

Introduction à l'apprentissage profond

Réseau de Neurones Artificiels

Chaker Maleke

Houimli Asma

Benoun Mohamed Mahdi

Session IDS4

Ingénierie en Science des Données

Faculté des Sciences de Tunis El Manar

February 10, 2024

Abstract

Ce projet vise à fournir une compréhension approfondie des principes fondamentaux du apprentissage profond (Deep Learning). Nous explorons les distinctions cruciales entre l'Apprentissage Machine (machine learning) et le l' apprentissage profond, mettant l'accent sur l'évolution des approches algorithmiques.

La première section du projet se consacre à l'exploration des concepts fondamentaux des réseaux de neurones. Elle propose une analyse détaillée du fonctionnement du perceptron monocouche (Single Layer Perceptron) ainsi que du perceptron multicouche (Multi-Layer Perceptron). Cette étape jette les bases essentielles pour la compréhension des réseaux de neurones artificiels (ANN).

Dans la seconde partie, nous approfondissons le concept des réseaux de neurones artificiels (ANN), nous explorons leur relation avec les réseaux neuronaux biologiques. Nous détaillons le processus du "Forward Pass" dans un ANN, suivi d'une explication approfondie de l'algorithme de rétropropagation (Backpropagation), qui joue un rôle essentiel dans l'ajustement des poids du réseau en fonction des erreurs. En outre, nous examinons les optimiseurs tels que la Descente de Gradient (Gradient Descent) et Adam, qui jouent un rôle crucial dans l'optimisation des modèles de réseaux de neurones.

Ce projet offre ainsi une introduction complète aux concepts clés du Deep Learning, de la structure des réseaux de neurones à l'application des optimiseurs, fournissant une base solide pour toute personne cherchant à approfondir ses connaissances dans ce domaine en constante évolution.

Contents

	Page
1 Introduction L'apprentissage profond	3
1.1 Introduction	3
1.2 Réseaux de Neurones	8
1.3 Perceptron Monocouche	9
1.4 Perceptron Multicouche	13
2 Les Réseaux de Neurones Artificiels	15
2.1 Introduction des Réseaux de Neurones Artificiels	15
2.2 Relation entre réseau neurone artificiel (ANN) et réseau neurone biologique	16
2.3 Processus du Forward Pass et Backpropagation	18
3 Optimisation dans les Artificial Neural Network	23
3.1 Descente de Gradient:	23
3.2 Optimiseurs Avancés	25
3.3 Sélection d'Optimiseurs en Fonction du Problème	28
4 Conclusion	29

1 Introduction L'apprentissage profond

1.1 Introduction

1.1.1 Définition de L'apprentissage automatique

L'apprentissage automatique est une sous-catégorie de l'intelligence artificielle (IA) qui permet aux ordinateurs d'apprendre à partir de données et de prendre des décisions sans programmation explicite. Il englobe diverses techniques et algorithmes qui permettent aux systèmes de reconnaître des motifs, de faire des prédictions et d'améliorer leurs performances au fil du temps.

1.1.2 Définition de L'apprentissage profond

L'apprentissage profond (Deep Learning) est un type d'apprentissage automatique (machine learning) qui apprend aux ordinateurs à accomplir des tâches en s'inspirant des exemples, tout comme le font les êtres humains. Imaginez enseigner à un ordinateur à reconnaître les chats : au lieu de lui dire de rechercher des vibrisses, des oreilles et une queue, vous lui montrez des milliers de photos de chats. L'ordinateur découvre les motifs communs tout seul et apprend à identifier un chat. C'est l'essence de l'apprentissage profond.

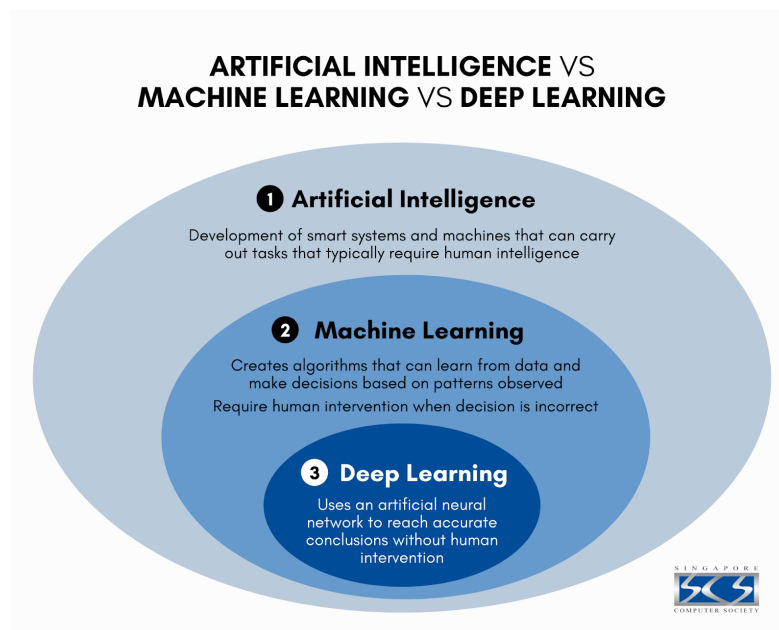


Figure 1.1: Intelligence Artificielle vs Machine Learning vs Deep Learning

1.1.3 L'importance de l'apprentissage profond

Les motivations derrière l'adoption généralisée de l'apprentissage profond dans l'industrie sont :

- Gestion des données non structurées : la capacité de l'apprentissage à traiter des informations complexes et non structurées, telles que les images, les vidéos.. ,ce qui réduit le temps et les ressources nécessaires à la normalisation des ensembles de données.
- Haute précision : Les modèles d'apprentissage profond fournissent les résultats les plus précis en vision par ordinateur, traitement du langage naturel (NLP) et traitement audio.
- Reconnaissance de motifs : La plupart des modèles nécessitent l'intervention d'ingénieurs en apprentissage automatique, mais les modèles d'apprentissage profond peuvent détecter automatiquement toutes sortes de motifs.

1.1.4 Les applications de l'apprentissage profond

Récemment, le monde de la technologie a connu une augmentation des applications d'intelligence artificielle, toutes alimentées par des modèles d'apprentissage profond. Voici quelques-unes des applications les plus célèbres construites à l'aide de l'apprentissage profond.

- La vision par ordinateur (Computer Vision) est utilisée dans les voitures autonomes pour détecter des objets et éviter les collisions. Elle est également utilisée pour la reconnaissance faciale, l'estimation de la pose, la classification d'images et la détection d'anomalies.
- La reconnaissance automatique de la parole (Automatic Speech Recognition) est utilisée par des milliards de personnes dans le monde. Elle est présente sur nos téléphones et est couramment activée en disant "Hey, Google" ou "Salut, Siri". De telles applications audio sont également utilisées pour la conversion texte-parole, la classification audio et la détection d'activité vocale.
- L'intelligence artificielle générative (Intelligence Artificielle Générative) a connu une forte demande avec la vente d'un CryptoPunk NFT pour 1 million de dollars. CryptoPunk est une collection d'art génératif créée à l'aide de modèles d'apprentissage profond. L'introduction du modèle GPT-4 par OpenAI a révolutionné le domaine de la génération de texte avec son puissant outil ChatGPT.
- Biomédical, Ce domaine a bénéficié énormément de l'introduction de l'apprentissage profond. L'apprentissage profond est utilisé en biomédecine pour détecter le cancer, élaborer des médicaments stables, pour la détection d'anomalies dans les radiographies thoraciques et pour assister les équipements médicaux.



Figure 1.2: Collection de CryptoPunk

1.1.5 Différence entre Machine Learning et Deep Learning

Le deep learning est une technique plus spécifique qui utilise des réseaux neuronaux artificiels pour modéliser et résoudre des problèmes complexes. Les réseaux neuronaux profonds sont inspirés de la structure du cerveau humain, avec des couches de neurones interconnectées travaillant ensemble pour traiter des informations de manière hiérarchique.

Ainsi, le processus d'apprentissage profond imite, dans une certaine mesure, la manière dont le cerveau humain apprend et extrait des connaissances à partir de l'expérience.

Machine Learning	Deep Learning
Sous-ensemble de l'AI	Sous-ensemble de Machine Learning
Peut s'entraîner sur des ensembles de données plus petits. Les données sont généralement structurées	Nécessite de grandes quantités de données. Les données peuvent être non structurées
Nécessite une intervention humaine plus importante pour corriger et apprendre:	Apprend de lui-même à partir de l'environnement et des erreurs passées.
Le feature extraction : sélectionner les variables avec lesquelles l'équipe Data va travailler	Pas de feature extraction : L'algorithme va être entraîné pour sortir lui-même les éléments influents dans la prédiction que vous souhaitez réaliser.
Entraînement plus court et précision plus basse.	Entraînement plus long et précision plus élevée.
Établit des corrélations simples et linéaires.	Établit des corrélations non linéaires et complexes.
Peut s'entraîner sur un CPU (unité centrale de traitement).	Nécessite une GPU (unité de traitement graphique) spécialisée pour l'entraînement.

Table 1.1: Tableau Comparative entre ML et DL

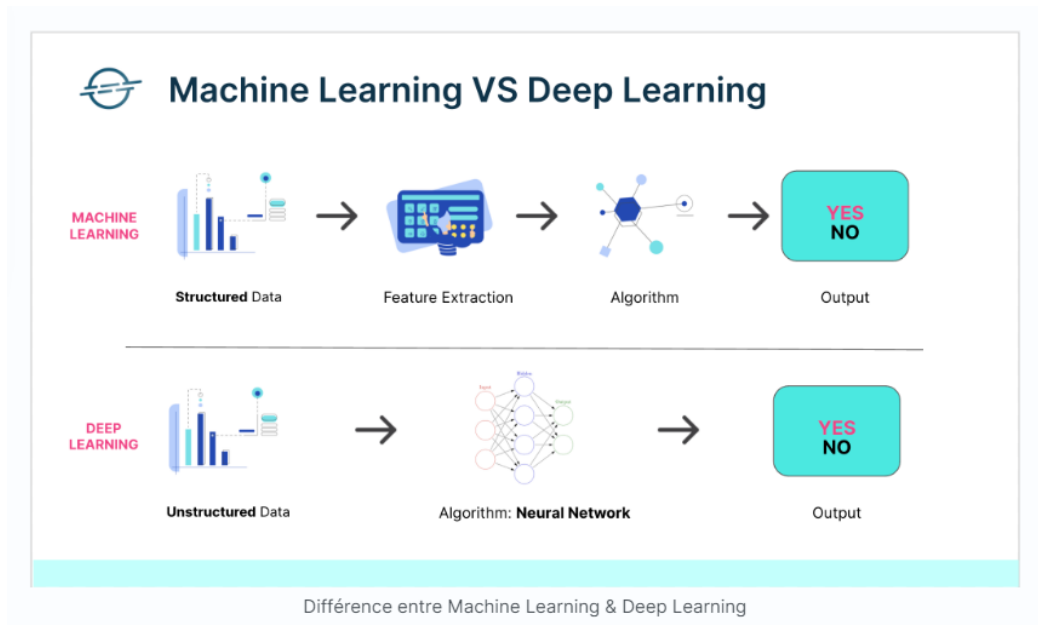


Figure 1.3: Machine learning VS Deep learning

1.1.6 Émergence récente du deep learning : Une convergence de données abondantes et de puissance de calcul accrue.

Les méthodes d'apprentissage machine ne sont pas récentes mais datent des années 60. Dès les débuts de l'informatique, les pionniers avaient imaginé des systèmes et des principes d'apprentissage à base de réseaux de neurones. Les bases de ce fonctionnement ont été développées sur des machines depuis 1986 avec notamment les travaux de Geoffrey Hinton, universitaire canadien.

Alors le deep learning existe depuis des décennies, mais pourquoi ses idées connaissent-elles un tel essor seulement maintenant ?

Aujourd'hui, nous disposons d'une immense quantité de données. Nous bénéficions également d'une puissance de calcul et d'une échelle bien plus importantes.

Si vous disposez d'une petite quantité de données, les algorithmes traditionnels (régression linéaire, SVM...) cessent généralement de s'améliorer à un niveau spécifique, même si nous alimentons l'algorithme avec davantage de données.

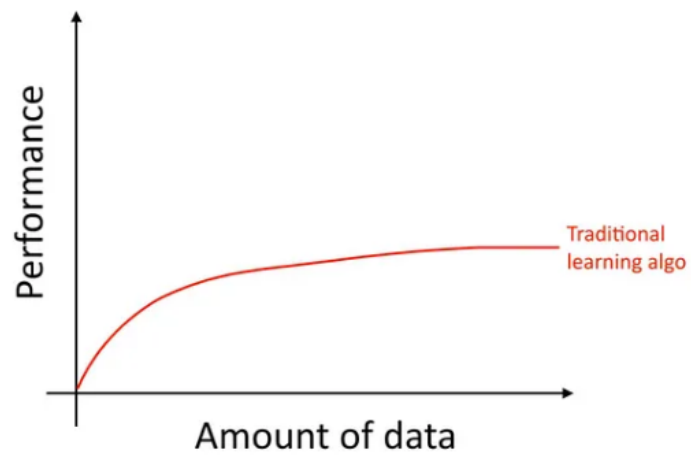


Figure 1.4: Performance des algorithmes traditionnels

En revanche, en formant un petit réseau neuronal avec la même quantité de données, vous pourriez obtenir de meilleures performances.

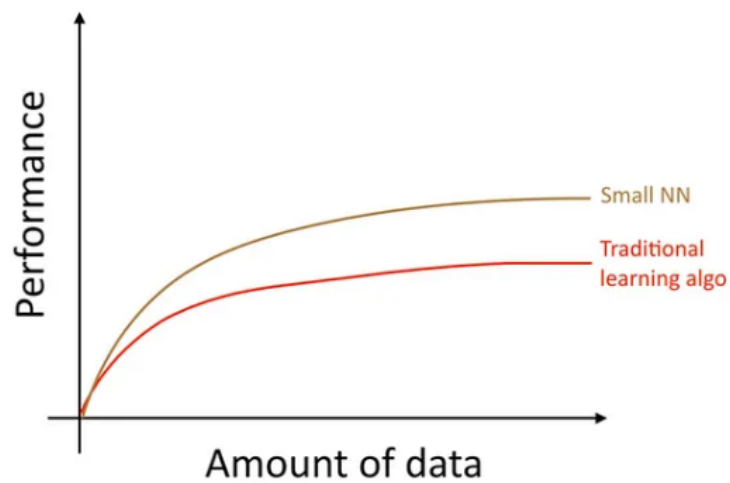


Figure 1.5: Algorithme traditionnel VS un petit réseau neuronal

Si nous formons un réseau neuronal plus grand, nous pouvons obtenir des performances bien meilleures.

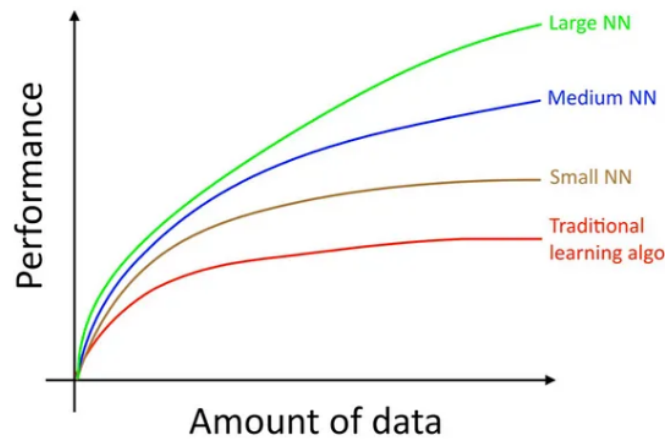


Figure 1.6: Algorithme traditionnel VS réseau neuronal en augmentant la quantité des données

Ainsi, si vous souhaitez atteindre ce très haut niveau de performance, vous avez besoin de deux éléments :

1. Souvent, vous devez être capable d'entraîner un réseau neuronal suffisamment grand afin de tirer parti de l'énorme quantité de données.
2. Vous avez besoin de beaucoup de données.

Cependant, cela fonctionne seulement jusqu'à un certain point, car finalement, vous pouvez épuiser vos données ou votre réseau neuronal devient si grand qu'il met trop de temps à être entraîné.

1.2 Réseaux de Neurones

Les réseaux de neurones artificiels (RNA) constituent la pierre angulaire du domaine du deep learning. Ils sont conçus pour imiter le fonctionnement du cerveau humain et sont largement utilisés dans diverses applications, de la reconnaissance d'images à la traduction automatique.

1.2.1 Architecture des Réseaux de Neurones

Un réseau de neurones est structuré en couches, chaque couche contenant un certain nombre de neurones. Les couches principales sont la couche d'entrée, les couches cachées et la couche de sortie. La couche d'entrée reçoit les données, les couches cachées effectuent des transformations complexes, et la couche de sortie produit les résultats finaux.

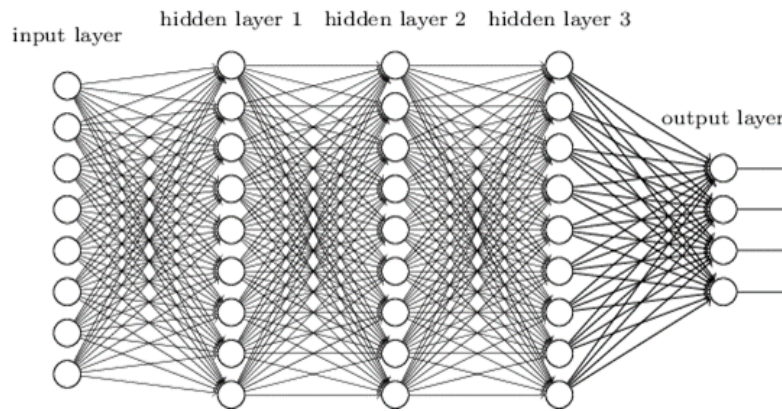


Figure 1.7: Architecture des Réseaux de Neurones

1. La couche d'entrée: est la première couche du réseau, recevant les données brutes. Chaque neurone dans cette couche représente une caractéristique d'entrée.
2. Les couches cachées: effectuent des calculs complexes sur les entrées. Plusieurs couches cachées permettent au réseau d'apprendre des représentations hiérarchiques des données, capturant des caractéristiques de plus en plus abstraites.
3. La couche de sortie: produit les résultats du réseau. La configuration de cette couche dépend de la tâche, par exemple, une seule sortie pour la classification binaire, ou plusieurs pour la classification multi-classes.

1.3 Perceptron Monocouche

1.3.1 Définition perceptron

Un Perceptron est un neurone artificiel, et donc une unité de réseau de neurones. Il effectue des calculs pour détecter des caractéristiques ou des tendances dans les données d'entrée.

Le Perceptron joue un rôle essentiel dans les projets de Machine Learning. Il est massivement utilisé pour classifier les données, ou en guise d'algorithme permettant de simplifier ou de superviser les capacités d'apprentissage de classificateurs binaires.

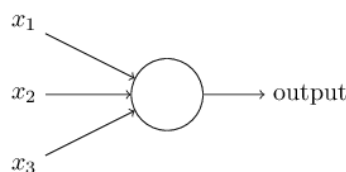


Figure 1.8: Neuron

1.3.2 Historique

C'est en 1957 que le Perceptron fut inventé par Frank Rosenblatt au laboratoire aéronautique de Cornell.

Dans les années 80, l'enchaînement de plusieurs couches était réalisé. c'est le perceptron multi-couche.

De 2000 à 2019, on a observé une augmentation tant de la puissance de calcul que de la taille des ensembles de données. Enfin, en 2010, l'apprentissage profond a émergé.



1.3.3 Définition perceptron monocouche

Le perceptron monocouche est le modèle le plus simple de réseau de neurones artificiels. Il consiste en une seule couche de neurones.

Le graphe de calcul du réseau neuronal monocouche implique plusieurs étapes et composants clés:

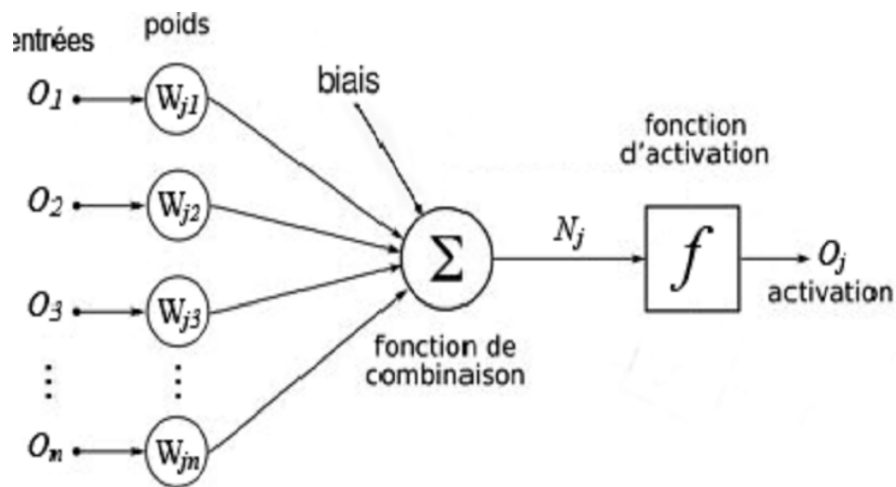


Figure 1.9: Le graphe de calcul du réseau neuronal monocouche

1. Couche d'Entrée: représente les caractéristiques d'entrée O_1, O_2, \dots, O_n
2. Somme Pondérée: Chaque caractéristique d'entrée est multipliée par son poids correspondant ($W_{j1}, W_{j2}, \dots, W_{jn}$) et puis calcule de la somme pondérée $Z = W_1X_1 + W_2X_2 + \dots + W_nX_n$
3. La fonction d'activation (la fonction de transfert): joue un rôle très important dans le comportement du neurone. Elle retourne une valeur représentative de l'activation du neurone, cette fonc-

tion a comme paramètre la somme pondérée des entrées ainsi que le seuil d'activation.

Exemple, La fonction sigmoïde est une fonction d'activation appliquée à la somme pondérée pour écraser la sortie entre 0 et 1 ,La fonction exponentielle...

4. La sortie: est le résultat de la fonction d'activation .

1.3.4 La fonction d'activation

Dans les réseaux neuronaux, la fonction d'activation produit des frontières de décision en sortie et est utilisée pour améliorer les performances du modèle.

La fonction d'activation est une expression mathématique qui décide si l'entrée doit passer à travers un neurone ou non en fonction de sa signification. Elle confère également une non-linéarité aux réseaux.

La non-linéarité est une des caractéristiques qui différencie les algorithmes de Deep Learning des algorithmes de Machine Learning traditionnels. Sans fonction d'activation, le réseau neuronal devient un simple modèle de régression linéaire.

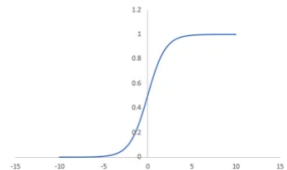
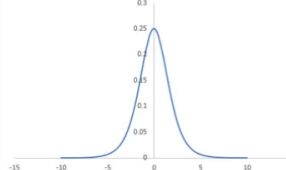
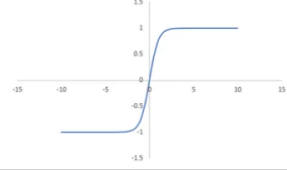
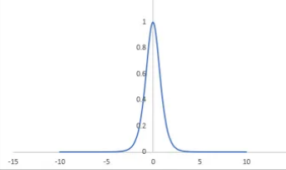
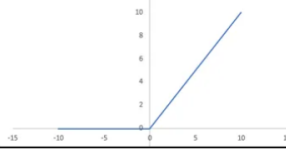
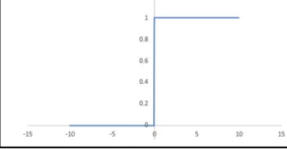
Activation function	$f(x)$	$f(x)$	$\frac{df(x)}{dx}$
sigmoid	$\frac{1}{1 + e^{-x}}$		
tanh	$\tanh(x)$		
ReLU	$\max(0, x)$		

Figure 1.10: Les fonctions d'activation courantes

Ce tableau montre trois fonctions d'activation courantes. Les graphiques de chaque fonction d'activation et de ses dérivées sont également présentés.

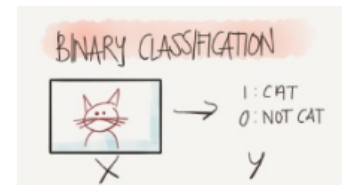
1. La fonction Rectified Linear Unit (ReLU) est la fonction d'activation la plus couramment utilisée en Deep Learning. Elle laisse passer les valeurs positives dans les couches suivantes et bloque les valeurs négatives. Ce filtre permet alors au modèle de se concentrer uniquement sur certaines caractéristiques des données, les autres étant éliminées.

2. La fonction Sigmoid est la fonction d'activation utilisée généralement en dernière couche d'un réseau de neurones construit pour effectuer une tâche de classification binaire. Elle donne une valeur entre 0 et 1 qui peut être interprétée comme une probabilité.
3. La fonction tanh permet d'appliquer une normalisation aux valeurs d'entrée. Elle peut également être utilisée au lieu de la fonction Sigmoid dans la dernière couche d'un modèle de classification binaire. Elle donne un résultat entre -1 et 1.

Le choix de la fonction d'activation dépend du problème que nous essayons de résoudre.

1.3.5 Régression logistique en tant que réseau neuronal

La régression logistique est un algorithme d'apprentissage automatique utilisé principalement pour les problèmes de classification binaire.



Son modèle mathématique combine linéairement les caractéristiques d'entrée pondérées par des coefficients.

$$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

Dans cette équation :

z : est la somme pondérée des caractéristiques d'entrée et des coefficients.

$w_0 = b$: est le terme de biais (intercept).

w_1, w_2, \dots, w_n : sont les coefficients associés à chaque caractéristique.

Cette somme pondérée z est ensuite utilisée comme argument pour la fonction sigmoïde, produisant la probabilité que la sortie soit de la classe positive. La fonction sigmoïde est définie comme suit :

$$h_{\theta}(x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

.



Figure 1.11: Logistic Regression

Une fois la fonction sigmoïde appliquée, la probabilité obtenue ($\sigma(z)$) peut être convertie en une prédiction de classe en utilisant un seuil. Un seuil couramment utilisé est 0,5. Si la probabilité est supérieure à ce seuil, la prédiction est souvent attribuée à la classe positive, sinon à la classe négative. Ce processus de seuillage (*thresholding*) permet de transformer les sorties continues de la régression logistique en prédictions binaires.

Alors, on peut considérer la régression logistique comme type spécifique de perceptron monocouche, avec une fonction d'activation particulière la fonction sigmoïde.

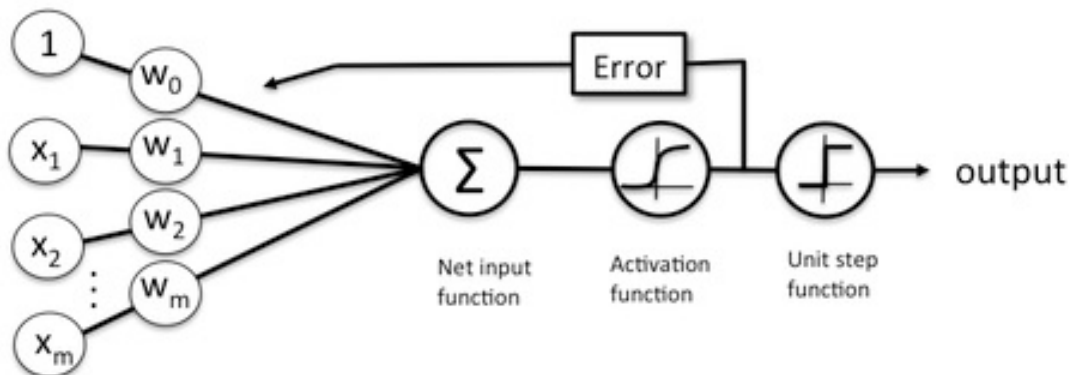


Figure 1.12: Régression logistique comme un réseau de neurones

1.4 Perceptron Multicouche

Les Perceptrons Multicouches (PMC) sont des réseaux de neurones pour lesquels les neurones sont organisés en couches successives.

Il revient essentiellement à prendre le réseau neuronal monocouche (la régression logistique) et à la répéter au moins deux fois.

En réseau neuronal monocouche, il y a la couche d'entrée et de sortie. Cependant, dans un réseau neuronal, il y a au moins une couche cachée entre la couche d'entrée et la couche de sortie.

La figure (1.11) donne l'exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie.

1. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du système. Elle ne contient aucun neurone.
2. Les couches cachées sont des couches de neurones. Dans l'exemple illustré, il y a 3 entrées, 3 neurones sur la première couche cachée, 4 neurones sur la deuxième et 2 neurones sur la couche de sortie.
3. Les sorties des neurones de la dernière couche correspondent toujours aux sorties du système.

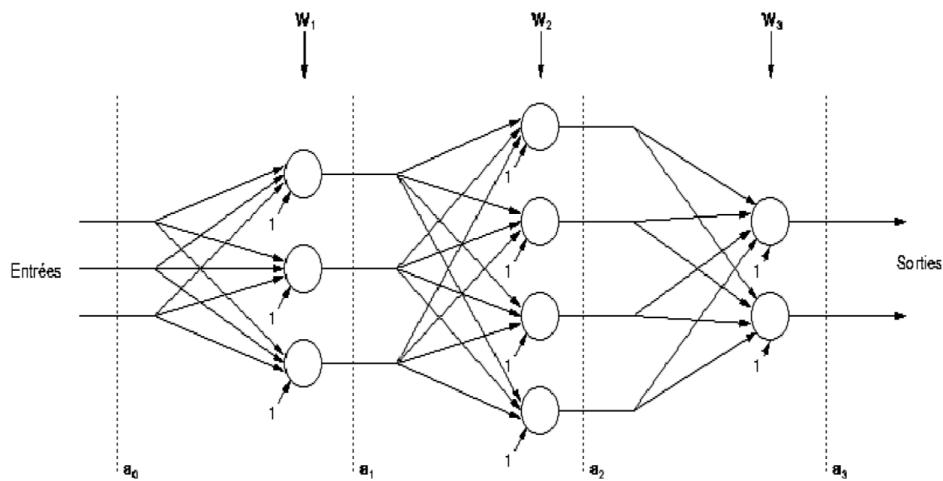


Figure 1.13: Le graphe de calcul du réseau neuronal multicouche

2 Les Réseaux de Neurones Artificiels

2.1 Introduction des Réseaux de Neurones Artificiels

Les Réseaux de Neurones Artificiels (RNA) constituent un élément fondamental du machine learning, s'inspirant des réseaux neuronaux biologiques. Les RNA, composés de neurones interconnectés, excellent dans l'apprentissage de motifs complexes à partir des données.

Les Composants Neuronaux sont:

1. **Neurones** : Traitent les entrées par des combinaisons pondérées et appliquent des fonctions d'activation pour produire des sorties.
2. **Structure du Réseau** : Organisée en couches - entrée, cachée, et sortie - facilitant l'apprentissage de représentations complexes.

2.1.1 Types de Réseaux de Neurones Artificiels

Les réseaux de neurones peuvent être classés en différents types, qui sont utilisés à des fins différentes. Voici les types les plus courants de réseaux de neurones que vous rencontrerez dans les cas d'utilisation les plus fréquents :

1. **Les réseaux de neurones à propagation avant**: sont ceux sur lesquels nous nous sommes principalement concentrés dans ce rapport appelées aussi Perceptrons Multi-Couches (PMC).

Ils sont constitués d'une couche en entrée, d'une ou de plusieurs couches cachées et d'une couche en sortie. Il est important de noter qu'ils sont en fait constitués de neurones sigmoïdes, et non de perceptrons, car la plupart des problèmes du monde réel sont non-linéaires.

Les données sont généralement introduites dans ces modèles pour les entraîner, et ils constituent la base de la vision par ordinateur, du traitement du langage naturel et d'autres réseaux de neurones.

2. **Réseaux Neuronaux Convolutifs (RNC)**: sont similaires aux réseaux à propagation avant, mais ils sont généralement utilisés pour la reconnaissance d'images, la reconnaissance de formes et/ou la vision par ordinateur. Ces réseaux exploitent les principes de l'algèbre linéaire, en particulier la multiplication des matrices, pour identifier les motifs dans une image.

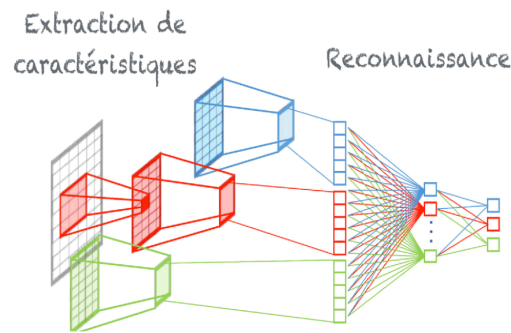


Figure 2.1: Réseaux Neuraux Convolutifs

3. **Les réseaux de neurones récurrents (RNN)** sont identifiés par leurs boucles de rétroaction.

Ces algorithmes d'apprentissage sont principalement exploités lorsqu'on utilise des données de séries chronologiques pour faire des prédictions sur des résultats futurs, comme les prédictions boursières ou les prévisions de ventes.

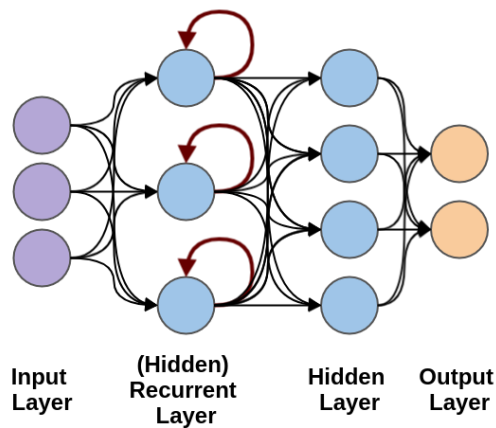


Figure 2.2: Réseaux Neuraux Récurrents

2.2 Relation entre réseau neurone artificiel (ANN) et réseau neurone biologique

2.2.1 Définition du neurone biologique

Les neurones sont les unités fonctionnelles de base du système nerveux, et ils génèrent des signaux électriques appelés potentiels d'action, ce qui leur permet de transmettre rapidement des informations sur de longues distances.

Presque tous les neurones ont trois fonctions de base essentielles au fonctionnement normal de toutes les cellules du corps. Il s'agit de :

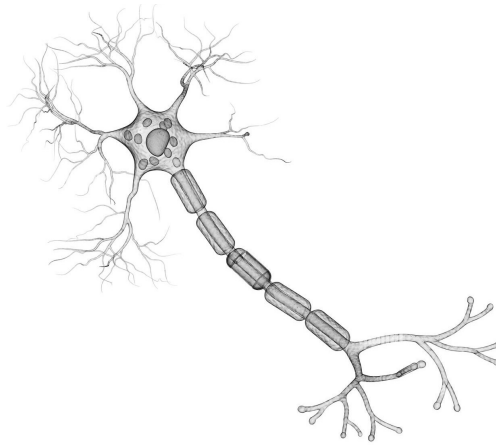


Figure 2.3: Le neurone biologique

1. Recevoir des signaux (ou des informations) de l'extérieur.
2. Traiter les signaux entrants et déterminer si l'information doit être transmise ou non.
3. Communiquer des signaux aux cellules cibles qui peuvent être d'autres neurones, des muscles ou des glandes.

Chaque neurone connecté à des milliers d'autres neurones forme un réseau neuronal. Ce réseau neuronal, appelé Réseaux Neuraux Biologiques (RNB), semble avoir des couches consécutives organisées de neurones dans la couche externe du cerveau.

2.2.2 Définition du neurone formel

Le neurone artificiel, également connu sous le nom de perceptron, est l'unité de base du réseau neuronal. En termes simples, c'est une fonction mathématique basée sur un modèle des neurones biologiques. On peut aussi le voir comme une porte logique simple avec des sorties binaires.

Chaque neurone artificiel a les fonctions principales suivantes :

1. Prend des entrées de la couche d'entrée.
2. Les pondère individuellement et les additionne.
3. Fait passer cette somme à travers une fonction non linéaire pour produire une sortie.

De nombreuses couches de perceptrons forment un Perceptron Multicouche (MLP). MLP se compose d'une couche d'entrée, d'une couche de sortie, et d'une ou plusieurs couches de TLU. Les couches supplémentaires de TLU sont appelées couches cachées. Si un ANN a de nombreuses couches cachées, on parle alors de Réseau de Neurones Profonds (DNN pour Deep Neural Network).

2.2.3 Relation entre réseau neurone artificiel (ANN) et réseau neurone biologique

Maintenant, vous pouvez voir que l'ANN imite d'une certaine manière le fonctionnement du BNN.

L'impulsion reçue par les dendrites dans le BNN correspond à la couche d'entrée dans l'ANN. L'impulsion passe à travers les axones dans le BNN comme le processus des couches d'entrée aux couches de sortie à travers les couches cachées dans l'ANN. Enfin, la transmission des terminaisons synaptiques vers les autres dendrites dans le BNN correspond à la couche de sortie dans l'ANN.

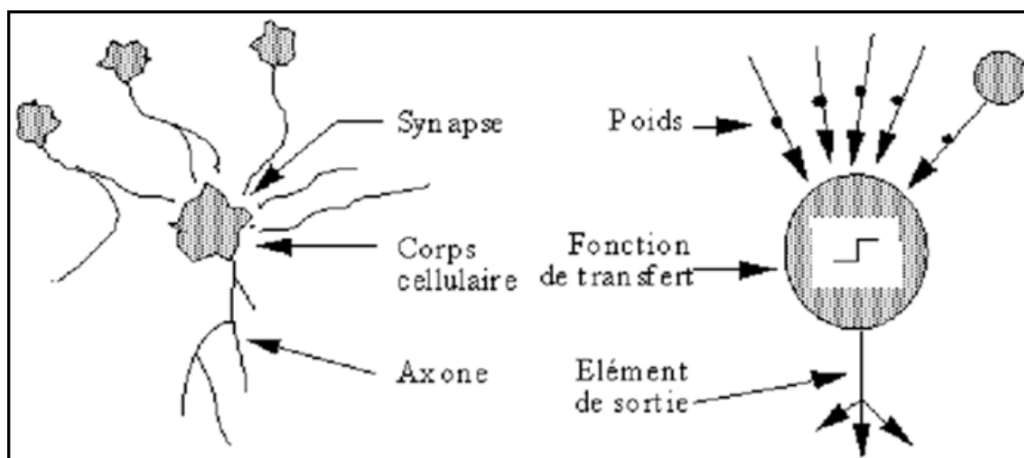


Figure 2.4: Comparaison entre le neurone biologique et le neurone formel.

Cependant, année après année, l'ANN est devenu beaucoup plus distinct du "vrai" réseau neuronal. Néanmoins, nous savons qu'il existe encore de nombreuses possibilités de nouvelles inventions qui peuvent être inspirées par la nature.

2.3 Processus du Forward Pass et Backpropagation

Le fonctionnement interne d'un réseau neuronal est orchestré à travers deux phases cruciales : la passe avant (forward pass) et la rétropropagation (backpropagation).

La forward pass, ou étape de propagation avant, représente le processus initial où le réseau évalue ses entrées pour générer des prédictions.

À la suite de cette phase, la rétropropagation entre en jeu, marquant une étape décisive dans laquelle le réseau ajuste ses paramètres internes pour minimiser les écarts entre les prédictions et les sorties réelles.

Explorons de manière approfondie ces deux aspects fondamentaux du fonctionnement des réseaux neuronaux : la façon dont la passe avant permet la génération de prédictions, et comment la rétropropagation joue un rôle essentiel dans l'ajustement itératif des paramètres pour améliorer la performance globale du réseau.

2.3.1 Forward pass ou Foward propagation:

1. **Entrée :** Les données sont introduites dans le réseau de neurones. Chaque entrée est associée à un poids.
2. **Propagation à travers les couches:**
 - Combinaison linéaire : Pour chaque neurone, les entrées sont pondérées par les poids associés, et la somme pondérée est calculée.
 - Activation : Une fonction d'activation (comme la fonction sigmoïde ou la fonction ReLU) est appliquée à la sortie de la combinaison linéaire. Cela introduit une non-linéarité dans le modèle.
3. **Sortie :** Ce processus est répété à travers toutes les couches jusqu'à atteindre la couche de sortie. La sortie de la dernière couche représente la prédiction du réseau.

2.3.2 Backward pass ou Backpropagation:

1. **Calcul du coût :**
 - Comparaison avec la vérité terrain : La sortie du réseau est comparée à la vérité terrain (les étiquettes réelles) à l'aide d'une fonction de coût (comme la perte quadratique ou la perte logistique)
2. **Rétropropagation de l'erreur :**
 - Calcul du gradient : On calcule la dérivée partielle du coût par rapport à chaque poids dans le réseau, en utilisant la règle de la chaîne. Cela mesure la sensibilité de la fonction de coût aux changements dans chaque poids.
 - Propagation inverse des gradients : Ces gradients sont alors propagés en sens inverse à travers le réseau.
3. **Mise à jour des poids :**
 - Optimisation : Les poids du réseau sont mis à jour en utilisant un algorithme d'optimisation tel que la descente de gradient. Les poids sont ajustés dans le sens qui réduit le coût global.
4. **Répétition :**
 - Les étapes du forward pass et du backpropagation sont répétées sur plusieurs itérations (epochs) jusqu'à ce que le modèle converge vers une solution.

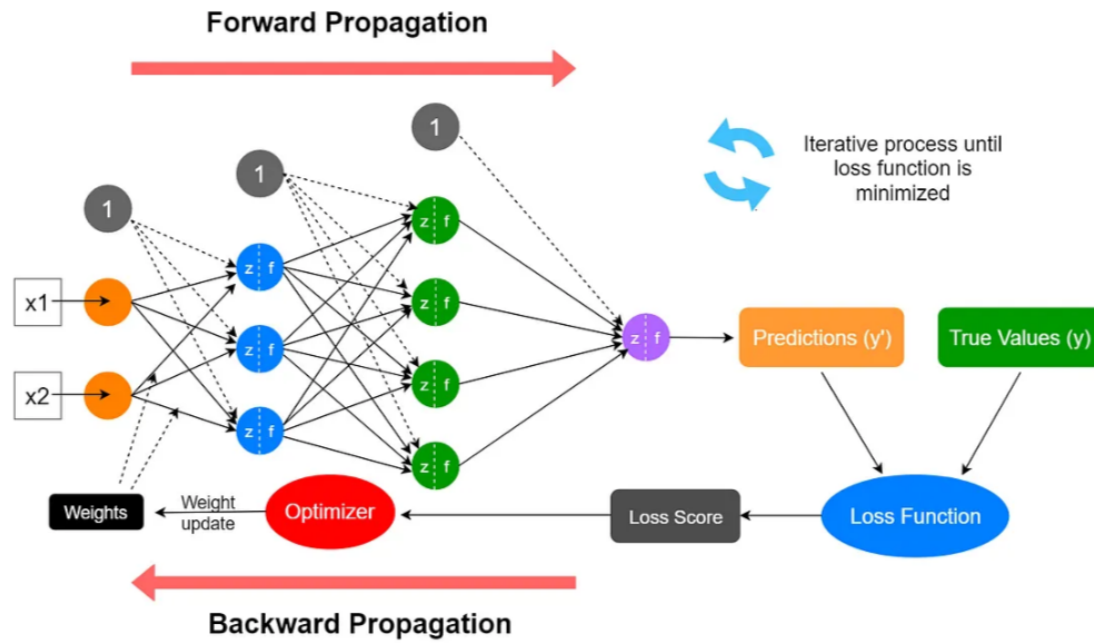


Figure 2.5: Forward Propagation et backward Propagation

Exemple de Régression Logistique:

La fonction coût de régression logistique $J(w,b)$:

$$J(w,b) = -\frac{1}{m} \sum_{i=1}^m [y_i \log(a_i) + (1 - y_i) \log(1 - a_i)]$$

Cette fonction est convexe ce qui peut faciliter sa optimisation. Elle peut être visualiser comme ci-dessous:

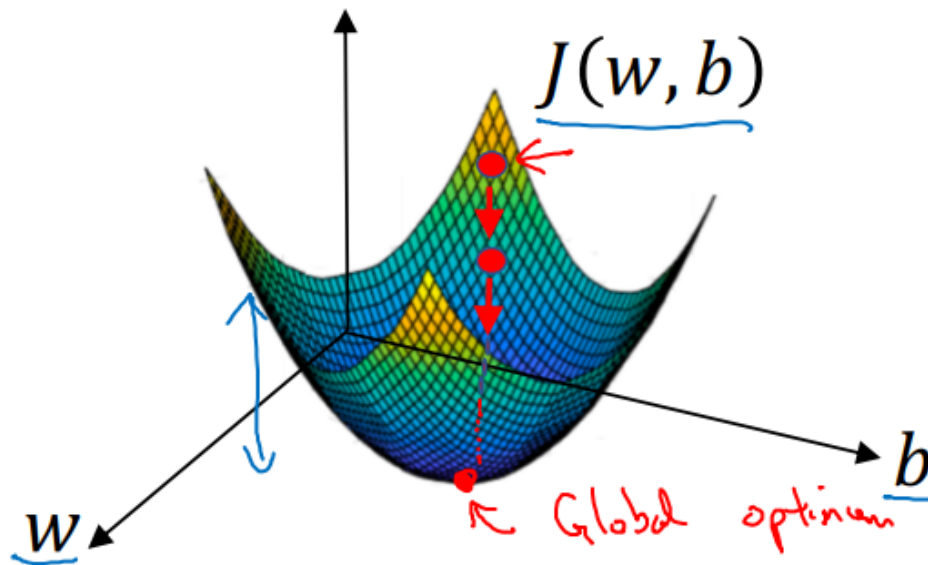


Figure 2.6: La fonction du coût du régression logistique

Pour trouver une bonne valeur pour les paramètres, nous devons initialiser w et b à une valeur initiale, généralement nous utilisons 0, mais des initialisations aléatoires fonctionnent également. Et comme c'est une fonction convexe, peu importe où vous l'initialisez, vous arriverez au même point ou approximativement au même point.

1. Forward Propagation:

- Les données d'entrées X sont introduites dans le réseau de neurones. Chaque caractéristique est associée à un poids w et un biais b .
- Pour chaque observation, le réseau effectue une combinaison linéaire des caractéristiques pondérées par les poids associés, puis ajoute le biais. Mathématiquement, la combinaison linéaire Z est calculée comme suit :

$$z = \sum_{i=1}^n w_i \cdot x_i + b$$

- Ensuite, la fonction sigmoïde est appliquée

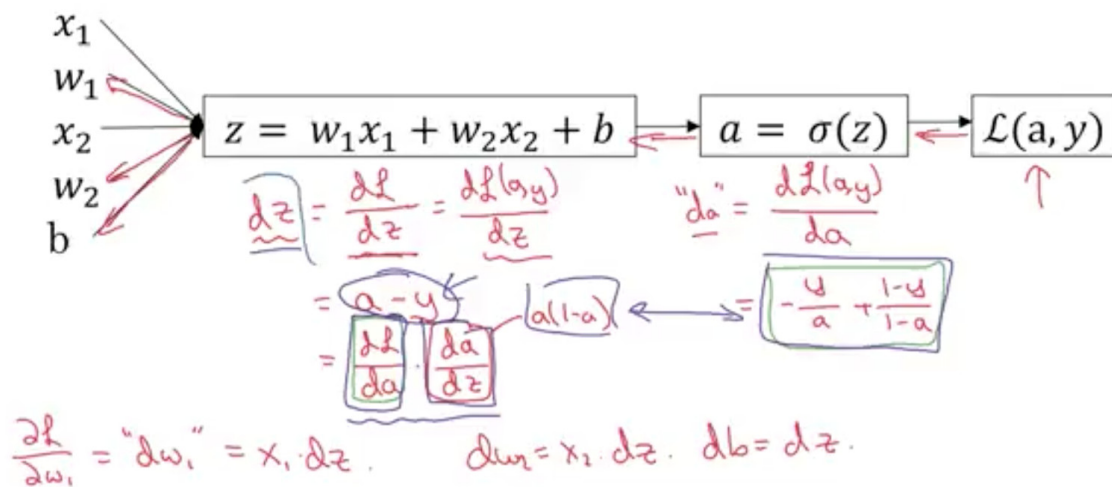
$$a = h_{\theta}(x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Et la sortie représente la probabilité estimée que l'observation appartienne à la classe positive.

2. Backward Propagation:

- Calcul de la fonction coût: Dans cette partie, nous cherchons les meilleurs estimateurs de w_i et b qui minimisent la fonction coût $J(w,b)$
- Calcul des gradients : Les gradients du coût ou dérivées partiels par rapport aux paramètres du réseau sont calculés pour chaque poids et biais en utilisant la règle de la chaîne.

Logistic regression derivatives



Andrew Ng

Figure 2.7: Calcul des gradients dans la backpropagation

- Mise à jour des paramètres : Les paramètres du réseau sont ajustés pour minimiser la perte en utilisant un algorithme d'optimisation tel que la descente de gradient.

Ce processus itératif de Forward Propagation et Backpropagation est répété sur plusieurs cycles jusqu'à ce que les paramètres convergent vers des valeurs qui minimisent la perte et permettent des prédictions précises.

Nous allons maintenant explorer, dans la partie suivante, les différents algorithmes d'optimisation qui complètent ce processus.

3 Optimisation dans les Artificial Neural Network

3.1 Descente de Gradient:

La descente de gradient (GD) est un algorithme itératif d'optimisation du premier ordre utilisé pour localiser un minimum ou maximum d'une fonction donnée.

Cette méthode est fréquemment employée en apprentissage machine (ML) et en apprentissage profond (DL) pour minimiser une fonction de coût ou de perte.

Cette méthode a été proposée bien avant l'ère des ordinateurs modernes par Augustin-Louis Cauchy en 1847. Depuis cette époque, il y a eu un développement significatif en informatique et en méthodes numériques.

L'algorithme de descente de gradient itère de manière répétée pour calculer le point suivant en utilisant le gradient à la position actuelle, le met à l'échelle (par un taux d'apprentissage), puis soustrait la valeur obtenue de la position actuelle pour effectuer un pas. La soustraction est effectuée car l'objectif est de minimiser la fonction (l'objectif serait d'ajouter). Mathématiquement, ce processus peut être formulé comme suit :

$$w = w - \alpha \frac{dJ}{dw}$$

Ici, α représente le taux d'apprentissage (learning rate).

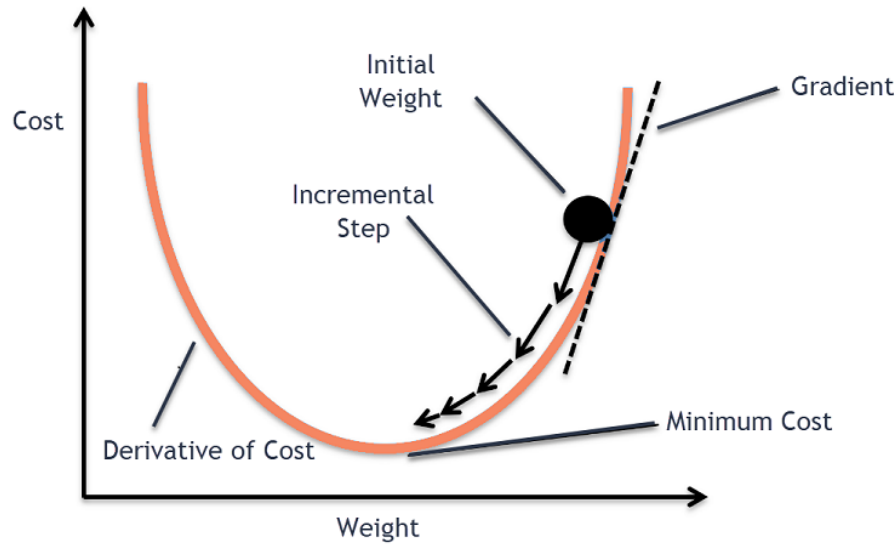


Figure 3.1: Gradien descent

Le choix du taux d'apprentissage est crucial. Un taux d'apprentissage plus petit conduit à une convergence plus lente et peut atteindre le nombre maximum d'itérations avant d'atteindre le point optimal. Un taux d'apprentissage trop élevé peut entraîner une non-convergence vers le point optimal (sauter autour). **Le choix du taux d'apprentissage dépend du problème spécifique et nécessite souvent une expérimentation pour trouver une valeur appropriée.**

En résumé, les étapes de la méthode de descente de gradient sont les suivantes :

1. Choisir un point de départ (initialisation)
2. Calculer le gradient à ce point
3. Effectuer un pas mis à l'échelle dans la direction opposée au gradient (objectif : minimiser)
4. répéter les points 2 et 3 jusqu'à ce que l'un des critères soit atteint :
 - Nombre maximal d'itérations atteint
 - La taille du pas est inférieure à la tolérance (en raison de la mise à l'échelle ou d'un faible gradient).

Le nombre maximal d'itérations est généralement défini avant l'entraînement pour éviter que l'algorithme ne s'exécute indéfiniment. Il représente le nombre maximum de fois que l'algorithme effectuera la mise à jour des paramètres. Certains problèmes peuvent converger rapidement, tandis que d'autres peuvent nécessiter plus d'itérations pour atteindre une solution optimale.

3.2 Optimiseurs Avancés

Nous avons examiné en détail les concepts mathématiques fondamentaux de l'optimiseur de base pour les réseaux neuronaux, à savoir la descente de gradient. En ce qui concerne les optimiseurs avancés, nous allons maintenant présenter leurs concepts ainsi que les avantages et les inconvénients de chacun, sans approfondir trop les détails mathématiques.

3.2.1 Descente de Gradient

Le Concept de Descente du gradient est de déterminer un vecteur directionnel d basé sur le gradient (vecteur des dérivées partielles), Mettre à jour les paramètres

$$w = w - d$$

et Répéter jusqu'à convergence.

Ca fonctionne car le gradient donne 0 si c'est un point critique.

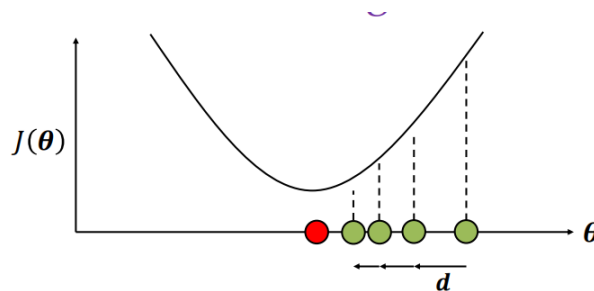


Figure 3.2: Gradient descent

L'inconvénients de la descente du gradient par batch est qu'il est trop sensible au minimum / maximum locaux pour être utilisable en pratique.

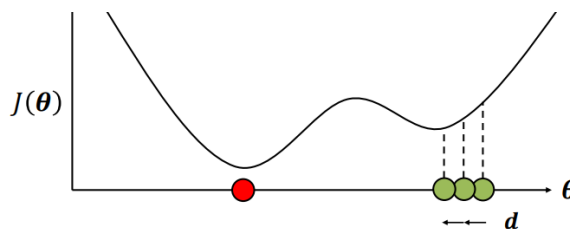


Figure 3.3: Inconvénient Gradient descent

De plus, il peut être lent sur de grands ensembles de données, car il nécessite le passage de l'ensemble du lot pour chaque mise à jour.

En réponse à ces inconvénients, une solution consiste à introduire la Descente du gradient stochastique (SGD), une variante de l'optimisation par descente de gradient.

3.2.2 la Descente du gradient stochastique (SGD)

La Descente de gradient stochastique (SGD) est une technique d'optimisation où les poids du modèle sont mis à jour après chaque exemple d'entraînement. Contrairement à la descente de gradient par batch qui utilise l'ensemble des données d'entraînement, la SGD utilise un seul exemple de taille m à la fois.

L'inconvénient de la Descente du gradient stochastique (SGD) est que La SGD est sujette à une variabilité significative dans les mises à jour des poids, car chaque exemple d'entraînement contribue à une mise à jour distincte. Cela peut entraîner une trajectoire de convergence moins stable par rapport à la descente de gradient par batch.

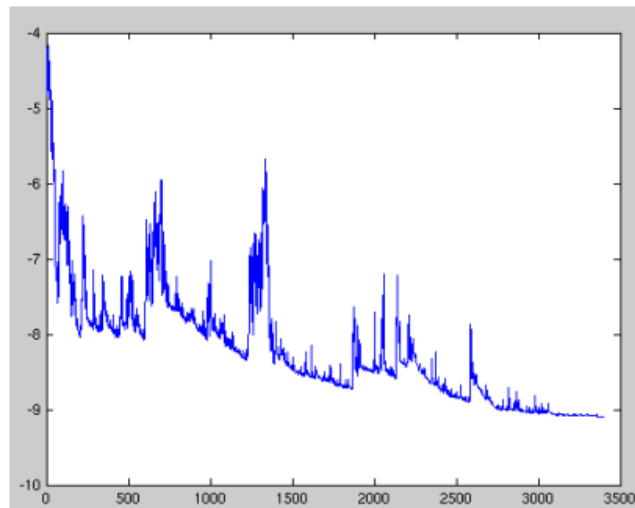


Figure 3.4: Inconvénient de la Descente du gradient stochastique

3.2.3 La Descente de gradient avec momentum

La Descente de gradient avec momentum est une amélioration de la Descente de gradient stochastique (SGD) qui vise à accélérer la convergence en introduisant un terme de momentum.

Le momentum est un coefficient qui pondère la contribution du gradient actuel par rapport à la mise à jour précédente des poids. Ce qui peut aider à atténuer les oscillations indésirables et à accélérer la convergence vers le minimum global.

C'est vrai que cette approche qu'elle peut survoler les min / max locaux. Mais, peut dépasser le min global

3.2.4 La Descente de gradient avec momentum Accéléré de Nesterov

Contrairement à la Descente de gradient avec momentum, l'Accéléré de Nesterov calcule le gradient non pas au point actuel des poids, mais à un point avancé dans la direction de l'inertie (momentum).

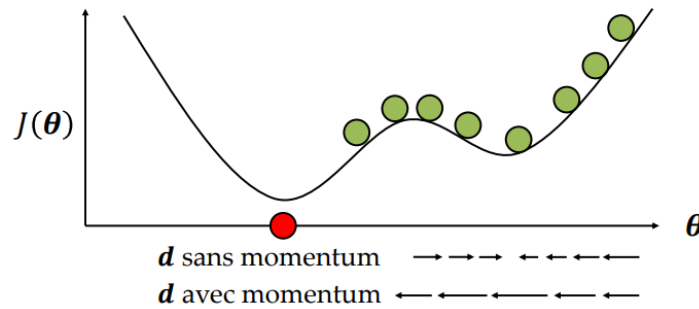


Figure 3.5: Descente du gradient sans momentum Vs Descente du gradient avec momentum

Cela permet d'anticiper le mouvement futur du gradient et d'ajuster la mise à jour en conséquence.

Cependant, il présente des inconvénients tels que la sensibilité aux hyperparamètres et la nécessité d'une adaptation fine. Son utilisation peut être justifiée dans des cas spécifiques où la convergence plus rapide est cruciale.

3.2.5 Adagrad (Adaptive Gradient Algorithm)

Adagrad (Adaptive Gradient Algorithm) est un algorithme d'optimisation qui adapte le taux d'apprentissage de chaque paramètre en fonction de la fréquence d'apparition des gradients.

Il favorise une mise à jour plus importante des paramètres associés à des gradients moins fréquents et une mise à jour moins importante pour les paramètres associés à des gradients fréquents. Cela permet à Adagrad de s'ajuster dynamiquement à la géométrie du problème d'optimisation.

Adagrad réduit le taux d'apprentissage au fil du temps en accumulant les carrés des gradients. Cela peut entraîner une convergence trop rapide pour certains paramètres, réduisant ainsi l'efficacité de l'apprentissage.

3.2.6 RMSprop (Root Mean Square Propagation)

La méthode d'optimisation RMSprop (Root Mean Square Propagation) est une variante de la descente du gradient qui adresse certains des inconvénients d'Adagrad en utilisant une moyenne mobile du carré des gradients pour normaliser les taux d'apprentissage.

Lors de son application, les paramètres et les taux d'apprentissage initiaux sont initialisés, suivis de la création d'un accumulateur de gradient.

À chaque itération, le gradient est calculé, l'accumulateur de gradient est mis à jour à l'aide d'une moyenne mobile du carré du gradient, et les paramètres sont ajustés avec un taux d'apprentissage adaptatif.

RMSprop offre une meilleure adaptation aux différentes échelles de gradients dans différentes dimensions, améliorant la convergence dans certains contextes, malgré le défi potentiel de la diminution

du taux d'apprentissage (Comme Adagrad).

3.2.7 Adam (Adaptive Moment Estimation)

La méthode d'optimisation Adam (Adaptive Moment Estimation) est une approche avancée qui combine des concepts de momentum et de RMSprop pour maintenir à la fois une moyenne mobile des gradients et de leurs carrés.

Dans le processus d'Adam, les paramètres sont initialement configurés avec des taux d'apprentissage initiaux, des facteurs de décroissance, et une petite constante pour la stabilité numérique.

Deux moments, du premier ordre et du second ordre, sont ensuite mis à jour à chaque itération en utilisant des moyennes mobiles exponentielles. Ces moments sont corrigés pour compenser leur initialisation à zéro. Enfin, les paramètres du modèle sont ajustés en utilisant les moments corrigés et un taux d'apprentissage adaptatif.

Adam s'adapte ainsi dynamiquement aux caractéristiques changeantes de la fonction de coût, ce qui conduit souvent à une convergence rapide et à de bonnes performances dans la pratique. Cependant, le choix des hyperparamètres reste crucial, et certaines considérations, comme la sensibilité à la taille des mini-lots, doivent être prises en compte pour optimiser son efficacité dans des scénarios spécifiques.

3.3 Sélection d'Optimiseurs en Fonction du Problème

Le choix de l'optimiseur dépend souvent des caractéristiques du problème spécifique et de la nature des données. Voici quelques considérations courantes lors de la sélection de l'optimiseur :

1. **Ensembles de données peu volumineux :** SGD avec Momentum peut être préféré pour des ensembles de données plus petits, offrant un équilibre entre stabilité de convergence et efficacité computationnelle..
2. **Grandes quantités de données :** Pour des ensembles de données massifs, Adam est souvent un choix efficace en raison de son adaptation dynamique des taux d'apprentissage.
3. **Problèmes de classification d'images (CNN) :** Adam se comporte généralement bien sur des tâches de classification d'images utilisant des architectures de réseaux de neurones convolutifs (CNN).
4. **Réseaux récurrents (RNN) :** Les méthodes d'adaptation des taux d'apprentissage, comme Adam et RMSprop, sont souvent utilisées pour l'entraînement de réseaux récurrents (RNN) en raison de la nature séquentielle des données.

4 Conclusion

Nous avons exploré les bases du deep learning, allant des réseaux de neurones au perceptron multicouche, en passant par la backpropagation. Nous avons également abordé l'optimisation avec la descente de gradient et les optimiseurs avancés. Les exemples pratiques ont illustré la pertinence du deep learning, notamment dans la reconnaissance d'images. En conclusion, cette introduction offre un aperçu clair du potentiel du deep learning dans divers domaines d'application.