Brève introduction aux réseaux de neurones artificiels

Culture Sciences de l'Ingénieur

Auteur: Xihe Ge
avec la relecture de Laurent Oudre

Publié en :
Juillet 2022

Xihe Ge est un étudiant en génie mécanique dans la discipline "Industrie du Futur et Système Intelligent" à l'ENS Paris-Saclay. Laurent Oudre est professeur d'université à l'ENS Paris-Saclay dans le laboratoire Centre Borelli (UMR 9010); ses activités de recherche portent sur le traitement du signal, la reconnaissance des formes et l'apprentissage automatique pour les séries temporelles.

1 Introduction

Ces dernières années, avec les développements des techniques de traitement des données et l'afflux d'investissement, l'intelligence artificielle (IA) a affirmé son efficacité à automatiser les taches. Dès lors, son application s'est élargie à tout type de domaine : la recherche, l'industrie et la vie publique.
En 2011, *Watson* d'IBM, un éminent système informatique de réponse aux questions, a battu les deux concurrents humains les plus performants de l'émission populaire américaine *Jeopardy*!, ce qui a fait discuter les gens sur "la capacité potentielle de réflexion des machines". En 2016, après que le champion du monde de go Lee Sedol a été battu (1:4) par le programme de go *AlphaGo* de Google, les termes "intelligence artificielle (IA)", "apprentissage automatique (AA)" et "réseaux de neurones artificiels (RNA)" attirent à nouveau l'attention des publics et du vocabulaire des médias. Un an plus tard, le programme de nouvelle génération, *AlphaGo Master*, remporte le match par 3:0 contre Ke Jie, le joueur humain le mieux classé au monde, ce qui ouvre une nouvelle période de jeux compétitives que l'IA domine.
Cet article présente tout d'abord les définitions, les applications et les méthodes largement utilisées de l'IA afin de permettre une compréhension globale et intuitive. En outre, il étudiera comment les neurones du cerveau humain ont inspiré l'origine des réseaux neuronaux artificiels. Ensuite, il fournira une introduction générale et un résumé des principales techniques connexes, notamment le cadre, l'entraînement du modèle et l'optimisation.

2 Qu'est-ce que l'intelligence artificielle (IA) ?

2.1 La définition de l'IA

Le concept "intelligence artificielle" a été inventé par l'informaticien américain John McCarthy et trois autres chercheurs en 1956 [1], défini pour des machines qui peuvent être améliorées pour assumer certaines capacités considérées comme similaires à l'intelligence humaine, telles que l'apprentissage, l'adaptation et l'autocorrection [2]. Autrement dit, un système de théories, de méthodes, de techniques et d'applications peut imiter et étendre le comportement humain, comme la perception de l'environnement, l'acquisition de connaissances et l'obtention de résultats optimaux.
L'IA a des significations et des impressions différentes du point de vue du public et des universitaires. Le public et les médias montrent une grande passion pour l'intelligence générale artificielle, comme C-3PO dans le film *Star Wars* ou T700 dans *Terminator*. Cette machine omnipotente possède toutes sortes de perceptions et de rationalité, peut penser de manière abstraite, comprendre des idées complexes, planifier et résoudre des problèmes aussi rapidement que les êtres humains. Alors qu'à l'heure actuelle, nous en sommes encore à la phase de recherche de l'intelligence artificielle computationelle, c'est-à-dire la technologie capable d'accomplir une tâche spécifique aussi bien ou mieux que l'homme tel que la reconnaissance d'image.

2.2 L'application de l'IA

L'IA a déjà des performances exceptionnelles dans certains domaines, ce qui a donné lieu à une série d'applications permettant d'accomplir des tâches que les méthodes traditionnelles ne peuvent pas résoudre. Quelques applications représentatives dans les domaines de la santé, du langage et de la finance sont présentées ci-dessous :

- Santé

  Aujourd'hui, les smartwatches peuvent surveiller nos données de santé telles que l'électrocardiographie, le taux d'oxygène dans le sang et l'état du sommeil. L'IA peut ensuite analyser de manière exhaustive notre état de santé et fournir des conseils de santé personnalisés. Les montres peuvent également surveiller la stabilité de marche et ainsi prédire si les personnes âgées risquent de tomber et prévenir les accidents.

  Dans le diagnostic du cancer, l'IA est capable d'étudier un nombre considérable de données de tomographie assistée par ordinateur et de prédire avec une grande précision les cancers, comparable à celle d'experts dans ce domaine. Ainsi un excellent modèle d'IA, une fois validé, peut être appliqué à tous les patients du monde entier à défaut d'un médecin expérimenté proche.

- Langue

  L'IA fait également une grande différence dans le domaine des langues. Grâce aux logiciels de traduction actuels, tous les textes peuvent être lus par tous quelle que soit la langue d'origine. Les communications entre les personnes en sont également facilitées dans le monde professionnel lors de collaborations ou dans le monde personnel lors de déplacements.

  L'assistant vocal est courant dans la vie quotidienne. Il peut répondre à toutes sortes de questions et nous aider à contrôler divers produits intelligents en les appelant simplement facilitant ainsi notre quotidien.

- Finance

  L'IA peut aider les institutions financières à analyser les enregistrements financiers (débit ou crédit) d'emprunteurs, afin d'établir des modèles de comportement pour les personnes devant emprunter. Les vérifications des données, de potentielles erreurs ou fraudes sont ainsi automatiquement effectuées, réduisant les délais et les réponses erronées.

D'autres domaines, tels que l'industrie de l'Internet, la sécurité publique, le service à la clientèle, l'éducation, la culture, le tourisme, le jeu, la logistique, les nouvelles énergies, la pharmacie, la fabrication et la construction connaissent également des changements spectaculaires grâce à la transformation digitale et à l'IA.

2.3 La méthode de l'IA : l'apprentissage automatique

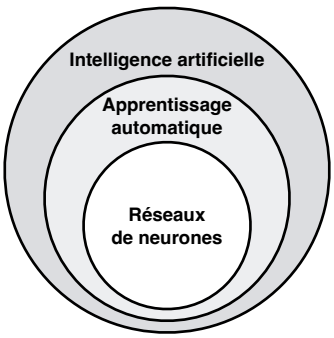

Figure 1. *"RN" ⊆ "AA" ⊆ "IA"*

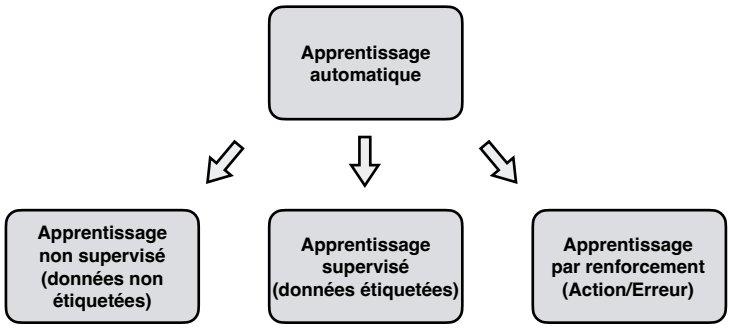

Figure 2. *Trois types d'algorithmes d'apprentissage automatique*

L'apprentissage automatique est la méthode d'IA la plus célèbre, qui permet à un système ou à un logiciel d'apprendre des connaissances à partir de données acquises. Comme le montre la figure 1, les réseaux neuronaux (RN) sont contenus dans l'apprentissage automatique (AA) et ils sont tous contenus dans l'intelligence artificielle (IA). L'apprentissage automatique fait appel à des connaissances telles que la théorie des probabilités, les statistiques, la théorie de l'approximation, l'analyse convexe et de nombreux autres sujets pluridisciplinaires. L'approche "basée sur les données" est l'idée centrale de l'apprentissage automatique qui permet aux algorithmes de faire des

prédictions et de prendre des décisions en fonction de l'analyse et de l'interprétation des données. Pour l'instant, l'apprentissage automatique peut être divisé en trois catégories (Figure 2) :

- Apprentissage supervisé

  Les données étiquetées désignent les éléments de données qui ont été marqués d'une ou plusieurs étiquettes identifiant des propriétés, des caractéristiques, des classifications ou des objets spécifiques. L'apprentissage supervisé permet d'établir un modèle de correspondance entre les entrées et les sorties sur l'ensemble de données d'exemples étiquetés, puis de prédire le résultat d'une nouvelle entrée en fonction des relations entrée-sortie que l'on a vues auparavant.

- Apprentissage non supervisé

  Les données non étiquetées font référence aux éléments de données qui n'ont pas été étiquetés avec des étiquettes identifiant les caractéristiques, les propriétés ou les classifications. L'apprentissage non supervisé permet de rechercher automatiquement des caractéristiques et des structures à partir de données non étiquetées, de regrouper les données en divers clusters, d'identifier des règles d'association et de réduire les dimensions pour réaliser des tâches telles que la segmentation, la détection de motifs et la détection d'anomalies.

- Apprentissage par renforcement

  Contrairement aux deux méthodes précédentes, les données ne sont plus obligatoires pendant l'apprentissage par renforcement. Il s'agit d'un processus consistant à recevoir des récompenses quantifiées de l'environnement avec différentes actions pour mettre à jour les paramètres du modèle. Autrement dit, l'apprentissage par renforcement est une approche d'apprentissage par "essai-erreur" qui consiste à interagir constamment avec l'environnement pour obtenir la meilleure stratégie en maximisant la récompense. "État, action, récompense" sont les trois éléments clés de l'apprentissage par renforcement. Le modèle observe le résultat de la décision à chaque étape, ce qui conduit à la décision suivante pour gagner l'objectif final. Le jeu et le robot sont les domaines les plus utilisés de cette méthode actuellement.

Le réseau de neurones artificiels (RNA) est actuellement l'un des algorithmes d'apprentissage supervisé les plus représentatifs. Cet article s'y intéressera pour parler de son origine et de son fonctionnement.

3 Réseau de neurones artificiels

En tant qu'algorithme d'apprentissage automatique le plus représentatif, le réseau neuronal artificiel est largement utilisé dans divers scénarios d'application et mène continuellement le développement de l'IA. Il est intéressant de noter que le RNA a été inspiré et conçu à partir de la structure du cerveau humain. Cette section commence par les neurones du cerveau humain, lève le voile sur le mystère des réseaux neuronaux artificiels étape par étape, et présente certaines techniques clés.

3.1 L'origine de RNA, neurone

Le cerveau est principalement constitué de neurones pour réaliser l'échange de signaux et le traitement de l'information. On estime qu'il y a près de 100 milliards de neurones dans le système nerveux central humain.
Le neurone peut recevoir, intégrer, conduire et émettre l'excitation. Structurellement, il peut être divisé en trois parties : la dendrite, le péricaryon et l'axone. Comme le montre l'image (Figure 3), il existe de nombreuses branches dendritiques sur la dendrite pour recevoir l'excitation d'autres axones de neurones adjacents, former les potentiels postsynaptiques et les transmettre au péricaryon. L'intensité du potentiel postsynaptique est liée à différentes propriétés et états des synapses. Ensuite, tous ces potentiels postsynaptiques sont combinés et additionnés dans le péricaryon ; un potentiel d'action sera généré si le potentiel seuil est atteint. Enfin, l'axone transmet le potentiel d'action du péricaryon à l'extrémité pour conduire l'excitation au neurone suivant.

3.2 Modèle d'imitation mathématique, perceptron

Le perceptron (Figure 4) est créé pour imiter mathématiquement les principes de fonctionnement du neurone [3]. Il possède plusieurs entrées binaires $x_1, x_2, ..., x_n$ (dendrites), à chaque entrée correspond un poids $w_1, w_2, ..., w_n$ (sensibilité des synapses). La somme de chaque entrée multipliée par son poids est ensuite comparée à un seuil -b (potentiel de seuil). La sortie est égale à 1 si la somme est supérieure au seuil (neurone actif). Sinon, la sortie est égale à 0 (neurone inhibé) (Figure 5 - 1. Fonction d'étape) :

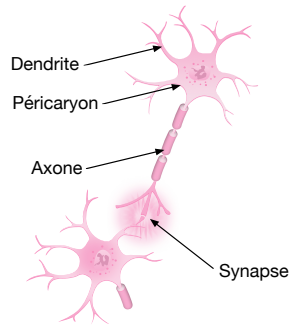

Figure 3. *Neurone*

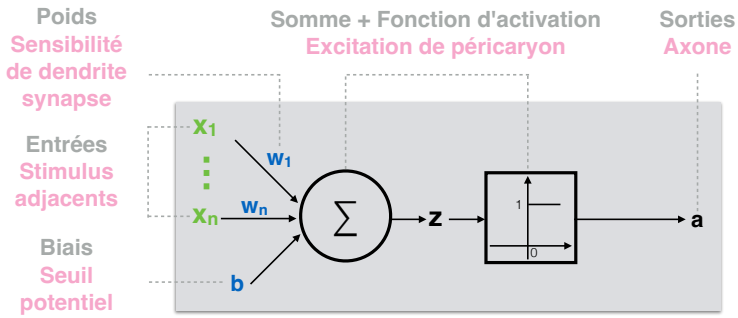

Figure 4. *Structure du perceptron*

$$a = \begin{cases} 0 \ si \ \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1 \ si \ \sum_{i=1}^{n} w_i x_i + b > 0 \end{cases} \tag{1}$$

3.3 Caractéristique non linéaire, fonction d'activation

Cependant, dans les applications algorithmiques, la fonction d'échelon s'avère compliquée à utiliser avec le développement des perceptrons et des RNA. Par conséquent, de nombreuses variantes de fonctions d'activation offrant des performances encore meilleures ont été découvertes.
La fonction sigmoïde (Figure 5 - 2) est une fonction de type "S" couramment utilisée, qui peut faire correspondre des variables à l'intervalle (0,1) :

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

et sa dérivée (Figure 5 - 3) peut être facilement calculée :

$$S'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = S(x)(1 - S(x)) \tag{3}$$

On observe que lorsque l'entrée de la fonction sigmoïde est très grande ou très petite, la sortie entre dans la zone "plate", et le gradient "disparaît". Alors que lorsque l'entrée est proche de 0, la valeur de la dérivée sigmoïde est plus grande.
En outre, la fonction sigmoïde produit directement un état de demi-saturation (sortie = 0,5) lorsque le seuil d'activation est tout juste atteint (entrée = 0), ce qui ne correspond pas aux véritables réseaux neuronaux biologiques. En général, seuls $1\% - 4\%$ des neurones du cerveau sont actifs (sortie > 0) simultanément et ne sont pas facilement saturés.
Par conséquent, la fonction unité linéaire rectifiée (ReLU) (Figure 5 - 4) est plus conforme aux modèles physiologiques et est plus adaptée pour éviter les gradients qui explosent et disparaissent. En outre, elle peut également simplifier le calcul et accélérer la convergence car sa dérivée est toujours égale à 1 si l'entrée est positive :

$$R(x) = max(0, x) \tag{4}$$

Les fonctions d'activation non linéaires ci-dessus peuvent introduire des caractéristiques non linéaires dans le modèle RNA. Dans certaines conditions, ces fonctions peuvent aider le modèle à se rapprocher de n'importe quelle relation de mappage non linéaire continue entre l'entrée et la sortie avec une certaine précision s'il y a suffisamment de neurones et de couches cachées [4].

3.4 Construction d'un modèle de réseau, perceptron multicouche

Le perceptron multicouche (Figure 6) est un modèle RNA à anticipation entièrement connecté qui peut faire correspondre un ensemble de données d'entrée à un ensemble de données de sortie. Par exemple, les entrées peuvent être les valeurs des caractéristiques d'images ou de documents, tandis que les sorties du modèle réaliseront une tâche de prédiction spécifique, telle que l'identification d'un objet.

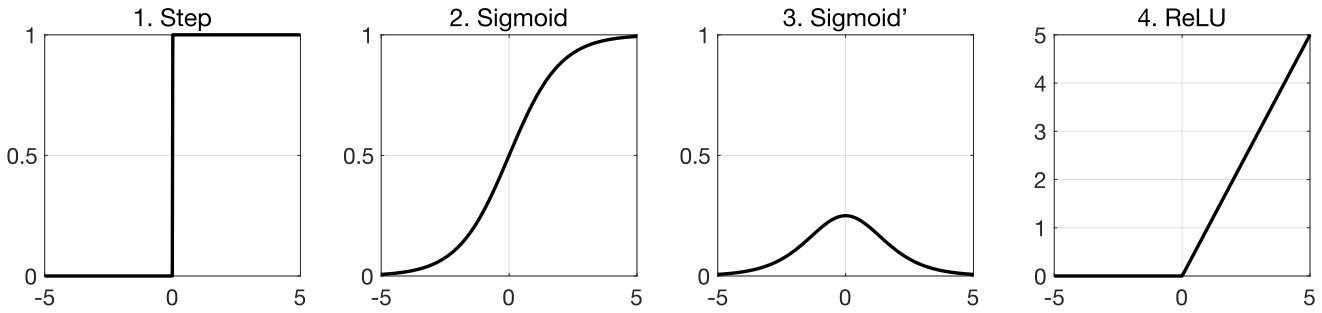

Figure 5. *Fonction Step, fonction Sigmoïde, sa dérivée et fonction ReLU*

Le modèle se compose de la couche d'entrée (variables d'entrée), des couches cachées (nœuds intermédiaires) et de la couche de sortie (variables de sortie). Les neurones sont entièrement connectés, ce qui signifie que les neurones de la dernière couche sont connectés à chaque neurone de la couche précédente.
Une fois que nous avons compris comment les perceptrons individuels reçoivent, traitent et transmettent les signaux, il est temps de voir comment calculer les valeurs couche par couche de l'entrée à la sortie dans un modèle entièrement connecté pour réaliser la propagation vers l'avant :

$$\begin{cases} a_1 = \sigma(z_1) = \sigma(x_1 w_1 + x_2 w_3 + b_1) \\ a_2 = \sigma(z_2) = \sigma(x_1 w_2 + x_2 w_4 + b_2) \\ a_3 = \sigma(z_3) = \sigma(a_1 w_5 + a_2 w_7 + b_3) \\ \dots \\ a_n = \sigma(z_n) = \sigma(\mathbf{a} * \mathbf{W} + b_n) \end{cases} \tag{5}$$

où a_n est la sortie du neurone courant, σ est la fonction d'activation, $\mathbf{a}$ et $\mathbf{W}$ sont les formes matricielles des sorties et de leurs poids correspondants de la couche précédente, b_n est le biais.

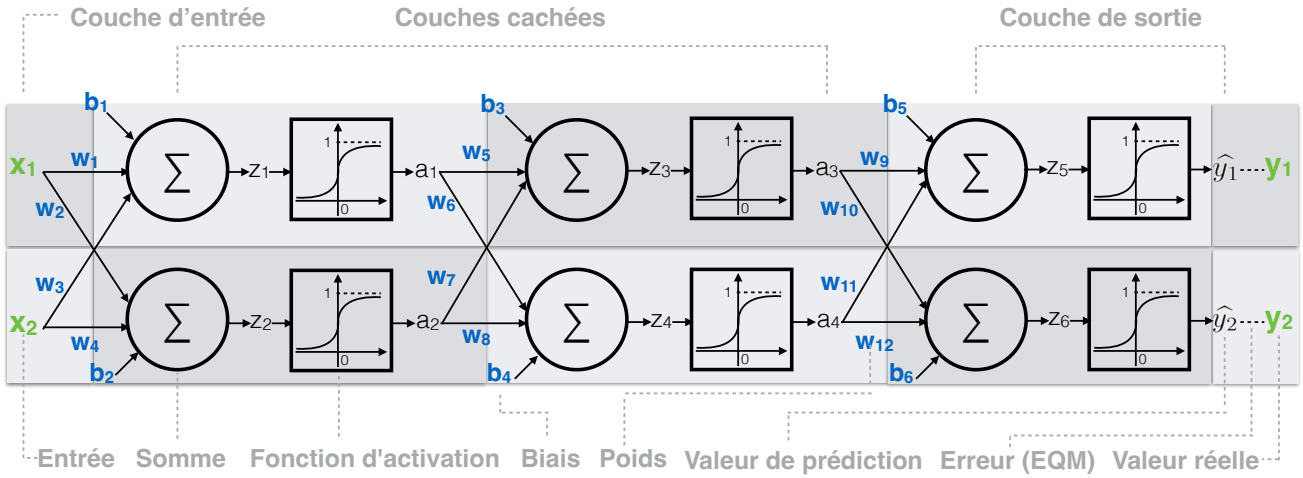

Figure 6. *Structure de perceptron multicouche*

3.5 Évaluation de la satisfaction des résultats, fonction de perte

Comme mentionné dans la section précédente, dans la propagation vers l'avant, chaque couche reçoit la sortie de la couche précédente, traite la valeur et transmet le résultat à la couche suivante. Nous obtenons la valeur prédite $\hat{y}_i$ à la couche de sortie. Pour évaluer la précision de cette prédiction , une fonction de perte permet de quantifier la qualité des sorties du RNA. L'erreur quadratique moyenne (EQM) est l'une des fonctions de perte les plus couramment utilisées :

$$L = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{6}$$

La valeur de perte peut exprimer l'écart entre la prédiction du RNA $\hat{y}_i$ et le résultat réel y_i. Plus la valeur de perte est grande, plus la prédiction est éloignée de notre attente. "L'entraînement" ou "l'apprentissage" consiste à itérer et à ajuster les paramètres internes du modèle de manière continue ($\theta = \{w_1, w_2, .., w_j, b_1, b_2, .., b_k\}$) pour obtenir des prédictions plus précises. En d'autres termes, l'apprentissage du modèle vise à minimiser la valeur de perte $L(\theta)$ en optimisant les paramètres internes θ pour que la valeur prédite soit aussi proche que possible de la valeur réelle :

$$\underset{\theta}{argmin}\, L(\theta) \tag{7}$$

3.6 Comment ajuster les paramètres, optimisation basée sur la descente de gradient

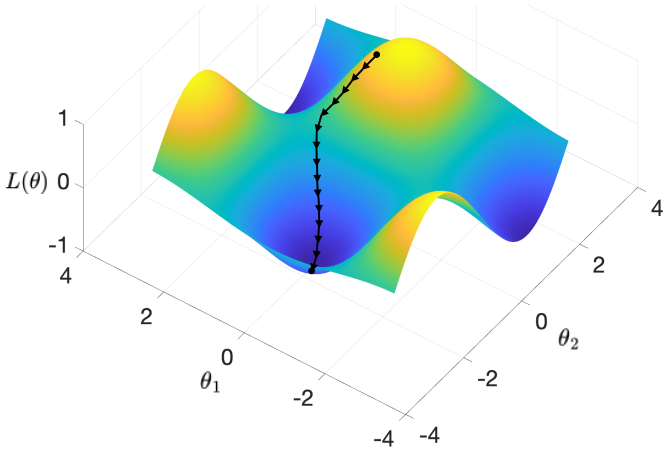

Figure 7. *Démonstration de l'optimisation par descente de gradient*

Bien que les paramètres internes du modèle soient à haute dimension, nous pouvons ici utiliser des paramètres à deux dimensions (θ_1, θ_2) comme exemple pour observer comment ils sont progressivement optimisés.
On peut observer que, pour une fonction différentiable, se déplacer dans la direction opposée au gradient de la fonction peut faire diminuer la valeur de la fonction de la manière la plus rapide. En effet, la direction du gradient est la direction de montée au premier ordre la plus rapide de la fonction au point donné. Le minimum local peut être finalement atteint en utilisant cette méthode de manière itérative (Figure7). En d'autres termes, le moyen le plus rapide de descendre est de trouver la direction la plus raide de la position actuelle, puis de descendre dans cette direction. Ensuite, il faut trouver la direction la plus raide de la position suivante et avancer dans cette nouvelle direction jusqu'à atteindre finalement le point local le plus bas. Par conséquent, une fois le gradient obtenu, la valeur de la perte peut diminuer en optimisant et en itérant les paramètres internes du RNA pour atteindre le minimum local de la fonction de perte [5] :

$$\theta^{n+1} = \theta^n - \alpha \bigtriangledown L(\theta) \tag{8}$$

où, θ^{n+1} est la position suivante (θ_1^{n+1}, θ_2^{n+1}), θ^n est la position actuelle (θ_1^n, θ_2^n), "-"représente la direction opposée, α est une petit étape, et $\bigtriangledown L(\theta)$ est la direction du gradient de l'augmentation la plus rapide.
α est appelé le facteur d'apprentissage dans l'algorithme de descente du gradient. Si α est trop grand, le point le plus bas peut être manqué ; si α est trop petit, la vitesse de descente peut être réduite, n représente le nombre d'itérations. Dans la pratique, l'itération peut être arrêtée lorsque :

- Le nombre maximal d'itérations est atteint
- La différence entre les deux valeurs de perte successives est inférieure à un petit seuil ϵ (la valeur de perte diminue très lentement et peut se rapprocher du minimum local/global).
- L'erreur absolue W, b de deux itérations successives est inférieure à un petit seuil (la valeur de la perte diminue très lentement et peut se rapprocher du minimum local/global).

3.7 Calcul du gradient couche par couche, rétropropagation (RP)

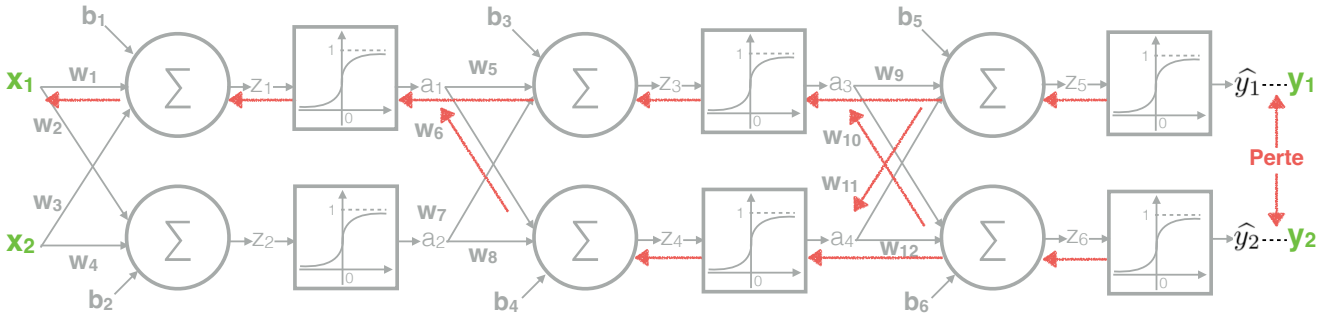

Figure 8. *Rétropropagation*

Afin d'obtenir les paramètres minimisant la valeur de la perte, il est nécessaire de calculer le gradient à chaque itération, ce qui est compliqué en raison de la masse des paramètres internes (w_j et b_k) du modèle RNA. Cependant, ces paramètres sont souvent liés les uns aux autres entre les couches successives. La méthode de rétropropagation inventée par Rumelhart et al. en 1986 [6] permet de calculer le gradient couche par couche à partir de la dernière. La rétropropagation réparti le montant de l'erreur entre les connexions réduisant la répétition des calculs et améliorant considérablement l'efficacité de la recherche du gradient. La partie suivante detaille l'algorithme d'optimisation du poids minimisant la valeur de perte.
La méthode RP est principalement basée sur la règle de la dérivation en chaîne, par exemple, pour les fonctions différentiables h, g, k, et les variables z, y, x, s :

- cas 1
  Si $z = h(y)$ et $y = g(x)$, alors $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$.
- cas 2
  Si $z = k(x, y)$, $x = g(s)$ et $y = h(s)$, alors $\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$

Les gradients des paramètres du RNA (par exemple, $\frac{\partial L}{\partial w_9}$, $\frac{\partial L}{\partial w_5}$ et $\frac{\partial L}{\partial w_1}$) peuvent être calculés, connaissant la valeur donnée (x_1,x_2,y_1,y_2) et la valeur calculée (z_1,...,z_6,a_1,...,a_4,$\hat{y_1}$,$\hat{y_2}$) dans la propagation (voir dans la section 3.4) :

- $\frac{\partial L}{\partial w_9}$ (voir dans la figure 8, Processus similaire pour le $\frac{\partial L}{\partial w_{10}}$,$\frac{\partial L}{\partial w_{11}}$,$\frac{\partial L}{\partial w_{12}}$,$\frac{\partial L}{\partial b_5}$,$\frac{\partial L}{\partial b_6}$)
  Selon la règle de la chaîne (cas 1) :

$$\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial \hat{y_1}}\frac{\partial \hat{y_1}}{\partial z_5}\frac{\partial z_5}{\partial w_9} \tag{9}$$

  1. $\frac{\partial L}{\partial \hat{y_1}} = \hat{y_1} - y_1$, parce que $L = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2$
  2. $\frac{\partial \hat{y_1}}{\partial z_5} = \hat{y_1}(1 - \hat{y_1})$, parce que $\hat{y_1} = S(z_5) = \frac{1}{1+e^{-z_5}}$ et $S'(z_5) = \frac{e^{-z_5}}{(1+e^{-z_5})^2} = S(z_5)(1 - S(z_5))$
  3. $\frac{\partial z_5}{\partial w_9} = a_3$, parce que $z_5 = a_3 * w_9 + a_4 * w_{11} + b_5$

- $\frac{\partial L}{\partial w_5}$ (Processus similaire pour le $\frac{\partial L}{\partial w_6}$,$\frac{\partial L}{\partial w_7}$,$\frac{\partial L}{\partial w_8}$,$\frac{\partial L}{\partial b_3}$,$\frac{\partial L}{\partial b_4}$)
  Selon la règle de la chaîne (cas 1 + cas 2) :

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial \hat{y_1}}\frac{\partial \hat{y_1}}{\partial z_5}\frac{\partial z_5}{\partial a_3}\frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial w_5} + \frac{\partial L}{\partial \hat{y_2}}\frac{\partial \hat{y_2}}{\partial z_6}\frac{\partial z_6}{\partial a_3}\frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial w_5} \tag{10}$$

$$= (\frac{\partial L}{\partial \hat{y_1}}\frac{\partial \hat{y_1}}{\partial z_5}\frac{\partial z_5}{\partial a_3} + \frac{\partial L}{\partial \hat{y_2}}\frac{\partial \hat{y_2}}{\partial z_6}\frac{\partial z_6}{\partial a_3})\frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial w_5} \tag{11}$$

$$= (\frac{\partial L}{\partial z_5}\frac{\partial z_5}{\partial a_3} + \frac{\partial L}{\partial z_6}\frac{\partial z_6}{\partial a_3})\frac{\partial a_3}{\partial z_3}\frac{\partial z_3}{\partial w_5} \tag{12}$$

1. $\frac{\partial L}{\partial z_5} = \frac{\partial L}{\partial \hat{y_1}} \frac{\partial \hat{y_1}}{\partial z_5}$ et $\frac{\partial L}{\partial z_6} = \frac{\partial L}{\partial \hat{y_2}} \frac{\partial \hat{y_2}}{\partial z_6}$ sont déjà calculés ci-dessus
2. $\frac{\partial z_5}{\partial a_3} = w_9$ et $\frac{\partial z_6}{\partial a_3} = w_{10}$
3. $\frac{\partial a_3}{\partial z_3} = a_3(1 - a_3)$
4. $\frac{\partial z_3}{\partial w_5} = a_1$

- $\frac{\partial L}{\partial w_1}$ (Processus similaire pour le $\frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial b_2}$)
  De même, selon la règle de la chaîne et les gradients calculés ci-dessus, nous avons :

$$\frac{\partial L}{\partial w_1} = (\frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial a_1} + \frac{\partial L}{\partial z_4}\frac{\partial z_4}{\partial a_1})\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1} \tag{13}$$

$$= (\frac{\partial L}{\partial z_3}w_5 + \frac{\partial L}{\partial z_4}w_6)a_1x_1 \tag{14}$$

Le calcul ci-dessus ne porte que sur un seul ensemble de données. Les gradients des paramètres de la fonction de perte de tous les ensembles de données sont la somme des résultats de chaque donnée :

$$\frac{\partial L(\theta)}{\partial w_j} = \frac{\partial \sum_{n=1}^{N} L^n(\theta)}{\partial w_j} = \sum_{n=1}^{N} \frac{\partial L^n(\theta)}{\partial w_j} \tag{15}$$

où n est le nombre de chaque donnée et N représente la volume de données.
Le même principe de calcul du gradient est utilisé pour les RNA avec un grand nombre de couches et de neurones :

$$\bigtriangledown L(\theta) = \begin{bmatrix} \frac{\partial L(\theta))}{\partial w_1} \\ \dots \\ \frac{\partial L(\theta))}{\partial w_j} \\ \frac{\partial L(\theta))}{\partial b_1} \\ \dots \\ \frac{\partial L(\theta))}{\partial b_k} \end{bmatrix} \tag{16}$$

3.8 Paramètre constant, hyperparamètre

Si la majorité des paramètre sont entraînable, il y a des paramètres constants non entraînables dont les valeurs sont fixées avant le processus d'apprentissage. Ils peuvent être définis comme des hyperparamètres et ont une influence cruciale sur les performances du modèle. Pour un RNA primaire, voici quelques hyperparamètres typiques :

- Structure du réseau

  Tels que le nombre de couches, le nombre de neurones par couche, le type de fonction d'activation, etc.

- Paramètre d'optimisation

  Tels que la méthode d'optimisation, le taux d'apprentissage, la taille du lot, etc.

L'ajustement des hyperparamètres est un problème d'optimisation non linéaire, non différentiable et non convexe qui ne peut être résolu avec les méthodes d'optimisation classiques (par exemple, la descente de gradient mentionnée ci-dessus). Pour chaque combinaison d'hyperparamètre, un entrainement entier doit être effectué pour évaluer les performances. Si le nombre d'hyperparamètre est élevé le nombre d'entrainement a effectué peut etre énorme. Voici trois approches courantes pour effectuer de manière optimal la selection des hyperparamètres:

- Recherche dans la grille

  Après sélections de différentes valeurs pour chaque hyperparamètres. Le modèle est entraîné pour toutes les combinaisons possibles d'hyperparamètres. La combinaison proposant les meilleurs performances est sélectionnée.

- Recherche aléatoire

  Certains hyperparamètres ont un impact limité, tandis que d'autres ont une influence beaucoup plus importante sur les performances du modèle. La méthode de recherche par grille fera des essais inutiles sur des hyperparamètres peu importants. La méthode de recherche aléatoire génère des combinaisons aléatoires des hyperparamètres afin de configurer l'optimum.

- Optimisation bayésienne

  Contrairement aux deux méthodes précédentes, lorsque plusieurs ensembles d'hyperparamètres sont déjà testés, il est raisonnable de tirer parti de ces résultats existants pour déterminer le suivant. L'optimisation bayésienne établit d'abord une fonction proxy probabiliste qui s'adapte aux résultats testés afin d'approcher la fonction de performance réelle. Elle recherche ensuite un nouvel ensemble d'hyperparamètres ayant la plus forte probabilité de performance optimale dans la fonction proxy et teste sa performance réelle. Après plusieurs itérations, la fonction proxy probabiliste se rapproche de la fonction de performance réelle, notamment dans la zone de performance la plus élevée. Enfin, la fonction proxy peut nous aider à trouver l'ensemble optimal d'hyperparamètres.

Le volume de calcul de ces méthodes augmente de façon exponentielle à mesure que la dimension des hyperparamètres augmente. Il est possible donc possible d'arreter le processus d'apprentissage dès le depart si les résultats ne sont pas promettant, ce qui limite les entrainements inutile. Il est aussi possible de faire appel à des ingénieurs possédant des connaissances et une expérience approfondies pour optimiser les hyperparamètres.

3.9 Qu'est-ce que l'apprentissage profond ?

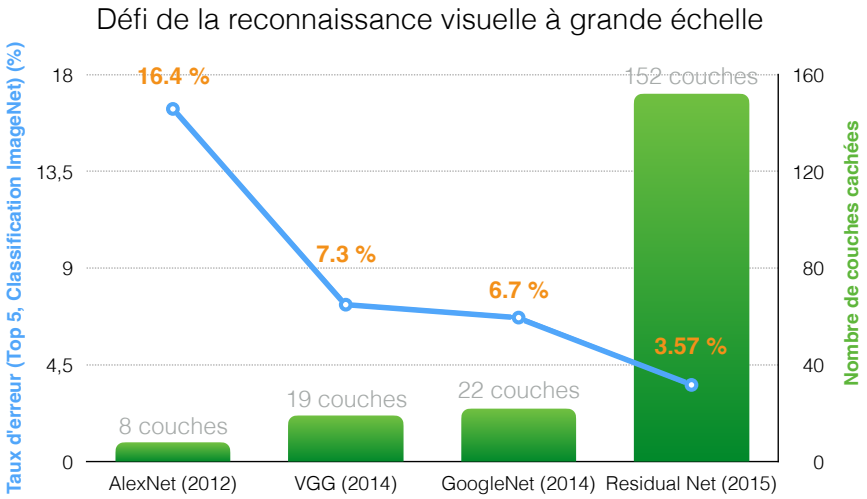

Figure 9. *Révolution de profondeur*

L'apprentissage profond est un type de réseaux neuronaux composés de plusieurs couches cachées qui permettent d'extraire les caractéristiques des données avec plusieurs degré d'abstraction [7]. Par exemple, les réseaux de neurones convolutionnels (CNN) prennent les pixels d'une image de visage en entrée. Les couches du réseau extraient les caractéristiques de l'images, en commençant par les caractéristiques globales (segment de courbe, bord orienté, couleur, forme du visage) en allant jusqu'à des caractéristiques plus détaillées (œil, nez, bouches). Le réseau retourne alors un score permettant d'identifier à qui appartient le visage.
En général, plus le nombre de paramètres composant les couches du modèle est important, plus la précision de la prédiction est élevée. Si le nombre de paramètres internes est limité, il est préférable d'utiliser plus de couches mais moins de neurones par couche.
Les paramètres du modèle CNN peuvent être considérablement réduits grâce à de nouvelles idées comme la perception locale et le partage des paramètres. En outre, avec les progrès de l'électronique et des technologies de l'information, en particulier le développement des GPU (processeurs graphiques) et l'application de capteurs à grande échelle pour l'acquisition massive de données, il est désormais possible d'entraîner le modèle CNN avec de plus en plus de couches et de réduire l'erreur de prédiction au fil du temps [8] (Figure 9).

3.10 Limitation actuelle

Le RNA présente certaines limites qui ne peuvent être ignorées à l'heure actuelle, comme l'obtention des données étiquetées de haute qualité, le problème de surapprentissage et l'impact écologique.
Comme nous le savons, pour l'apprentissage supervisé, les données d'apprentissage doivent être étiquetées. La quantité et la qualité des données d'apprentissage déterminent largement les performances du modèle RNA. Cependant, les données étiquetées de haute qualité sont aujourd'hui coûteuses et difficiles à obtenir.
Le surapprentissage est un autre problème courant pendant l'apprentissage des RNA, ce qui signifie que le modèle s'adapte trop specifiquement à l'ensemble d'apprentissage et commence à apprendre le bruit et les informations inutiles. Cela détériore les performances de généralisation du modèle sur le nouvel ensemble de données et nécessite certaines techniques pour l'éviter.
Lors de l'exemple de l'Alpha Go cité en tout début d'article, énormément d'énergie a été consommée pour l'entraînement de l'IA. Durant le match de Go contre Lee Sedol, l'impact environnemental n'a pas été négligeable : l'Alpha Go a utilisé 1202 CPU (processeur) et 176 GPU pour le calcul !

4 Conclusion

En raison de l'explosion de la puissance de calcul et du volume des données ces dernières années, les réseaux de neurones artificiels ont connu un grand essor dans l'agriculture, l'industrie et les services, accomplissant les fonctions qui ne peuvent être réalisées avec les méthodes traditionnelles. Les RNA ont déjà apporté une valeur ajoutée importante et une amélioration d'efficacité significative dans l'évolution de la société.
Par certains aspects, comme le stockage/traitement de données à grande échelle ou encore l'analyse statistique, le RNA présente plus de cohérence que l'être humain. Il démontre également d'excellentes performances dans des domaines spécifiques, par exemple, la vision par ordinateur, la reconnaissance vocale et les recommandations intelligentes. En revanche, en termes d'innovation/conception et de prise de décision, il lui reste encore un long chemin à parcourir, parce que "la machine ne peut pas penser et avoir une conscience comme les humains".
Malgré les nombreux outils permettant de déduire certaines règles à partir des modèles entrainés de manière très précise, cette technologie a encore une capacité d'interprétation très limitée. Elle ne peut pas gagner la confiance des gens dans les applications nécessitant une fiabilité, une précision et une stabilité élevées, comme la conduite autonome, les équipements médicaux, les centrales nucléaires, les robots collaboratifs et l'industrie aérospatiale.
Par conséquent, pour concrétiser un projet d'IA, il serait essentiel d'analyser les besoins des utilisateurs, les possibilités de rentabilité d'un point de vue commercial, mais aussi d'évaluer la faisabilité du point de vue de la science des données. Nous pourrions également avoir besoin de l'aide d'algorithmes traditionnels efficaces, de modèles personnalisés et d'une coopération étroite avec d'autres technologies avancées pour intégrer un écosystème bien organisé pour atteindre finalement la transformation digitale et intelligente de notre société.

References

[1] Nilsson, N. J., Nilsson, N. J. (1998). Artificial intelligence: a new synthesis. Morgan Kaufmann.

[2] Stephenson Smith, S. (2003). The new international webster's comprehensive dictionary of the english language: deluxe encyclopedic edition (No. REF 428.03 STE. CIMMYT).

[3] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

[4] Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.

[5] Machine learning course: https://www.coursera.org/learn/machine-learning

[6] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

[7] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[8] ILSVRC data: http://www.image-net.org/challenges/LSVRC/