Université Ibn Tofail
Faculté des Sciences Kénitra

Département d'Informatique
Master Informatique et Intelligence Artificielle

Rapport de

Mini-Projet

Introduction à l'intelligence artificielle

Modèles de Réseaux de Neurones (non supervisés) : Auto-Encodeur & Courbes de validation

Réaliser par :
Oussama LAHLIMI

Encadré par :
Mr. Ali CHOUKRI

Devant le jury :

Mr. : Ali CHOUKRI
Mme. : Salma AZZOUZI

Année universitaire : 2022/2023

Remerciements

“

Premièrement et avant tout je vous remercie Dieu tout puissant qui je a aidé, et je a donné la patience et le courage pour la réalisation de ce Mini-Projet. je vous remercie sincèrement Mr. Ali CHOUKRI notre encadrant Universitaire et mon Prof de module Introduction à l'intelligence artificielle, qui s'est toujours montré disponible tout au long de la réalisation de ce Mini-Projet, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu je consacrer, et sans lui, ce Mini-projet n'aurait jamais vu le jour

”

- Oussama Lahlimi

Résumé

Le Mini-projet sur les Modèles de Réseaux de Neurones non supervisés, en particulier l'Auto-Encodeur, consiste à explorer les concepts fondamentaux des réseaux de neurones et à comprendre le fonctionnement de l'Auto-Encodeur.

Le Mini-projet comprend plusieurs étapes, notamment la compréhension des concepts de base des réseaux de neurones et de l'Auto-Encodeur, la collecte et la préparation des données d'entraînement, la mise en place de l'architecture de l'Auto-Encodeur.

Le but ultime de ce projet est de fournir une compréhension pratique et approfondie de l'Auto-Encodeur en tant que modèle de réseau de neurones non supervisé et de ses applications potentielles.

Le Mini-projet sur les courbes de validation vise à explorer l'utilisation de ces courbes pour évaluer la performance des modèles d'apprentissage automatique. Les courbes de validation, également appelées courbes d'apprentissage, sont des graphiques qui montrent comment la performance d'un modèle varie en fonction de la taille de l'ensemble d'apprentissage.

Glossaire : Réseaux de Neurones, , ACP, Auto-Encodeur, Courbes de validation, Encodeur, courbes d'apprentissage, Décodeur.

Contents

List of Figures

liste des abréviations

ACP	*L'Analyse en Composantes Principales.*
AE	*Autoencodeur.*
SVM	*Support Vector Machine.*
AI	*artificial intelligence.*
ANN	*Artificial Neural Networks.*
ReLU	*Unité de Rectification Linéaire.*

Introduction Générale

Contexte

Les modèles de réseaux neuronaux non supervisés sont des algorithmes d'apprentissage automatique qui peuvent apprendre des caractéristiques intrinsèques des données sans étiquettes de sortie explicites. L'un des modèles les plus courants de réseaux neuronaux non supervisés est l'Autoencodeur, qui consiste en un réseau de neurones qui apprend à reconstruire ses entrées en compressant les informations utiles dans une représentation latente de dimension inférieure.

Les Autoencodeurs sont souvent utilisés pour la compression de données, la détection d'anomalies, la génération de données et la réduction de dimensionnalité. Ils peuvent également être utilisés comme étape préalable à des tâches supervisées, comme la classification ou la régression.

Les courbes de validation sont utilisées pour évaluer la performance des modèles de réseaux neuronaux et d'autres algorithmes d'apprentissage automatique. Elles consistent à tracer les scores de performance (par exemple, l'exactitude ou la perte) en fonction de divers paramètres, tels que le nombre d'itérations d'apprentissage, la taille du lot ou les hyperparamètres du modèle.

Les courbes de validation permettent de déterminer si un modèle est en surapprentissage (overfitting) ou en sous-apprentissage (underfitting), et peuvent aider à identifier les meilleurs paramètres pour un modèle donné. Elles sont souvent utilisées en conjonction avec des techniques de validation croisée pour évaluer la performance d'un modèle de manière fiable.

Organisation du travail

- **Modèles de Réseaux de Neurones (non supervisés) : Auto-Encodeur & Courbes de validation**
- **Conclusion et perspectives**

Modèles de Réseaux de Neurones (non supervisés) : Auto-Encodeur
Courbes de validation :

0.1 Réseaux de Neurones

0.1.1 Définition

Un Réseau de Neurones artificiel peut être défini comme étant un modèle dont la conception est inspirée du fonctionnement du neurone biologique. Ce dernier contient trois composantes essentielles à savoir (Voir figure 1.1) :

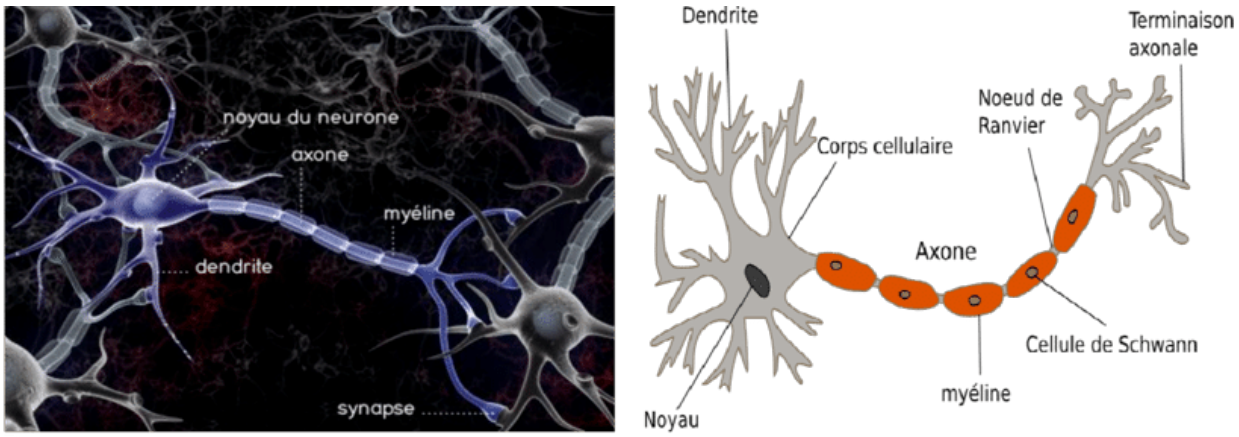

Figure 1: Neurone biologique.[1]

- **les Dendrites :** elles constituent les entrées du neurone et servent à acheminer vers le corps du neurone des signaux électriques en provenance d'autres neurones.

- **le Corps:** cellulaire accumule des charges électriques. Par la suite, le neurone peut être suffisamment excité (i.e la charge électrique dépasse un certain seuil) à tel Auto-Encoder qu'il engendre un potentiel électrique se propageant à travers son axone afin exciter d'autres neurones.

- **l'Axone:** elle représente la sortie du neurone vers d'autres neurones. Le Auto-Encoder de contact entre l'axone d'un neurone et le dendrite d'un autre neurone s'appelle le synapse.

Parallèlement au neurone biologique, tel qu'illustré par la figure 1.2, un neurone artificiel reçoit sous forme vectorielle des entrées $x_1, x_2, x_3, ..., x_p$, puis effectue une combinaison affine de ses entrées moins le seuil d'activation du neurone ou biais
$b(w_{i,1}\, x1 + w_{i,2}\, x2 + ... + w_{i,p}\, xp - b)$. Ensuite une fonction d'activation f est appliquée sur cette sortie :
$f(w_{i,1}\, x1 + w_{i,2}\, x2 + ... + w_{i,p}\, xp - b) = f(\sum_{i=1}^{p} w_{i,p}\, x_i - b)$ afin de créer une sortie y.Les paramètres $w_{i,1}\,, w_{i,2}\,, ..., w_{i,p}$ et b représentent respectivement les poids et le biais du neurone

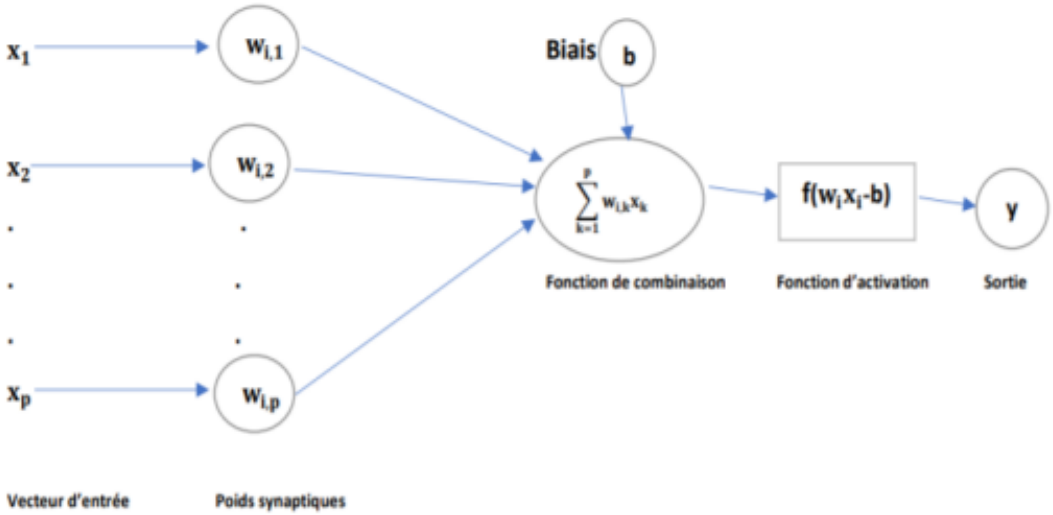

Figure 2: Neurone Artificiel.

Un ensemble de neurones reliés en réseau forme ce qu'on appelle un réseau de neurones. Les poids et le biais sont estimés lors de la phase d'apprentissage du réseau de neurones. Il existe deux types d'apprentissage des réseaux de neurones : l'apprentissage supervisé et l'apprentissage non supervisé . L'apprentissage supervisé est caractérisé par la présence d'un professeur ayant des connaissances sur l'environnement dans lequel évolue le réseau. Cependant l'apprentissage non supervisé s'effectue sur des données dont on ne connaît pas les étiquettes des observations. Il est caractérisé par l'absence de professeur. Il s'agira donc de regrouper les individus statistiques présentant des caractéristiques communes.

0.1.2 Fonction d'activation

Plusieurs fonctions d'activation sont utilisées dans les réseaux de neurones artificiels. Parmi elles, nous pouvons citer entre autres :
— la fonction Seuil (Heaviside):

$$g(x) = 1_{[0,+\infty]} \tag{1}$$

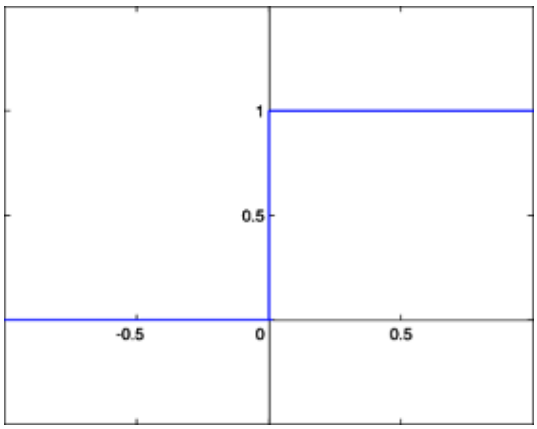

Figure 3: Fonction Seuil (Heaviside)

— la fonction ReLU (Unité de Rectification Linéaire) :

$$f(x) = max(0, x), x \in \mathbb{R} \tag{2}$$

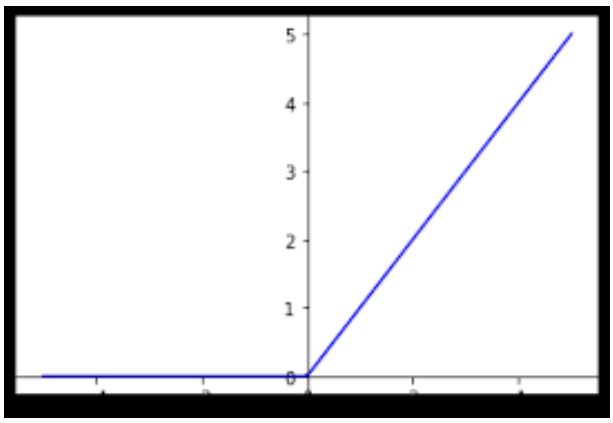

Figure 4: la fonction ReLU

— la fonction Sigmoïde :

$$g(x) = \frac{1}{1 + e^{-x}}, x \in \mathbb{R} \tag{3}$$

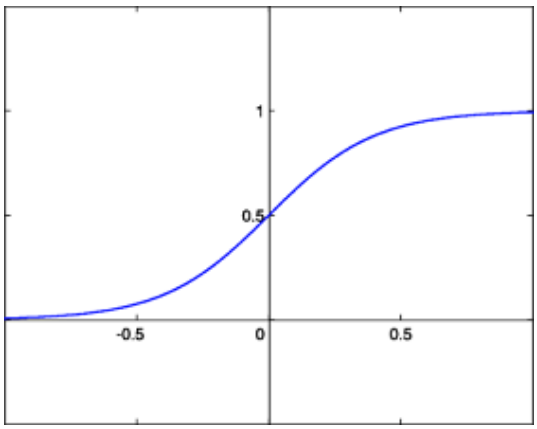

Figure 5: Fonction Sigmoïde

0.1.3 Descente du Gradient

L'Entrainement d'un réseau de neurones est en général effectué par la méthode de la descente du gradient. Tel qu'illustré par la figure 6, la descente du gradient cherche à perfectionner le réseau en répétant une mise à jour des poids, visant à minimiser la fonction objectif. La fonction objectif est en général l'erreur quadratique moyenne :

$$E = \sum_{i=1}^{n} \zeta(x_i, y_i) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2 \tag{4}$$

ζ étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les sorties.

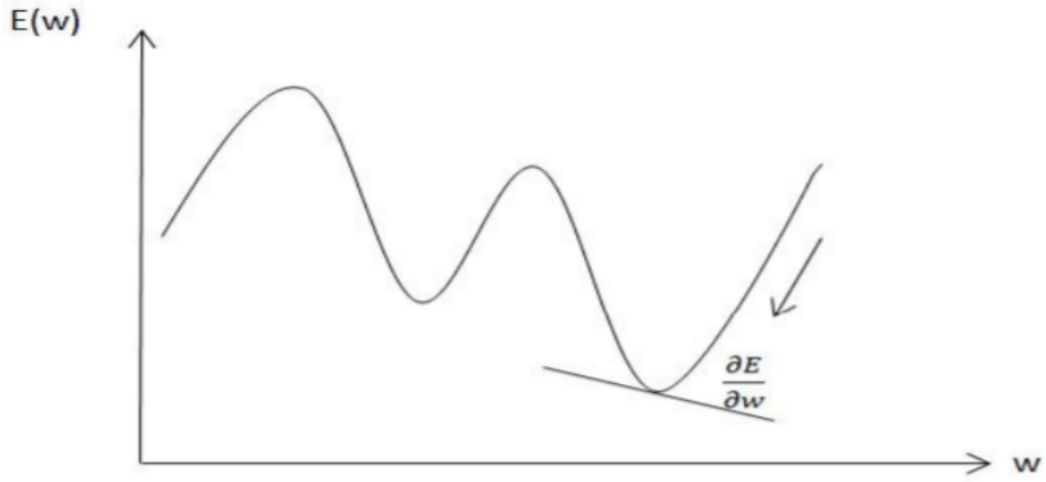

Figure 6: La Descente du Gradient suivant W

Plus précisément, les étapes de l'algorithme de la descente du gradient sont les suivantes

1. Phase d'initialisation : on choisit aléatoirement la matrice des poids w ;
2. Avant qu'un processus d'apprentissage est effectué, on présente les données d'entrainement x_i au réseau par propagation pour obtenir les sorties y_i
3. On calcule la fonction objectif entre les x_i et y_i puis on la minimise par rapport aux poids w en résolvant l'équation $\frac{\partial E}{\partial w} = 0$
4. On effectue la rétro-propagation du gradient de la fonction objectif. Cette dernière consiste à la mise à jour des poids des neurones à partir de la couche de sortie vers la couche d'entrée afin de minimiser la fonction objectif.
5. On fait une itération de la mise à jour des poids jusqu'à ce que la fonction objectif convergence vers un minimum global.
   Par abus de langage, le terme rétro-propagation du gradient est aussi utilisé pour faire référence à la fois à l'algorithme de descente du gradient et à la rétro-propagation du gradient.

0.2 Auto-Encodeur

0.2.1 Définition

Un Auto-Encodeur est un réseau de neurones artificiels dans lequel la couche d'entrée a le même nombre de neurones que celle de la sortie. Il est utilisé pour l'apprentissage non supervisé et vise à reconstruire l'entrée avec le minimum d'erreur possible. En d'autres termes, un auto-encodeur apprend une approximation de la fonction identité. Comme pour l'ACP, une des principales applications des Auto-Encodeurs est la réduction de dimensionnalité.

0.2.2 Quelques exemples d'algorithmes de Réseaux de Neurones non Supervisés

1. Auto-encodeurs : comme je l'ai mentionné précédemment, les auto-encodeurs sont des réseaux de neurones non supervisés qui apprennent à reconstruire des données d'entrée en utilisant une représentation latente de dimension inférieure.

2. Réseaux de neurones à convolution (CNN) non supervisés : ces réseaux de neurones sont souvent utilisés pour la reconnaissance d'images et peuvent apprendre des caractéristiques visuelles à partir des données sans étiquettes.

3. Réseaux de neurones récurrents (RNN) non supervisés : ces réseaux de neurones sont souvent utilisés pour l'analyse de séquences de données, telles que des séquences de texte ou de musique. Ils peuvent être entraînés sans étiquettes pour découvrir des modèles dans les données.

4. Réseaux de neurones Boltzmann : ces réseaux de neurones sont souvent utilisés pour la modélisation de données probabilistes et peuvent être utilisés pour la génération de données ou la reconnaissance de motifs dans les données.

Il convient de noter qu'il existe également de nombreux algorithmes de réseaux de neurones supervisés, tels que les réseaux de neurones à propagation avant, les réseaux de neurones à mémoire à long terme (LSTM), les réseaux de neurones adverses génératifs (GAN) et bien d'autres encore.

0.2.3 Pourquoi ce sujet ? À quoi sert ce sujet ?

- Les Auto-Encodeurs sont souvent utilisés pour des tâches telles que la réduction de la dimensionnalité des données, la compression de données, la restauration d'images, la génération de données, et la détection d'anomalies.

0.2.4 Le choix des Auto-Encodeurs

- Il est possible que vous ayez choisi un Auto-Encodeur pour votre tâche particulière en raison de ses avantages spécifiques par rapport à d'autres algorithmes d'apprentissage automatique. Par exemple, vous pouvez utiliser un Auto-Encodeur si vous avez un grand ensemble de données avec de nombreuses fonctionnalités et que vous souhaitez réduire la dimensionnalité des données pour une analyse plus efficace. En revanche, d'autres algorithmes peuvent être plus appropriés pour des tâches spécifiques telles que la classification ou la prédiction.

- L'Auto-Encodeur est généralement utilisé pour la réduction de la dimensionnalité, la compression de données, la restauration d'images et la génération de données

0.2.5 Avantages à utiliser un Auto-Encodeur

1. Réduction de la dimensionnalité : Les Auto-Encodeurs peuvent être utilisés pour réduire la dimensionnalité des données d'entrée en apprenant une représentation compressée de l'entrée. Cela permet de réduire la complexité des données, d'améliorer l'efficacité du stockage et de l'apprentissage, et de faciliter l'exploration des données.

2. Détection d'anomalies : Les Auto-Encodeurs peuvent être utilisés pour détecter les anomalies dans les données d'entrée en apprenant une représentation des données normales et en comparant les entrées réelles avec cette représentation. Les anomalies peuvent être identifiées comme des données qui ne correspondent pas à la représentation normale.

3. Prétraitement des données : Les Auto-Encodeurs peuvent être utilisés pour effectuer un prétraitement des données en apprenant une représentation des données d'entrée qui est plus facile à traiter pour d'autres algorithmes de réseau de neurones. Cela peut améliorer l'efficacité et la précision des modèles de réseau de neurones qui utilisent ces données en entrée.

4. Génération de données : Les Auto-Encodeurs peuvent être utilisés pour générer de nouvelles données en échantillonnant à partir de la distribution latente apprise. Cela peut être utile pour la génération de données synthétiques à des fins de test et d'évaluation.

5. Adaptation non supervisée : Les Auto-Encodeurs peuvent être utilisés pour adapter un modèle de réseau de neurones non supervisé à de nouvelles données sans nécessiter de labels de données supervisées. Cela permet d'adapter le modèle à des données spécifiques sans nécessiter de nouvelles annotations de données supervisées.

0.2.6 Limites courantes des Auto-Encodeurs

1. Perte d'information : Bien que les Auto-Encodeurs soient utilisés pour la compression de données, ils peuvent perdre de l'information lors de la reconstruction des données. Par conséquent, il est important de prendre en compte ce facteur lors de l'utilisation des Auto-Encodeurs.

2. Sur-apprentissage : Les Auto-Encodeurs peuvent également souffrir de sur-apprentissage, ce qui signifie qu'ils peuvent trop bien s'adapter aux données d'apprentissage et avoir du mal à généraliser pour de nouvelles données.

3. Complexité des données : Les Auto-Encodeurs peuvent avoir du mal à traiter des données complexes, telles que des images ou des vidéos en haute résolution, car ils nécessitent souvent des réseaux de neurones profonds avec un grand nombre de paramètres, ce qui peut entraîner des problèmes de temps de calcul et de mémoire.

4. Besoin de données d'apprentissage : Les Auto-Encodeurs nécessitent des données d'apprentissage suffisamment grandes et représentatives pour apprendre des représentations significatives. Par conséquent, si les données sont limitées, les performances des Auto-Encodeurs peuvent être compromises.

5. Hyperparamètres : Les performances des Auto-Encodeurs dépendent fortement des choix des hyperparamètres tels que la taille de l'encodeur et du décodeur, la taille de l'espace latent, le taux d'apprentissage, etc. Trouver les meilleurs hyperparamètres peut nécessiter des essais et des erreurs, ce qui peut être chronophage.

0.2.7 Étapes de création d'un Auto-Encodeur

- Préparation des données : Les données doivent être préparées pour l'entraînement de l'autoencodeur. Cela peut inclure la normalisation des données, la mise en place de séries chronologiques, la suppression des données manquantes et la conversion des données dans un format utilisable pour l'apprentissage automatique.

- Définition de l'architecture de l'autoencodeur : L'architecture de l'autoencodeur doit être définie. Cela inclut la définition du nombre de couches, du nombre de neurones par couche, du type de couche (dense, convolutionnelle, etc.) et de la fonction d'activation.

- Définition de la fonction de perte : La fonction de perte doit être définie pour l'autoencodeur. Dans la plupart des cas, il s'agit de la différence entre les données d'entrée et de sortie. L'objectif est de minimiser la perte pour reconstruire les données d'entrée avec une précision élevée.

- Entraînement de l'autoencodeur : L'autoencodeur est entraîné sur les données préparées à l'aide d'un algorithme d'optimisation comme la descente de gradient stochastique. Le but est de minimiser la fonction de perte sur les données d'entraînement.

- Validation et évaluation de l'autoencodeur : L'autoencodeur est validé sur un ensemble de données de validation distinct de l'ensemble de données d'entraînement. Cela permet de s'assurer que le modèle généralise bien et ne surajuste pas les données d'entraînement. Ensuite, l'autoencodeur est évalué sur un ensemble de données de test pour mesurer sa performance.

- Utilisation de l'autoencodeur : Une fois que l'autoencodeur est entraîné, il peut être utilisé pour encoder les données dans un espace de représentation de dimension réduite. Cela peut être utile pour la visualisation de données, la détection d'anomalies, la réduction de dimensionnalité et d'autres tâches liées à l'apprentissage automatique.

0.3 Architecture d'un Auto-Encodeur

- Un Auto-Encoder a une architecture très spécifique, car les couches cachées sont plus petites que les couches d'entrée. On appelle ce type d'architecture une architecture « bottleneck ». On peut décomposer un auto encodeur en deux parties à gauche et à droite de ce « bottleneck ».

- La partie gauche s'appelle l'Encodeur. L'encodeur transforme l'entrée en une représentation dans un espace de dimension plus faible appelé espace latent. L'encodeur compresse donc l'entrée dans une représentation moins coûteuse.

- La partie droite est appelée Décodeur, car elle doit reconstruire à l'aide de la représentation latente de l'entrée, une sortie la plus fidèle à l'entrée.

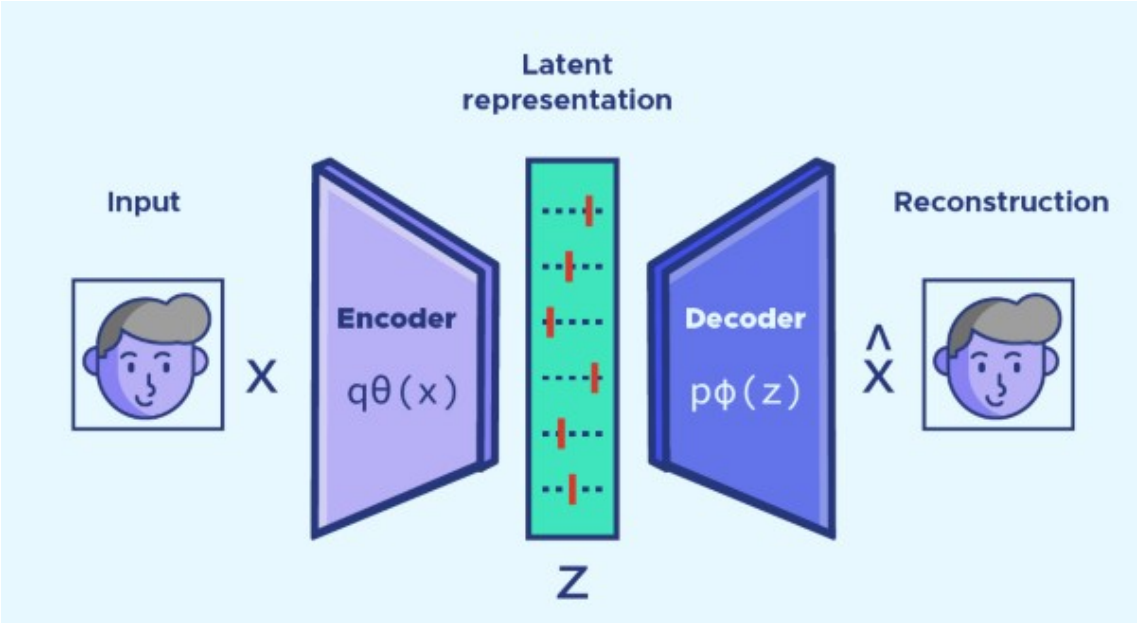

Figure 7: Schéma inspiré d'un cours de Fabien Moutarde - Mines de Paris (2021)

- Lors de l'apprentissage, l'Auto-Encodeur va donc apprendre à chercher à garder le plus d'information possible entre l'entrée et l'espace latent, afin que le décodeur ait les informations les plus essentielles pour reconstruire l'image. Ainsi, l'encodeur apprend les composantes les plus importantes d'une entrée pour avoir la meilleure compression possible.

- À première vue, cela ressemble fortement à l'analyse en composantes principales (ACP ou PCA en anglais). Cependant, les auto encodeurs permettent d'ajouter des non-linéarités grâce aux fonctions d'activation et à leur structure de réseau de neurones, ce qui distingue un auto encodeur simple d'une ACP.

0.3.1 Types d'Aut-Encodeur

Nous pouvons distinguer deux types d'Auto-Encodeurs selon la contrainte supplémentaire imposée pour limiter la capacité de représentation de l'Auto-Encodeur lors de l'optimisation de la fonction objectif. Nous avons, d'une part, les Auto-Encodeurs « undercomplete » qui limitent la dimension du code latent et, d'autre part, les Auto-Encodeurs « overcomplete ». Ces derniers ajoutent un terme de régulation dans la fonction objectif.
En fait, il existe deux types d'Auto-Encodeur :

1. les Under-Complete ont ceux dont les unités centrales sont en plus petit nombre que les unités d'entrée ; ils effectuent une réduction de dimension, comme vu précédemment.

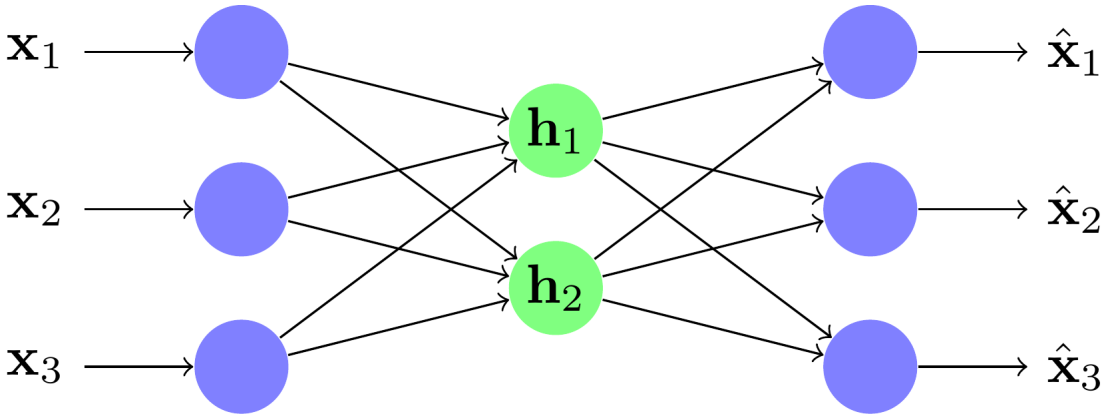

Figure 8: Un exemple de réseau Under-Complete

2. les Over-Complete sont ceux dont les unités centrales sont en plus grand nombre que les unités d'entrée ; ils cherchent, eux, une meilleure représentation des données pour un traitement ultérieur (cela correspond à la projection dans une plus grande dimension des SVM) .

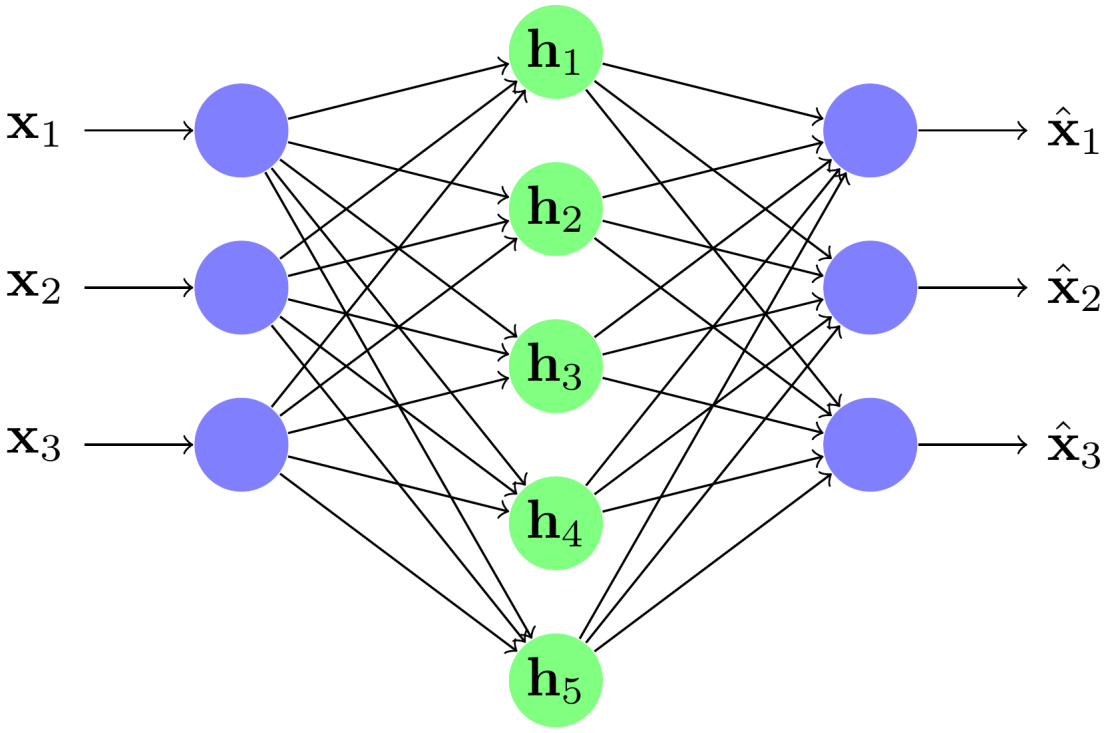

Figure 9: Un exemple de réseau Over-Complete

0.3.2 Architecture d'un Auto-Encodeur de Base

l'Architecture d'un Auto-Encodeur de base. Comme précédemment, nous partons du bas avec l'entrée x qui est soumise à un encodeur (transformation affine définie par w_h suivie

d'un écrasement). Il en résulte la couche cachée intermédiaire h Celle-ci est soumise au décodeur (une autre transformation affine définie par w_x, suivie d'un autre écrasement). Cela produit la sortie $\hat{x}$ qui est la prédiction reconstruction de l'entrée par notre modèle.

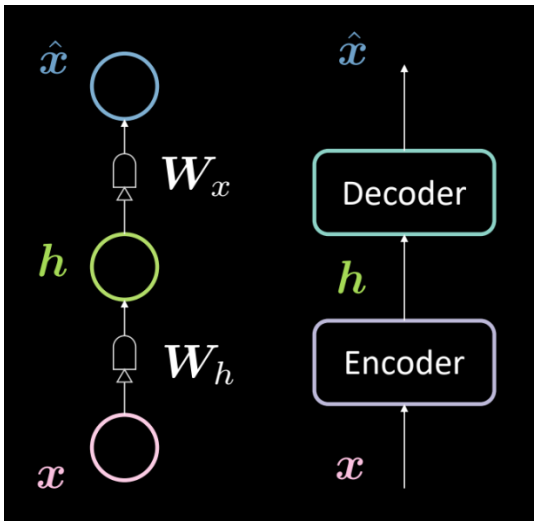

Figure 10: Architecture d'un Auto-encodeur de base

Nous pouvons représenter mathématiquement le réseau ci-dessus en utilisant les équations suivantes :

$$h = f(w_h x + b_h) \tag{5}$$

$$\hat{x} = g(w_x h + b_x) \tag{6}$$

Nous précisons également les dimensions suivantes :

$$x, \hat{x} \in \mathbb{R}^n$$
$$\mathrm{h} \in \mathbb{R}^d$$
$$\mathrm{w}_h \in \mathbb{R}^{d*n}$$
$$\mathrm{w}_x \in \mathbb{R}^{n*d}$$

Note

Pour représenter l'ACP, nous pouvons avoir des poids serrés (ou des poids liés) définis par le symbole $w_x = w_h^T$

0.3.3 Détails mathématique de Architecture de Auto-Encodeur

On peut décomposer un Auto-Encodeur en deux parties, à savoir un Encodeur,f, suivi d'un décodeur,h. L'encodeur permet de calculer le code $z_j = f(x_i)$ pour chaque échantillon d'apprentissage en entrée(x_{ij}), avec i allant de 1 jusqu'à n, n étant le nombre de lignes et j allant de 1 jusqu'à p, p étant le nombre de colonnes. Le Décodeur vise à reconstituer l'entrée à partir du code z_i : $\hat{x}_i = h(z_i)$. Les paramètres de l'encodeur et du décodeur sont

appris simultanément pendant la tâche de reconstruction, tout en minimisant la fonction objectif :

$$\zeta = \sum_{i=1}^{n} \varsigma(x_i, h(f(x_i))) = \sum_{i=1}^{n} \varsigma(x_i, h(z_i))) = \sum_{i=1}^{n} \varsigma(x_i, \hat{x}_i) \tag{7}$$

ς étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites.
f et h sont des fonctions de transition :
$f : \chi \to F$
$h : F \to \chi$ où F et χ sont respectivement les ensembles d'entrée et de sortie de l'encodeur:
$f \circ h = argmin \, \|x - (f \circ h)(x)\|^2 \, ou (f \circ h)(x) = f[g(x)]$ Dans le cas où il n'y a qu'une seule couche cachée, l'étape d'encodage prend l'entrée $x \in \mathbb{R}^p = \chi, h \in \mathbb{R}^k = F, p \geq k$

$$h = f(wx + b) \tag{8}$$

où h est généralement appelé code, variable latente ou représentation latente, w est une matrice de poids du réseau de neurones et b un vecteur de biais. Ensuite, l'étape de décodage associe h à la reconstruction $\hat{x}$ de forme identique à x :

$$\hat{x} = g(w'z + b) \tag{9}$$

où w'et b 0 pouvant être différents ou non de w et b de l'encodeur, selon la conception de l'Auto-Encodeur. La figure 11 illustre l'architecture d'un Auto-Encodeur

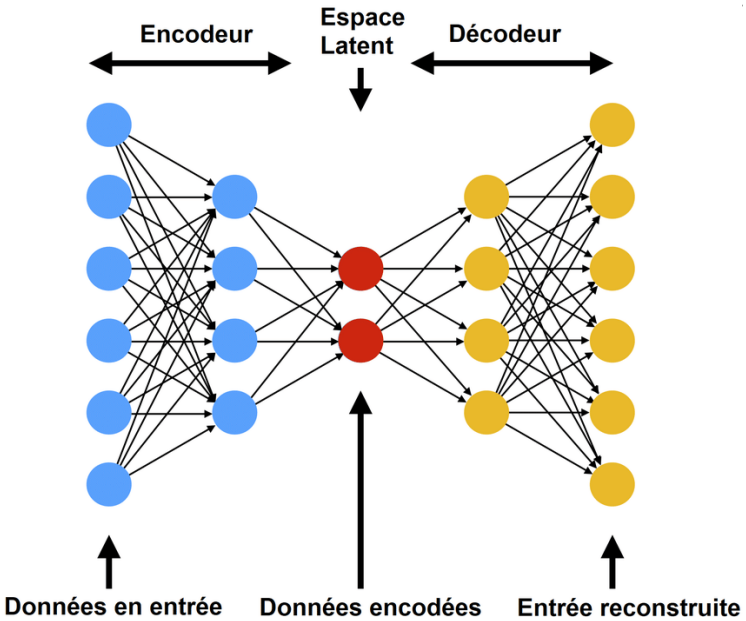

Figure 11: Architecture d'un Auto-Encodeur avec 2 couches d'encodeur et de décodeur et une couche cachée dite code

L'encodeur représente la partie du réseau qui compresse l'entrée dans un espace latent représentant la couche code. Le code est la couche cachée qui est généralement représentée sous forme compressée dans une dimension réduite. Il constitue aussi l'entrée alimentée du décodeur. Le décodeur est la partie du réseau qui tente de reconstruire l'entrée à partir de l'espace latent.

0.3.4 Perte de reconstruction

Examinons maintenant les pertes liées à la reconstruction que nous utilisons généralement avec les auto-encodeurs. La perte globale pour le jeu de données est donnée comme la perte moyenne par échantillon, c'est-à-dire :

$$L = \frac{1}{m} \sum_{j=1}^{m} (x^{(j)}, \hat{x}(j)) \tag{10}$$

Lorsque l'entrée est catégorielle, nous pouvons utiliser la perte d'entropie-croisée pour calculer la perte par échantillon qui est donnée par :

$$l(x, \hat{x}) = -\sum_{i=1}^{n} [x_i log(\hat{x}_i) + (1 - x_i)] \tag{11}$$

Et lorsque l'entrée est évaluée en valeur réelle, nous pouvons utiliser la perte d'erreur moyenne au carré donnée par :

$$l(x, \hat{x}) = \frac{1}{2} \|x - \hat{x}\|^2 \tag{12}$$

0.3.5 Paramètres à définir pour entraîner un Auto-Encodeur

Comme pour le réseau de neurones en général, l'algorithme de descente du gradiant est souvent utilisé pour entrainer un Auto-Encodeur. Quatre paramètres sont à prendre en compte pour entrainer un Auto-Encodeur :

- La taille du code (couche cachée) qui représente le nombre de neurones dans la couche code.
- Le nombre de couches : un Auto-Encodeur peut avoir une ou plusieurs couches.
- Le nombre de neurones par couches qui, en général, diminue dans l'encodeur et augmente dans le décodeur (symétrie par rapport à la couche code de l'encodeur et du décodeur).
- La fonction objectif : Un Auto-Encodeur est aussi entrainé pour minimiser la fonction objectif. Cette dernière est en général l'erreur quadratique. Dans le cas où les données d'entrée sont sous forme de probabilités (compris entre 0 et 1), la fonction de perte la plus adaptée est l'entropie croisée.
- Le nombre d'époques :C'est le nombre de fois que l'on entraine le réseau de neurones avec toutes les données

0.4 Courbes de Validation pour l'évaluation de modèles

0.4.1 Introduction

Les modèles d'apprentissage automatique sont des outils puissants pour résoudre des problèmes complexes tels que la reconnaissance d'images, la classification de texte, etc. Cependant, la performance des modèles dépend fortement des hyperparamètres choisis et des données d'entraînement. Il est donc important d'évaluer la performance des modèles et de choisir les hyperparamètres optimaux.

Les courbes de validation sont des outils graphiques qui permettent d'évaluer la performance des modèles en fonction des hyperparamètres et de choisir les meilleurs hyperparamètres pour un modèle donné. Dans cette partie, nous allons étudier les courbes de validation et leur utilisation pour évaluer la performance des modèles.

0.4.2 Définition

Les courbes de validation sont utilisées pour évaluer la performance d'un modèle d'apprentissage automatique en traçant une courbe qui montre comment la performance du modèle varie en fonction de la taille de l'ensemble de données d'entraînement. En général, la performance est mesurée en utilisant une métrique de performance, telle que la précision, le score F1, ou une autre métrique appropriée pour le problème de classification ou de régression considéré.

Théorie mathématique de la courbe

La courbe de validation est une représentation graphique qui permet d'évaluer la performance d'un modèle d'apprentissage automatique en fonction de ses paramètres d'hyperparamètres. Elle est généralement construite en traçant l'évolution d'une mesure de performance (par exemple, l'erreur quadratique moyenne ou la précision) en fonction de la variation des valeurs des hyperparamètres.
Mathématiquement, la courbe de validation peut être représentée par une fonction f qui prend en entrée les hyperparamètres h du modèle et renvoie la mesure de performance (par exemple, l'erreur quadratique moyenne) correspondante pour le modèle entraîné avec ces paramètres :

$f(h)$= mesure de performance pour le modèle entraîné avec les hyperparamètres h

En général, la courbe de validation est construite en faisant varier un ou plusieurs hyperparamètres à des valeurs différentes, puis en traçant la mesure de performance correspondante pour chacune de ces valeurs. La courbe de validation peut ensuite être utilisée pour identifier les valeurs optimales des hyperparamètres qui permettent d'obtenir la meilleure performance du modèle sur de nouveaux ensembles de données.

0.4.3 Étapes pour créer une courbe de validation

1. Diviser le jeu de données en ensemble d'entraînement et ensemble de validation. L'ensemble d'entraînement est utilisé pour entraîner le modèle, tandis que l'ensemble de validation est utilisé pour évaluer les performances du modèle.

2. Définir un modèle d'apprentissage automatique. Il peut s'agir de tout modèle, tel qu'un arbre de décision, un réseau de neurones.

3. Entraîner le modèle sur l'ensemble d'entraînement et calculer les scores de performance (précision, score F1, etc.) sur l'ensemble de validation.

4. Répéter les étapes 2 et 3 plusieurs fois avec différentes valeurs de paramètres de modèle pour obtenir plusieurs scores de performance sur l'ensemble de validation.

5. Tracer les scores de performance en fonction des valeurs de paramètres de modèle. Cela crée une courbe de validation.

6. Analyser la courbe de validation pour sélectionner les paramètres optimaux du modèle

Il est important de noter que la division des données en ensembles d'entraînement et de validation, ainsi que le choix des paramètres à tester et la sélection du meilleur ensemble de paramètres, doivent être effectués de manière à éviter le surapprentissage (overfitting) du modèle aux données d'entraînement. Par exemple, la validation croisée (cross-validation) peut être utilisée pour évaluer la performance du modèle sur plusieurs ensembles de validation différents, afin d'éviter de sélectionner un ensemble de validation qui serait trop proche des données d'entraînement.

0.4.4 Types de Courbes de Validation

Il existe principalement deux types de courbes de validation : la courbe de validation croisée et la courbe d'apprentissage.

1. La Courbe de Validation croisée : elle représente la performance du modèle en fonction de différents paramètres de modèle. Elle est généralement construite en entraînant le modèle en utilisant différents paramètres de modèle et en évaluant sa performance sur l'ensemble de validation. La courbe de validation croisée permet de sélectionner le meilleur modèle en fonction des paramètres qui donnent la meilleure performance sur l'ensemble de validation.

2. La Courbe d'Apprentissage : elle représente la performance du modèle en fonction de la taille de l'ensemble de données d'entraînement. Elle est généralement construite en entraînant le modèle sur différents sous-ensembles d'entraînement de différentes tailles et en évaluant sa performance sur l'ensemble de validation. La courbe d'apprentissage permet de détecter si le modèle est surajusté ou sous-ajusté, en observant la différence de performance entre l'ensemble de formation et l'ensemble de validation.

En résumé, la courbe de validation croisée permet de sélectionner le meilleur modèle en fonction des paramètres, tandis que la courbe d'apprentissage permet de détecter si le modèle est surajusté ou sous-ajusté.

0.4.5 Courbes d'apprentissage

Un modèle d'apprentissage d'un modèle d'apprentissage automatique montre comment l'erreur de prédiction d'un modèle d'apprentissage automatique change lorsque la taille de l'ensemble d'apprentissage augmente ou diminue. Avant de continuer, nous devons d'abord comprendre ce que signifient la variance et le biais dans le modèle d'apprentissage automatique.

- Biais : il ne s'agit essentiellement que de la différence entre la prédiction moyenne d'un modèle et la valeur correcte de la prédiction. Les modèles avec un biais élevé font beaucoup d'hypothèses sur les données de formation. Cela conduit à une simplification excessive du modèle et peut entraîner une erreur élevée sur les ensembles d'apprentissage et de test. Cependant, cela rend également le modèle plus rapide à apprendre et facile à comprendre. Généralement, les algorithmes de modèles linéaires comme la régression linéaire ont un biais élevé.

- Variance : il s'agit de la quantité de prédiction d'un modèle qui changera si les données d'apprentissage sont modifiées. Idéalement, un modèle d'apprentissage automatique ne devrait pas trop varier avec un changement dans les ensembles d'apprentissage, c'est-à-dire que l'algorithme devrait être capable de saisir des détails importants sur les données, quelles que soient les données elles-mêmes. Des exemples d'algorithmes à forte variance sont les arbres de décision, les machines à vecteurs de support (SVM).

Idéalement, nous voudrions un modèle avec une faible variance ainsi qu'un faible biais. Pour obtenir un biais plus faible, nous avons besoin de plus de données d'entraînement, mais avec des données d'entraînement plus élevées, la variance du modèle augmentera. Donc, il faut trouver un équilibre entre les deux. C'est ce qu'on appelle le compromis biais-variance.

Une courbe d'apprentissage peut aider à trouver la bonne quantité de données d'entraînement pour ajuster notre modèle avec un bon compromis biais-variance. C'est pourquoi les courbes d'apprentissage sont si importantes.

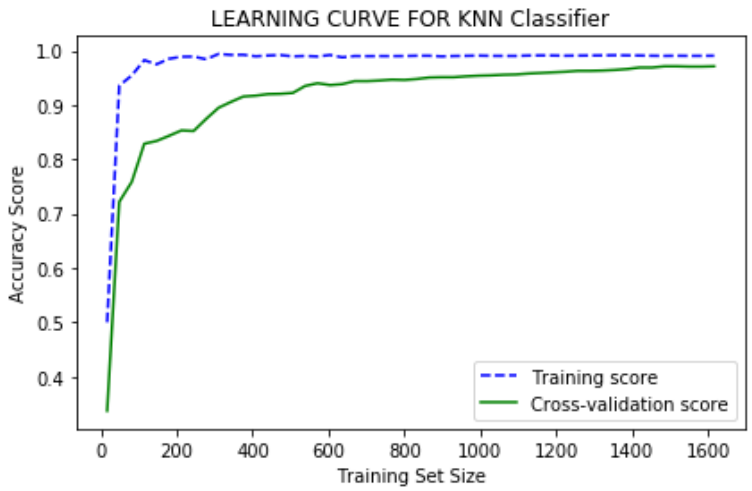

Figure 12: la courbe des scores d'apprentissage et la courbe des scores de validation croisée convergent

À partir de la courbe, nous pouvons clairement voir qu'à mesure que la taille de l'ensemble d'apprentissage augmente, la courbe des scores d'apprentissage et la courbe des scores de validation croisée convergent. La précision de la validation croisée augmente à mesure que nous ajoutons plus de données de formation. L'ajout de données d'entraînement est donc utile dans ce cas. Étant donné que le score d'entraînement est très précis, cela indique un faible biais et une variance élevée. Ainsi, ce modèle commence également à surajuster les données car le score de validation croisée est relativement plus faible et augmente très lentement à mesure que la taille de l'ensemble d'apprentissage augmente.

0.4.6 Les phases de la courbe d'apprentissage

La courbe d'apprentissage est généralement composée de trois phases distinctes :

1. La phase de sous-apprentissage (underfitting) : Dans cette phase, le modèle n'a pas assez de données pour apprendre la structure des données. Par conséquent, la performance du modèle est faible à la fois sur les données d'entraînement et sur les données de validation. Cela se manifeste par une courbe qui montre une performance médiocre, qui stagne ou qui diminue à mesure que la quantité de données d'entraînement augmente.

2. La phase d'apprentissage optimal (optimal fitting) : Dans cette phase, le modèle a suffisamment de données pour apprendre la structure des données, sans être trop complexe. Par conséquent, la performance du modèle est bonne à la fois sur les données d'entraînement et sur les données de validation. Cela se manifeste par une courbe qui montre une performance croissante à mesure que la quantité de données d'entraînement augmente, mais qui commence à se stabiliser à un certain point.

3. La phase de sur-apprentissage (overfitting) : Dans cette phase, le modèle est trop complexe par rapport à la quantité de données d'entraînement, et peut donc surajuster les données d'entraînement. Cela se manifeste par une performance très bonne

sur les données d'entraînement, mais une performance médiocre sur les données de validation. La courbe de validation peut montrer une performance qui stagne ou qui diminue à mesure que la quantité de données d'entraînement augmente, tandis que la courbe d'entraînement continue d'augmenter.

Conclusion

les courbes d'apprentissage sont un excellent outil de diagnostic pour déterminer le biais et la variance dans un algorithme d'apprentissage automatique supervisé. Dans cet article, nous avons appris quelles courbes d'apprentissage et comment elles sont implémentées en Python.

Conclusion et perspectives

Conclusion générale

nous avons voir dans ce Mini-projet Les modèles de réseaux de neurones non supervisés, tels que les auto-encodeurs, sont de puissants outils d'apprentissage automatique qui peuvent être utilisés pour extraire des caractéristiques pertinentes à partir de données non étiquetées. Les auto-encodeurs sont des réseaux de neurones qui ont pour but de reconstruire une entrée donnée en sortie, en passant par une couche cachée de dimensions plus faibles. Les auto-encodeurs peuvent être utilisés pour des tâches telles que la compression de données, la réduction de dimensionnalité, l'augmentation de données, la détection d'anomalies, etc.

Les courbes de validation sont une méthode efficace pour évaluer la performance d'un modèle de machine learning en fonction de différents paramètres. Les courbes de validation sont particulièrement utiles pour ajuster les hyperparamètres d'un modèle, tels que le taux d'apprentissage, le nombre de couches dans le réseau de neurones, etc. Les courbes de validation permettent de choisir les meilleurs paramètres pour le modèle, en évitant le surapprentissage aux données d'entraînement.

En conclusion, les auto-encodeurs sont des modèles de réseaux de neurones non supervisés puissants pour l'analyse et la transformation de données non étiquetées, tandis que les courbes de validation sont une méthode efficace pour évaluer et ajuster les paramètres d'un modèle de machine learning. En combinant les deux approches, on peut obtenir des modèles de réseaux de neurones non supervisés optimisés pour une tâche donnée, avec une performance maximale sur les données de validation.

Perspectives

En conclusion, les modèles de réseaux de neurones non supervisés, tels que les auto-encodeurs, ainsi que les courbes de validation, sont des outils puissants pour l'apprentissage automatique. Les perspectives futures de ces approches sont prometteuses, avec des applications potentielles dans de nombreux domaines, tels que la reconnaissance d'images, la synthèse de données, la détection d'anomalies, etc.

- Moindre des ressources dans la partie de courbe de validation pour les détailles mathématique et son utilisation seulement avec des bibliothèques de python

Bibliography

[1] *CLASSIFICATION, RÉDUCTION DE DIMENSIONNALITÉ ET RÉSEAUX DE NEURONES : DONNÉES MASSIVES ET SCIENCE DES DONNÉES*. Nov. 2020.

[**2**] https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning/5814621-initiez-vous-aux-autoencodeurs. 2023

[**3**] https://datascientest.com/les-autoencoders-modeles-dapprentissage-non-supervise. 2023

[**4**] https://scikit-learn.org/stable/modules/neural networks nsupervised.html. 2023

[**5**]https://www.projectpro.io/recipes/plot-validation-curve-in-python. 2023

[**6**]https://stacklima.com/utilisation-des-courbes-dapprentissage-ml/. 2023

[**7**]https://alp.developpez.com/tutoriels/intelligence-artificielle/reseaux-de-neurones. 2023

[**8**]https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle/-/blob/master/AE/01-Prepare-MNIST-dataset.ipynb. 2023

[**9**]https://atcold.github.io/pytorch-Deep-Learning. 2023

[**10**]ChatGPT, GPT-3.5 .2023