Université de Lille
Faculté des Sciences et Technologies

PJI: MultiLayer Perceptron

Zeyd BOUMEHDI

Master Informatique
Master mention Informatique

Département d'Informatique
Faculté des Sciences et Technologies

mars, 2022

Résumé

De nos jours des modèles bio-inspirés par le cerveau humain ont été créées *(réseaux de neurones)*, le premier est le Perceptron en 1957 et en 1986, le Multilayer Perceptron *(MLP)*.

Ce dernier, grâce son efficacité et sa simplicité d'utilisation, est toujours très utilisé.

Lorsqu'on utilise des réseaux de neurones, plusieurs algorithmes sont à connaitre et plusieurs paramètres sont à configurer pour avoir un apprentissage optimal. Concernant les algorithmes, il est nécessaire de maitriser la descente de gradient qui permet de trouver l'optimal d'une fonction grâce à ses dérivées partielles et il faut aussi comprendre la rétropropagation du gradient afin de calculer les dérivées partielles d'un réseau de neurones. Pour les paramètres à configurer sont concernés le pas d'apprentissage, la taille du réseau de neurones ou encore le nombre d'itération utilisé lors de la descente de gradient, car cela influe sur le temps de calcul et la performance du réseau.

Enfin, les applications d'un MLP sont diverses, car ce réseau permet de faire la régression ou de la classification.

Abstract

Nowadays, bio-inspired models of the human brain have been created, the first one is the Perceptron in 1957 and in 1986, the Multilayer Perceptron.

The Multilayer Perceptron is very efficient and still widely used because it is easy to use.

When using neural networks, several algorithms are to be known and several parameters are to be configured to have an optimal learning. The algorithms to know are the gradient descent which allows to find the optimal of a function with this partial derivatives and it is also necessary to understand the gradient backpropagation in order to calculate the partial derivatives of a neural network. The parameters to configure are the learning rate, the size of the neural network or the number of iterations used during the gradient descent, as this affects the computation time and the performance of the network.

Finally, the applications of an MLP are diverse, because this network allows to do regression or classification.

Indice

Chapitre 1

Introduction

1.1 Contextualisation

Depuis les années 2000 avec l'essor d'Internet et des ordinateurs de plus en plus performants, l'intelligence artificielle et plus particulièrement l'apprentissage profond (Deep Learning) sont devenus des techniques populaires. Dans ce rapport de PJI, nous nous intéresserons à un modèle de Deep Learning très connu et efficace, nommé le MultiLayer Perceptron (MLP).

Premièrement, nous définirons les termes importants, puis dans une deuxième partie, nous étudierons les paramètres importants pour une bonne configuration d'un MLP, et enfin en dernière partie, nous explorerons les différentes applications possibles avec un MLP.

1.2 Définitions

Intelligence artificielle (IA) : Concept visant à permettre aux ordinateurs/machines d'avoir une forme d'intelligence.

Machine Learning (Apprentissage automatique) : Approche mathématique et statistique pour qu'un modèle apprenne à partir de données et sans une programmation explicite du problème. Le Machine Learning est une des sous-catégories de l'intelligence artificielle.

Apprentissage profond (Deep Learning) : Le Deep Learning est une sous-catégorie du Machine Learning et se caractérise par l'utilisation de réseau de neurones.

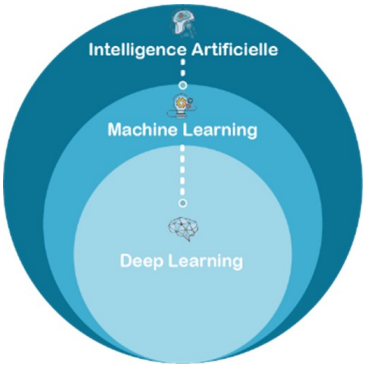

FIGURE 1.1 – Schéma des différentes composantes de l'Intelligence Artificielle.

Classification : Modèle de Machine Learning qui prédit une classe (une valeur catégorielle).

Régression : Modèle de machine learning qui prédit une valeur continue.

Réseau de neurones : Méthode s'inspirant du cerveau humain pour recréer de l'intelligence artificiellement. Un réseau de neurones est composé de plusieurs couches. La première couche est celle d'entrée (en vert), elle correspond aux données que l'on renseigne dans le modèle. La deuxième couche (en bleu) est ce qu'on appelle la couche cachée. Il peut y en avoir plusieurs et c'est donc dans ces couches que notre réseau apprend. La dernière couche (en jaune) est celle de sortie, elle représente la prédiction du modèle (résultat).

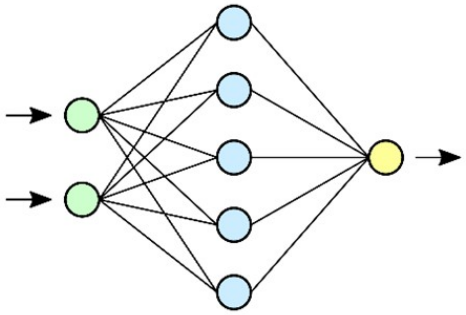

FIGURE 1.2 – Schéma d'un réseau de neurones.

Chaque rond sur le schéma est un neurone et ils sont liés par des arêtes qui ont des poids. Le choix du réseau de neurones dépend de l'application à mettre en place.

MultiLayer Perceptron (MLP ou en français Perceptron Multicouche) : Le MLP est le premier réseau de neurones, c'est une extension du Perceptron. Il s'agit d'un réseau de neurones où chaque neurone est un Perceptron.

Un Perceptron est un modèle assez simple car il fait une somme pondérée avec ses entrées (x) et les poids (w) sur ses arêtes puis passe ce résultat dans une fonction d'activation (f(x)) pour avoir une sortie (y). *Exemple : f(x1w1 + x2w2 + ... + xnwn) = y.*

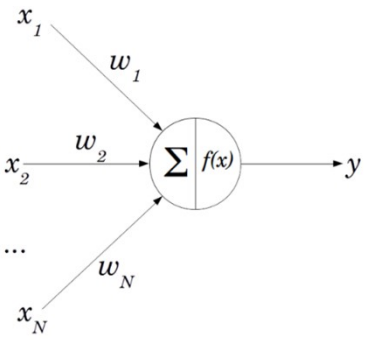

FIGURE 1.3 – Schéma d'un Perceptron.

Grâce à l'empilement de plusieurs Perceptrons, le MLP peut résoudre des problèmes non-linéairement séparable (qui ne se séparent pas par une droite).

Exemple :

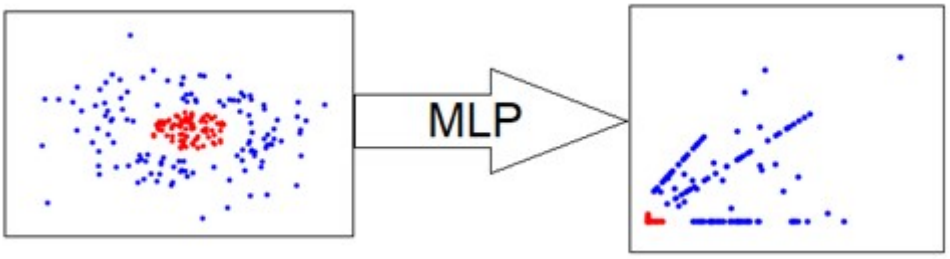

FIGURE 1.4 – Schéma d'un MLP qui rend un problème linéairement séparable.

On est parti d'un problème non linéairement séparable puis en le passant dans un MLP il est devenu linéairement sérapable en mettant les bleus d'un côté et les rouges de l'autre.

Chapitre 2

Les paramètres importants

2.1 Fonction de perte

Une fonction de perte est une fonction qui mesure la différence entre des données prédites et des données réelles. Les fonctions de perte possibles pour un MLP sont la Cross-Entropy Loss, la Log Loss *(elles sont presque similaires)* ou la MSE *(Mean Square Error)*. La Log Loss est utilisée pour des problèmes de classification binaire alors que la Cross-Entropy Loss est utilisée pour des classifications binaires et multi-classes. Lorsque le problème à résoudre est une régression alors la fonction de perte utilisée est la MSE.

2.2 Algorithme de descente de gradient

La descente de gradient est un algorithme d'optimisation. Il vise à réduire une fonction de perte afin d'avoir l'erreur la plus petite possible (optimum). Son but est ainsi de mettre à jour les paramètres du réseau de neurones par le biais de la fonction de perte et lorsqu'on trouve l'optimum global de la fonction de perte alors nous obtenons les meilleurs paramètres possibles.

Calculer un gradient dans un réseau de neurones peut être compliqué car il y a beaucoup de paramètres et de couches à trouver. Toutefois, il existe une solution la rétropropagation du gradient.

➢ *Rétropropagation du gradient (Back propagation) :* C'est une méthode pour obtenir le gradient dans un réseau de neurones en calculant ses dérivées partielles.

Ensuite le gradient nous donne la direction dans laquelle il faut aller pour trouver l'optimum dans l'algorithme de descente de gradient.

Lorsqu'on utilise cet algorithme, on peut rencontrer des problèmes de configuration :

Dans les réseaux de neurones, les fonctions de perte sont non-convexes, c'est-à-dire qu'il peut y avoir plusieurs optimums locaux qui sont ceux que nous voulons éviter car nous cherchons l'optimum global.

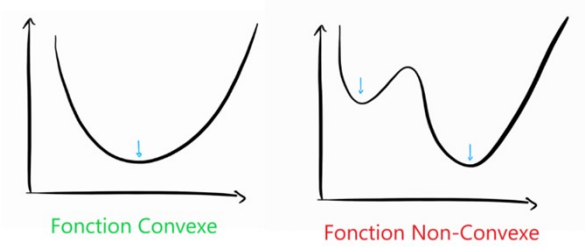

FIGURE 2.1 – Schéma de comparaison entre fonction convexe et fonction non-convexe.

Deux paramètres à bien configurer pour trouver l'optimum global :

1. Le pas d'apprentissage :

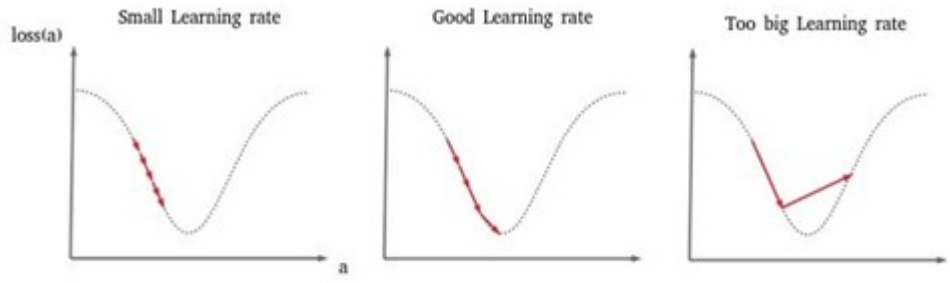

FIGURE 2.2 – Schéma de comparaison entre les différentes tailles de pas d'apprentissage.

Si le pas d'apprentissage est trop petit alors il se peut que l'algorithme ne converge pas ou se bloque dans un optimum local. A l'inverse, s'il est trop grand, il se peut qu'il loupe l'optimum global et ne converge jamais. Alors il nous faut trouver un bon pas d'apprentissage qui ne soit ni trop grand, ni trop petit.

2. Le nombre maximum d'itération :

   C'est le nombre maximal d'exécution pour l'algorithme de descente de gradient s'il n'a pas convergé avant. Cette valeur ne doit pas être trop grande sinon cela risque de ralentir l'apprentissage *(Le package python pour le Machine Learning, Scikit-Learn, le définit à 200 itérations)*.

Il existe différents algorithmes de descente de gradient. Les plus adaptés pour les réseaux de neurones sont ceux qui utilisent de l'aléatoire *(Descente de Gradient Stochastique)* ou ceux qui utilisent un pas d'apprentissage adaptatif *(Descente de Gradient de Newton)*, car ils permettent, tous deux, une convergence plus rapide vers l'optimum global et ils peuvent s'extraire des optimums locaux facilement.

2.3 Taille du réseau

La taille du réseau de neurones correspond aux dimensions de la couche cachée et celle-ci peut s'étendre sur plusieurs couches.

Donc ce paramètre est important car il ne doit pas être trop grand sinon son temps de calcul pourrait devenir astronomique, ni trop petit car le modèle doit pouvoir apprendre les caractéristiques des données. Il n'existe pas de règle précise à ce sujet car cela dépend aussi de la machine utilisée et de sa puissance.

2.4 Fonction d'activation

Dans cette dernière sous partie, nous allons rentrer dans un neurone. Comme vu précédemment un MLP fait une somme pondérée de ses entrées et des poids sur ses arêtes et passe le résultat obtenu dans une fonction f(x).

f(x) est la fonction d'activation, elle sert à retransmettre une information d'un état du *synapse*, son état peut être activé ou désactivé. La fonction d'activation la plus utilisée est ReLU car elle est simple et efficace d'utilisation.

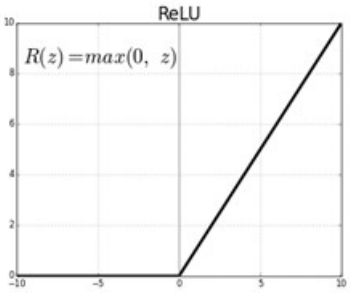

FIGURE 2.3 – Schéma de la fonction d'activation ReLU.

La fonction ReLU prend le maximum entre 0 et la valeur d'entrée. Donc si votre valeur est négative alors ReLU retournera 0, et si la valeur est positive alors elle retournera votre valeur.

Note : Il existe pleins d'autres fonctions activations, c'est à vous de choisir la meilleure selon votre problème.

Chapitre 3

Applications

Les MLP peuvent avoir plusieurs utilisations, comme par exemple, prédire le loyer d'un appartement en fonction du mètre du carré *(régression)* mais il peut aussi être utilisé dans un but plus complexe de classification d'images.

3.1 Régression linéaire

Pour montrer ce qu'est une régression linéaire, nous allons générer des données aléatoires avec la fonction make-regression() de Scikit-Learn.

Exemple et affichage de données dans le dataset (jeu de données) :

	X	y
0	-0.359553	-19.955886
1	0.976639	21.339773
2	0.402342	11.556895
3	-0.813146	-16.342069
4	-0.887786	-35.700638

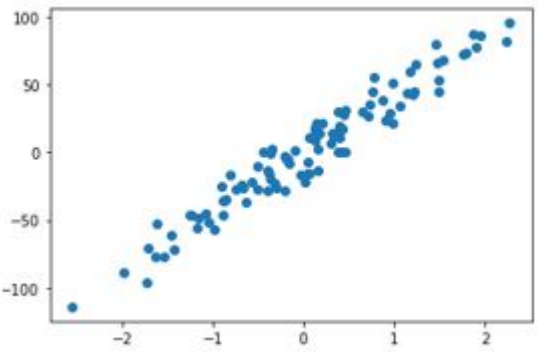

FIGURE 3.1 – Données d'une régression linéaire.

La colonne **X** renseigne toutes les valeurs des points en abscisse et la colonne cible **y** est la valeur en ordonnée.

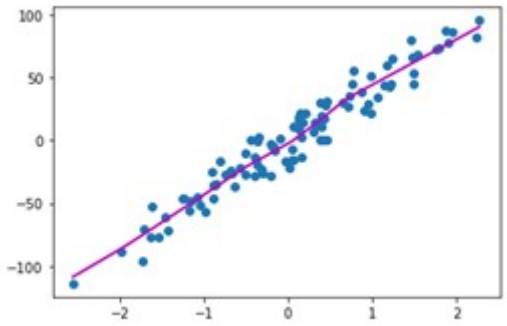

FIGURE 3.2 – Graphique d'une régression linéaire.

Le but de la régression est de trouver la courbe qui passe par le nuage de points et qui minimise la fonction de perte. Ici, elle est représentée par la courbe violette.

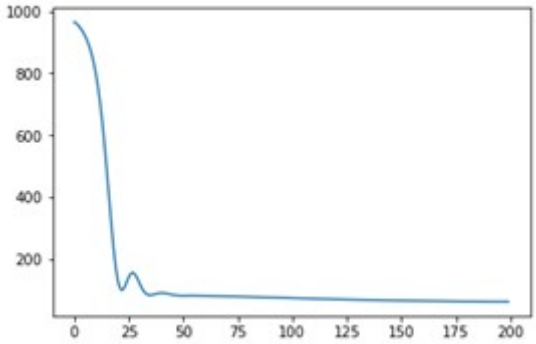

FIGURE 3.3 – Graphique de la fonction de perte d'une régression linéaire.

Sur la courbe ci-dessus, nous voyons la fonction de perte qui diminue ce qui signifie que le modèle a appris. De plus, notre modèle a un score de 98% cela veut dire qu'il n'y a 2% de taux d'erreur.

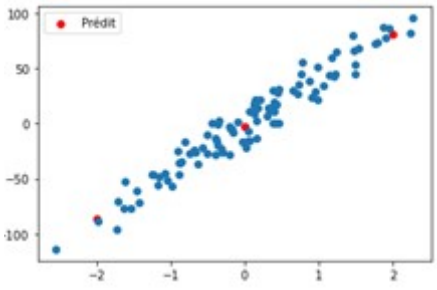

FIGURE 3.4 – Graphique d'une prédiction avec une régression linéaire.

Lorsqu'on veut prédire avec une régression linéaire il suffit de lui donner un X (un point en abscisse) et il nous donnera un point en ordonnée. Par exemple, à gauche nous avons demandé de nous prédire y pour X = [-2, 0, 2] (en rouge).

Note : Le MLP peut être aussi utilisé sur des régressions non-linéaires.

3.2 Classification d'iris

Note : Le dataset iris est celui de Scikit-Learn.

L'objectif est de prédire la classe d'un iris en fonction de la taille et de la longueur de son sépale et de son pétale donc nous avons 4 valeurs qui définissent cette fleur et grâce à ces données nous pouvons déduire sa classe en le passant dans un MLP.

Exemple de données dans le dataset :

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

FIGURE 3.5 – Données du dataset iris.

On retrouve bien les 4 valeurs citées auparavant et apparaît aussi une nouvelle colonne **target** qui représente la vraie classe de l'iris. La **target** a 3 valeurs possibles 0 si la variété est Setosa, 1 si c'est Versicolor et 2 pour Virginica. On vise à minimiser la différence entre la **target** et les valeurs prédites par le modèle, c'est la fonction de perte.

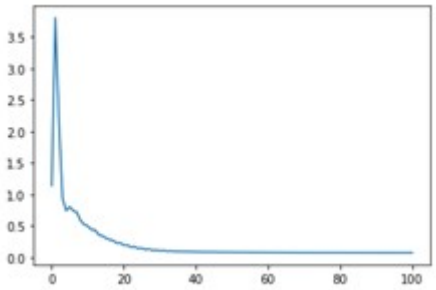

FIGURE 3.6 – Graphique de la fonction de perte.

Voici à quoi ressemble l'évolution de la fonction de perte selon le nombre d'itération de l'algorithme de descente de gradient. On remarque que plus il itère, plus il apprend c'est-à-dire que sa fonction de perte diminue (il converge). Ici, la meilleure perte est à 0.07.

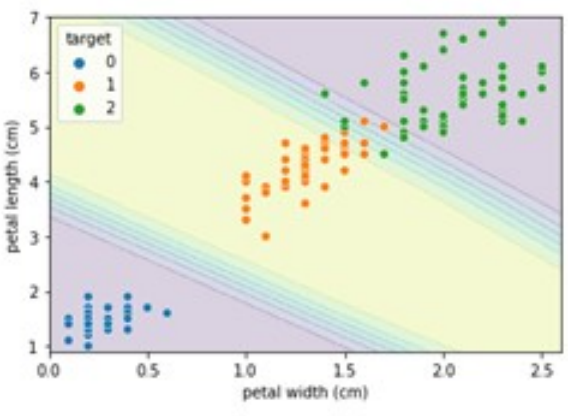

FIGURE 3.7 – Graphique de la prédiction pour le dataset iris.

Sur la figure 3.7, on constate la séparation obtenue après avoir trouvé l'optimum. On voit bien que chaque classe est linéairement séparée par une droite et donc si une nouvelle donnée d'iris est rentrée dans le MLP alors il essayera de le classifier en le mettant soit dans la partie en bas à gauche *(Setosa)*, au milieu *(Versicolor)* et en haut à droite *(Viriginica)*.

3.3 Classification simpliste pour l'analyse d'images

Le but est de prédire le genre d'une personne à partir d'une image. Pour cela, il existe plusieurs datasets trouvable sur Internet.

Dataset : Disponible sur Kaggle.

Exemple de données dans le dataset :

	age	ethnicity	gender	pixels
0	1	2	0	129 128 128 126 127 130 133 135 139 142 145 14...
1	1	2	0	164 74 111 168 169 171 175 182 184 188 193 199...
2	1	2	0	67 70 71 70 69 67 70 79 90 103 116 132 145 155...
3	1	2	0	193 197 198 200 199 200 202 203 204 205 208 21...
4	1	2	0	202 205 209 210 209 209 210 211 212 214 218 21...

FIGURE 3.8 – Données du dataset de Kaggle.

Le dataset contient plusieurs colonnes à prédire (age, ethnicity et gender) et des images en noir et blanc qui sont dans la colonne pixels, car une image est une matrice de pixels avec des valeurs qui vont de 0 à 255.

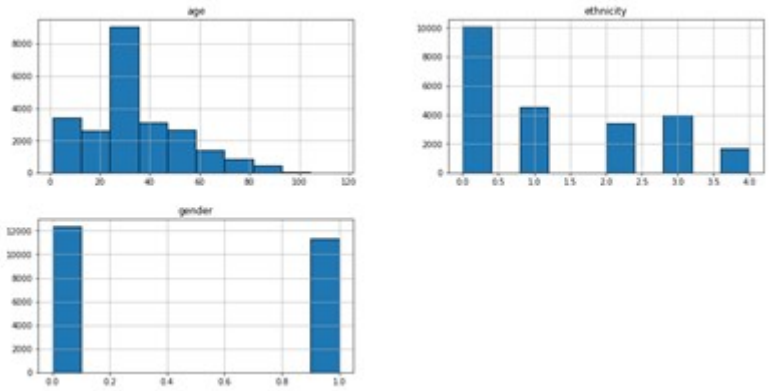

FIGURE 3.9 – Statistiques du dataset de Kaggle.

Note : Sur ce dataset, la moyenne d'âge est de 30 ans, il y autant de femmes que d'hommes.

Pour rendre lisible les images, une transformation des données est nécessaire car le réseau de neurones a besoin d'une image sous forme de vecteur (l'image doit être aplatie en une ligne de valeur). De plus, une seconde transformation a été effectuée, appelée la normalisation des images, c'est une méthode modifiant une image ayant des valeurs entre 0 et 255, en des valeurs entre 0 et 1, ce qui permet un meilleur temps de calcul.

Aussi, une recherche des meilleurs paramètres du modèle a été effectuée, pour cela il nous a suffi d'utiliser un GridSearch (Fonction Scikit-Learn) avec les paramètres à optimiser. Grâce à ce travail préalable de mise en place, on peut enfin entrainer le MLP avec les paramètres trouvés avec le GridSearch.

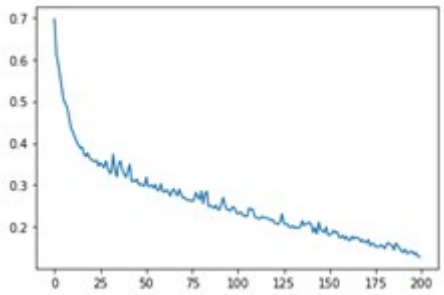

FIGURE 3.10 – Graphique de la fonction de perte.

On obtient la fonction de perte ci-dessus, le MLP réussit à apprendre les différences caractéristiques selon le genre.

Note : On pourrait aussi entrainer un autre modèle pour prédire l'âge (Régression).

FIGURE 3.11 – Schéma montrant ce que voit les neurones.

Le côté pratique avec les réseaux de neurones pour l'analyse d'images est que l'on peut observer dans les différents neurones pour voir ce qu'il a appris. Ici, on voit que les neurones extraient les caractéristiques du visage. Sur le schéma ci-dessus, on y voit plusieurs visages où l'on peut y reconnaitre les yeux, le nez et la bouche.

Voici le résultat final, lorsqu'on implémente le modèle sur une webcam :

FIGURE 3.12 – Prédiction obetnu par un MLP sur une webcam.

Chapitre 4

Conclusion

Pour conclure, les MLP sont les réseaux de neurones les plus basiques mais ils restent néanmoins très efficaces et très utilisés.

Dans ce rapport, nous avons vu les algorithmes les plus importants pour faire fonctionner un MLP comme la descente de gradient ou la rétropropagation et nous avons vu aussi comment bien les configurer. Ces algorithmes sont aussi utilisés dans d'autres réseaux de neurones et d'autres méthodes de machine learning.

Les MLP ont plusieurs applications possibles comme l'analyse d'images simplistes. Cette analyse est simpliste car une amélioration est possible de rendre le MLP plus performant lors de l'analyse d'image pour qu'il extrait des caractéristiques de meilleures qualités. Pour cela, il faudra utiliser un CNN (Convolutional Neural Network) qui est un MLP amélioré avec des couches de convolutions après la couche d'entrée.

Références

1. Informations concernant l'algorithme de descente de gradient.
2. Informations concernant la fonction d'activation.
3. Informations concernant la rétropropagation du gradient.
4. Différences entre un SGD et la rétropropagation.
5. Différences entre la Log Loss et la Cross Entropy Loss.
6. Documentation Scikit-learn sur les MLP Classifier.
7. Documentation Scikit-learn sur les MLP Regressor.
8. Extrait d'un livre sur le deep learning (CNN).
9. Vidéo youtube sur les CNN.