif est à peu près constante. Les articles précédents détaillent des méthodes permettant de trouver un tel chemin.

Ceci suggère que, pour une valeur c proche de celle de l'optimum global, les ensembles de sous-niveau de la fonction objectif

$$L_c^- = \{\theta \mid \mathcal{L}(\theta) \leq c\}$$

sont connexes. Ceci signifierait que les optima se trouvent au fond d'une "vallée" commune.

Ce fait fût montré pour les réseaux à une couche cachée par [24] qui a montré le théorème suivant :

Théorème 3. *Soit f_θ un réseau à une couche cachée de largeur m, qu'on entraîne avec l'objectif*

$$\mathcal{L}(\theta) = \mathbb{E}_{(X,Y)\sim P} \|f_\theta(X) - Y\|_2^2 + \kappa\mathcal{R}(\theta)$$

avec P une distribution, $\kappa > 0$ et $\mathcal{R}(\theta)$ une régularisation (ℓ_1 ou ℓ_2).

Soient $\theta_1, \theta_2 \in \Theta$ des paramètres et $\lambda \in \mathbb{R}$ tels que $\mathcal{L}(\theta_{\{1,2\}}) \leq \lambda$. Alors il existe un chemin $\gamma : [0,1] \to \Theta$ tel que $\gamma(0) = \theta_1$, $\gamma(1) = \theta_2$ et pour tout $t \in [0,1]$

$$\mathcal{L}(\gamma(t)) \leq \max(\lambda, \epsilon)$$

avec $\epsilon = O(m^{-\frac{1}{n}})$ où n est la dimension de X.

Remarquons que ce résultat implique une régularisation de l'objectif qui n'est pas forcément utilisée dans le cas général. De plus, on voit que le résultat s'améliore lorsque la largeur du réseau est grande (surparamétrisation).

Des travaux plus récents [41], ont établi cette propriété pour des réseaux profonds et pour différents objectifs d'entraînement. Ces derniers font des hypothèses sur l'activation utilisée et sur la largeur des couches relativement à la taille du jeu de données.

5 Convergence de l'entraînement d'un réseau simple suivant un réseau de référence

Le problème de montrer que l'entraînement d'un réseau aboutit à un minimum global est un problème mal posé. En effet, l'existence d'un tel minimum est déjà difficile à établir. De plus, vue la construction d'un réseau de neurones donnée section 2 et en supposant que l'activation ReLU est utilisée, il existe une infinité de paramètres définissant la même fonction f_θ[1]. Enfin, l'objectif à optimiser est souvent défini à partir d'un échantillon de données sur lequel il est nécessaire de poser des hypothèses, une question de généralisation se pose également.

Une étude de l'entraînement d'un réseau à une seule couche cachée a été réalisée par [23]. Ce dernier considère des entrées $x \in \mathbb{R}^d$ issues d'une loi gaussienne dont on extrait k patchs de taille p notés $Z = Z(x) \in \mathbb{R}^{p \times k}$. On suppose que l'intersection entre les patchs est vide. Le modèle est défini par :

$$f_\theta(Z) = \sum_{i=1}^{k} a_i \sigma\left(w^T Z_i\right)$$

ayant pour paramètre $\theta = (a, w)$ et utilisant ReLU comme activation σ. Remarquons qu'ici, le paramètre w est partagé par les patchs, il s'agit d'un réseau convolutionnel.

Afin de se placer dans un cadre où le minimum global est identifiable, on suppose que les labels sont obtenus à partir d'un réseau "teacher" de même architecture f_{θ^*} paramétré par un θ^* fixé qu'on souhaite retrouver. L'objectif entraîné est donc le suivant :

$$\min_\theta \mathcal{L}(\theta) = \mathbb{E}_Z\left[\left(f_\theta(Z) - f_{\theta^*}(Z)\right)^2\right]$$

Dans ce cadre, [23] conclut que l'optimisation retrouve le paramètre θ^* avec probabilité constante. Par ailleurs, sous certaines conditions sur θ^*, il existe une initialisation menant l'optimisation à un minimum local.

6 Neural Tangent Kernel

Une découverte importante fût celle du noyau neuronal tangent ou Neural Tangent Kernel (NTK) par [31]. Ce dernier a montré que pour un réseau de neurones dont la largeur des couches tant vers l'infini le noyau tangent associé converge vers un noyau asymptotique constant :

Soit $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ un réseau de neurones paramétré par $\theta \in \mathbb{R}^P$, le noyau tangent $K : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_y \times d_y}$ s'exprime comme le produit extérieur suivant :

[1] en introduisant des facteurs multiplicatifs sur les paramètres et en exploitant l'homogénéité de ReLU

$$K_\theta(x, x') = \sum_{p=1}^{P} \partial_{\theta_p} f_\theta(x) \otimes \partial_{\theta_p} f_\theta(x'). \tag{4}$$

De plus, la descente de gradient effectuée pendant l'entraînement peut être décrite comme une "kernel gradient descent" par rapport au noyau précédent, pour le voir, il faut exprimer la dynamique suivie par $\nabla_y \ell(f_\theta(x), y^*)$ pendant la descente de gradient avec $\ell(y, y^*)$ la fonction de perte (voir notations section 2). Illustrons d'abord ceci sur un problème de régression linéaire. Soit $x_1, \dots, x_n \in \mathbb{R}^d$ un jeu de données et soit $y^* \in \mathbb{R}^n$ le vecteur des labels associés, on note $X = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^T \in \mathbb{R}^{n \times d}$ et on optimise un paramètre $\theta \in \mathbb{R}^d$ pour l'objectif suivant :

$$\min_\theta \frac{1}{2} \|X\theta - y^*\|_2^2 = \min_\theta \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i^*)$$

où $f_\theta(x) = x^T\theta$ et $\ell(y, y^*) = \frac{1}{2}(y - y^*)^2$. Dans ce cas, la descente de gradient fait que $\theta^{(t)}$ suit la dynamique suivante (en temps continu)

$$\partial_t \theta^{(t)} = -X^T X \theta^{(t)} + X^T y^*.$$

Ainsi, en définissant le résidu $r^{(t)} = X\theta^{(t)} - y^* = (\nabla_y \ell(f_{\theta^{(t)}}(x_i), y_i^*))_{i=1}^n$, ce dernier suit la dynamique suivante :

$$\partial_t r^{(t)} = -XX^T r^{(t)},$$

ce qui correspond à une *kernel gradient descent* suivant le noyau positif $K = XX^T$. Dans le cas où ce noyau est, de plus, défini positif (données non dégénérées), le problème est convexe et l'optimisation atteint un minimum global. L'optimisation d'un réseau de neurones peut être vue d'un angle similaire à l'exception du fait que le noyau engendré par un réseau est plus compliqué à exprimer.

Dans un cadre plus général, soit $x_1, \dots, x_n \in \mathbb{R}^{d_x}$ un jeu de données et soient $y_1^*, \dots, y_n^* \in \mathbb{R}^{d_y}$ des labels associés. Soit aussi f_θ un réseau de neurones paramétré par $\theta \in \mathbb{R}^d$, on optimise θ pour minimiser un objectif :

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i^*).$$

Dans ce cas, on a la dynamique suivante pour la sortie du réseau pour une entrée x :

$$\partial_t f_{\theta^{(t)}}(x) = -\sum_{i=1}^{n} K_{\theta^{(t)}}(x, x_i) \partial_y \ell(f_{\theta^{(t)}}(x_i), y_i^*)$$

où apparaît le noyau $K_\theta(x, x')$ analogue à l'équation (4). La subtilité ici est que ce noyau dépend du paramètre θ et risque de changer avec les pas de gradient. Néanmoins dans le cas d'un DNN[2] et dans la limite où les couches deviennent infiniment larges, [31] montre que ce noyau converge vers un noyau constant K_c. Sous certaines hypothèses additionnelles sur les fonctions d'activations utilisées (non polynomiales) et sur les données (distinctes sur la sphère unité), ce noyau sera défini positif rendant le problème d'optimisation convexe. De plus, le noyau limite peut être calculé par une formule analytique récursive.

[2]Deep Neural Network : réseau de neurones profond

Ces résultat furent étendus par [56] et [42] avec des hypothèses allégées et une formule est aussi disponible pour le cas des réseaux convolutionnels (CNTK).

Ces noyaux ont effectivement été utilisés pour entraîner des modèles et obtenir de bonnes performances sur des tâches de classification [7].

7 Convergence de l'entraînement de réseaux ultra larges

Relativement récemment, des résultats de convergence ont été prouvés pour l'entraînement des réseaux profonds sous condition qu'ils aient un nombre suffisant de neurones [3, 22, 59]. Nous considérons de près le travail de [3] vu que ce fût le premier à fournir une preuve avec des taux polynomiaux.

On considère un réseau de neurones défini par des matrices $A \in \mathbb{R}^{d_x \times m}$, $B \in \mathbb{R}^{m \times d}$ et $W_1, \ldots, W_L \in \mathbb{R}^{m \times m}$. la fonction associée f_θ est définie pour $x \in \mathbb{R}^{d_x}$ par :

$$\begin{array}{ll} g_0 = Ax & h_0 = \phi(g_0) \\ g_l = W_l h_{l-1} & h_l = \phi(g_l) \quad \text{pour } l = 1, \ldots, L \\ y = f_\theta(x) = Bh_L & \end{array}$$

où ϕ désigne ReLU. Toutes les matrices précédentes sont initialisées avec des entrées gaussiennes iid. On suppose disposer d'un échantillon $\{x_i\}_{i=1}^n$ avec $x_i \in \mathbb{R}^{d_x}$ ainsi que de labels correspondants $\{y_i^*\}_{i=1}^n$ avec $y_i^* \in \mathbb{R}^d$. On note $y_i = f_\theta(x_i)$ et on optimise la fonction objectif suivante :

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \|y_i - y_i^*\|_2^2 \tag{5}$$

On fait l'hypothèse que le jeu de données soit δ-séparé pour un $\delta > 0$, i.e. :

$$\|x_i - x_j\|_2 \geq \delta \quad \text{pour } i \neq j$$

Le théorème suivant énonce le résultat de convergence pour un réseau suffisamment large (m grand).

Théorème 4. *On suppose que $m \geq \Omega\left(\text{poly}(n, L, \delta^{-1}) \cdot d\right)$ (sur-paramétrisation polynomiale). Soit une précision $\epsilon > 0$, la descente de gradient définie 3 appliquée à l'objectif équation (5) avec pas de descente $\eta = \Theta\left(\frac{d\delta}{\text{poly}(n,L)\cdot m}\right)$ atteint une valeur*

$$\mathcal{L}(\theta) \leq \epsilon$$

après un nombre de pas $T = \Theta\left(\frac{\text{poly}(n,L)}{\delta^2} \cdot \log \epsilon^{-1}\right)$.

Un résultat équivalent tient également avec un gradient stochastique (SGD). Donnons un bref aperçu de la preuve :

- Les premiers résultats établis concernent la stabilité des activations sur les couches. En exploitant l'aléa sur l'initialisation gaussienne des paramètres du réseau et l'utilisation de ReLU, on montre que les activations $(h_l)_{l=1,\ldots,L}$ sont d'ordre constant i.e. $\|h_l\| = \Theta(1)$.
- Une analyse de stabilité est faite par la suite, on montre qu'une perturbation θ' d'ordre contrôlé du paramètre initial n'engendre que des perturbations minimes sur les activations h_l ainsi que sur le calcul du gradient.

- Ensuite, on montre que la norme du gradient est à la fois minorée et majorée en fonction de l'objectif, ce qui implique qu'un point critique n'est pas atteint avant que l'objectif soit bien optimisé. Ceci est exprimé par les inégalités :

$$\|\nabla_\theta \mathcal{L}(\theta)\|_F^2 \leq O\left(\mathcal{L}(\theta) \times \frac{Lnm}{d}\right) \quad \text{et} \quad \|\nabla_\theta \mathcal{L}(\theta)\|_F^2 \geq \Omega\left(\mathcal{L}(\theta) \times \frac{\delta m}{dn^2}\right) \tag{6}$$

- Par la suite, une propriété de "semi-smoothness" est montrée pour l'objectif. Il s'agit de l'inégalité suivante pour un θ proche du paramètre initial $\breve{\theta}$ et une petite perturbation θ' :

$$\mathcal{L}(\theta+\theta') \leq \mathcal{L}(\theta) + \langle \nabla_\theta \mathcal{L}(\theta), \theta' \rangle + \frac{\text{poly}(L)\sqrt{nm\log m}}{\sqrt{d}} \|\theta'\| \sqrt{\mathcal{L}(\theta)} + O\left(\frac{nL^2 m}{d}\right) \|\theta'\|_2^2 \tag{7}$$

- À l'aide des deux éléments précédents, on montre une convergence linéaire de l'objectif durant la descente de gradient. Le résultat peut également être établi pour SGD.

Le résultat de convergence s'étend également au cas des réseaux résiduels et convolutionnels. Sous certaines hypothèses, des résultats plus faibles peuvent être obtenus pour d'autres fonctions de perte (atteinte d'un point critique).

Il faut garder à l'esprit que le degré polynomial de sur-paramétrisation caché par les notations est très élevé et irréaliste pour un réseau utilisé en pratique. En effet, largeur m du réseau doit satisfaire $m \geq \Omega\left(\frac{n^{24}L^{12}}{\delta^8}\right)$. Certains travaux ont tenté de ramener ces hypothèses à des taux plus réalistes [13, 32, 60].

Des travaux concurrents [22] ont obtenu un résultat similaire pour des taux de sur-paramétrisation plus faibles sur la taille du jeu de données n et des hypothèses différentes sur l'activation. Ces taux sont cependant exponentiels en la profondeur du réseau, sauf pour les réseaux résiduels.

Critique "Lazy Training". Malgré que les résultats présentés dans les deux sections précédentes semblent apporter des explications claires pour la convergence de l'entraînement des réseaux de neurones. Il faut garder à l'esprit qu'ils les considèrent dans des régimes qu'on ne retrouve pas forcément dans la pratique. Les taux de sur-paramétrisation supposés par [3] sont irréalistes et les résultats de [31] concernent des réseaux dont la largeur tend vers l'infini.

Une analyse critique de ces travaux fût faite par [17] qui les qualifie satiriquement de "lazy training". Dans le régime de sur-paramétrisation considéré, les paramètres du modèle changent peu ou pas du tout pendant la phase d'entraînement, de sorte qu'il se comporte comme sa linéarisation et une approximation de Taylor au voisinage de l'initialisation suffit pour l'analyser.

Ceci ne reflète pas ce qui est observé en pratique, de plus, [17] a expérimentalement montré que les modèles dans ce régime ont de moins bonnes propriétés de généralisation.

8 L'Approche par Champ Moyen

Une dernière approche que nous mentionnons est celle qui passe par la théorie du champ moyen (Mean field). Prenons l'exemple d'un réseau à une couche cachée $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}$ qui s'écrirait :

$$f_\theta(x) = \frac{1}{N}\sum_{i=1}^{N} a_i \sigma\left(w_i \cdot x\right),$$

avec $\theta = (a_1, w_1, \ldots, a_N, w_N)$. Remarquons l'introduction du facteur de normalisation $\frac{1}{N}$ spécifique à cette approche, on réécrit ce modèle sous la forme suivante :

$$f_\theta(x) = \langle a\sigma\left(w \cdot x\right), \delta_\theta \rangle = \int a\sigma\left(w \cdot x\right) \delta_\theta(da, dw),$$

où l'on a définit la mesure suivante sur l'espace des paramètres $\delta_\theta = \frac{1}{N} \sum_{i=1}^{N} \delta_{(a_i, w_i)}(da, dw)$ en notant $\delta_{(a,w)}$ la mesure de Dirac en (a, w) pour $a \in \mathbb{R}$ et $w \in \mathbb{R}^{d_x}$.

Pour une régression avec l'erreur quadratique, le but est de montrer qu'à la limite de largeur infinie $N \to \infty$ la solution $(\delta_{\theta_t})_{t\in[0,T]}$ de l'équation différentielle définie par l'entraînement des paramètres θ pendant une durée T converge faiblement (ou en loi) vers une solution déterministe $(\mu_t)_{t\in[0,T]}$.

Un tel résultat est présenté en détail pour les réseaux à une couche cachée par [52]. La même approche peut être appliquée pour des réseaux de profondeur arbitraire en prenant la limite de largeur infinie pour chaque couche récursivement (voir [53]). Dans le cas de plusieurs couches, la limite précédente de largeur infinie est prise itérativement sur chaque couche pour obtenir la convergence vers une mesure sur le sous espace correspondant de paramètres. De plus, sous certaines hypothèses, on peut montrer que l'objectif tend vers 0 lorsque la durée d'entraînement est arbitrairement longue $t \to \infty$.

Dans ce cadre, l'entraînement est gouverné par un *flot de Wasserstein* appliqué à une mesure sur l'espace des paramètres i.e. cette dernière évolue suivant l'action du flot du gradient de l'objectif optimisé. Ce phénomène est présenté et étudié en détail pour un réseau à une couche cachée entraîné à la classification binaire par [15].

9 Conclusion

Nous avons d'abord vu que les réseaux de neurones ont un pouvoir expressif qui en fait des approximateurs universels. Leur expressivité augmente d'autant plus rapidement par rapport à leur largeur lorsqu'ils sont dotés d'une architecture multi-couches.

Nous avons vu rapidement les plus importants travaux récents visant à expliquer, par de multiples approches, le succès des méthodes d'optimisation de premier ordre pour leur entraînement. Certains procèdent à une étude du paysage de l'objectif optimisé, d'autres tentent directement de montrer que l'optimisation converge vers un minimum global. Cependant, les deux ont le point commun de considérer des réseaux de grande largeur (grand nombre de neurones) souvent même jusqu'à l'infini. Un objectif futur pourrait être de tenter d'étendre les résultats actuels à des régimes intermédiaires de sur-paramétrisation plus proches de la pratique. Par exemple, une caractérisation plus fine de la convergence qui se produit à la limite de largeur infinie permettrait de déterminer plus précisément quand le comportement d'un modèle devient comparable à cette limite.

Dans les travaux que nous avons mentionnés, on étudie généralement l'entraînement sur un jeu de données arbitraire sans définir de modèle dessus. Une formalisation plus élaborée sur cet aspect serait nécessaire pour obtenir des garanties sur la généralisation au delà de la convergence de l'entraînement.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] A. F. Agarap. Deep Learning using Rectified Linear Units (ReLU). *arXiv e-prints*, page arXiv:1803.08375, Mar. 2018.

[3] Z. Allen-Zhu, Y. Li, and Z. Song. A Convergence Theory for Deep Learning via Over-Parameterization. *arXiv:1811.03962 [cs, math, stat]*, June 2019.

[4] Z. Allen-Zhu, Y. Li, and Z. Song. On the convergence rate of training recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 6673–6685, 2019.

[5] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. H. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015.

[6] S. Arora, N. Cohen, N. Golowich, and W. Hu. A convergence analysis of gradient descent for deep linear neural networks. *CoRR*, abs/1810.02281, 2018.

[7] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. *CoRR*, abs/1904.11955, 2019.

[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *CoRR*, abs/1708.05866, 2017.

[9] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[10] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *arXiv e-prints*, page arXiv:1606.04838, June 2016.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

[12] Y. Cao and Q. Gu. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *CoRR*, abs/1902.01384, 2019.

[13] Z. Chen, Y. Cao, D. Zou, and Q. Gu. How much over-parameterization is sufficient to learn deep relu networks?, 2019.

[14] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. Xu, E. M. Cofer, C. A. Lavender, S. C. Turaga, A. M. Alexandari, Z. Lu, D. J. Harris, D. DeCaprio, Y. Qi, A. Kundaje, Y. Peng, L. K. Wiley, M. H. Segler, S. M. Boca, S. J. Swamidass, A. Huang, A. Gitter, and C. S. Greene. Opportunities and obstacles for deep learning in biology and medicine. *bioRxiv*, 2018.

[15] L. Chizat and F. Bach. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. *arXiv e-prints*, page arXiv:2002.04486, Feb. 2020.

[16] L. Chizat and F. Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss, 2020.

[17] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming, 2018.

[18] F. Chollet et al. Keras, 2015.

[19] G. Cybenkot. Approximation by superpositions of a sigmoidal function *. 2006.

[20] Y. N. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, abs/1406.2572, 2014.

[21] F. Draxler, K. Veschgini, M. Salmhofer, and F. A. Hamprecht. Essentially no barriers in neural network energy landscape, 2018.

[22] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. *CoRR*, abs/1811.03804, 2018.

[23] S. S. Du, J. D. Lee, Y. Tian, B. Póczos, and A. Singh. Gradient descent learns one-hidden-layer CNN: don't be afraid of spurious local minima. *CoRR*, abs/1712.00779, 2017.

[24] C. D. Freeman and J. Bruna. Topology and geometry of half-rectified network optimization, 2016.

[25] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018.

[26] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems, 2014.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[29] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[30] L. Hui and M. Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, 2020.

[31] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2018.

[32] Z. Ji and M. Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks, 2019.

[33] K. Kawaguchi, L. Pack Kaelbling, and Y. Bengio. Generalization in Deep Learning. *arXiv e-prints*, page arXiv:1710.05468, Oct. 2017.

[34] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, Dec. 2013.

[35] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[37] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent, 2019.

[38] T. P. Lillicrap, J. J. Hunt, A. e. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, page arXiv:1509.02971, Sept. 2015.

[39] D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, 2015.

[40] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. *CoRR*, abs/1706.08947, 2017.

[41] Q. Nguyen. On connected sublevel sets in deep learning. *CoRR*, abs/1901.07417, 2019.

[42] R. Novak, L. Xiao, J. Lee, Y. Bahri, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes, 2018.

[43] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand, 2019.

[44] S. Oymak and M. Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *CoRR*, abs/1902.04674, 2019.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[46] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[47] D. Rolnick and M. Tegmark. The power of deeper networks for expressing natural functions. *CoRR*, abs/1705.05502, 2017.

[48] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[49] O. Shamir. Exponential convergence time of gradient descent for one-dimensional deep linear neural networks. *CoRR*, abs/1809.08587, 2018.

[50] B. Shickel, P. Tighe, A. Bihorac, and P. Rashidi. Deep EHR: A survey of recent advances on deep learning techniques for electronic health record (EHR) analysis. *CoRR*, abs/1706.03446, 2017.

[51] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[52] J. Sirignano and K. Spiliopoulos. Mean field analysis of neural networks: A law of large numbers, 2018.

[53] J. Sirignano and K. Spiliopoulos. Mean field analysis of deep neural networks, 2019.

[54] R. Sun. Optimization for deep learning: theory and algorithms, 2019.

[55] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2014.

[56] G. Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *CoRR*, abs/1902.04760, 2019.

[57] G. Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation, 2019.

[58] L. Zhang, S. Wang, and B. Liu. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883, 2018.

[59] D. Zou, Y. Cao, D. Zhou, and Q. Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *CoRR*, abs/1811.08888, 2018.

[60] D. Zou and Q. Gu. An improved analysis of training over-parameterized deep neural networks. *CoRR*, abs/1906.04688, 2019.

[61] D. Zou and Q. Gu. An improved analysis of training over-parameterized deep neural networks, 2019.