Projet

Intelligence Artificielle : Réseau de Neuron

Mohammed Mchich & Nizar Ouazzani Chahdi

Table des matières

1 Présentation des résultats obtenus

Dans cette première section du rapport nous allons présenter les résultats obtenus dans les deux cas implementés (réseau monocouche et réseau à 3 couches) avec différents paramètres.

1.1 Paramètres influençant la précision du réseau

— Normalisation des InputNeurons : Le choix de la normalisation joue une rôle crucial dans la réussite de la phase d'apprentissage.
En effet avec une normalisation de 1 le réseau n'apprendra quasiment pas (les poids ne seront pas modifiés de sorte à ce qu'ils permettent l'apprentissage). Cela est dû au fait que la fonction feed de chaque neuron dans la couche de sortie/-cachée prennent une somme de InputNeurons*LeursPoidsRespectifs. Cette somme sera incluse dans un intervalle de [-28*28,28*28] (en fait [-254*28*28,254*28*28] mais puisque l'initialisation des poids se fait aléatoirement, on aura autant de plus que de moins ce qui annule l'effet des 254 qui représente la valeur maximale prise par chaque pixel). Dans cet intervalle, les fonctions d'activations appliquées à cette somme ne pourront pas "voir" de différence entre les valeurs passées en argument puisque Sigmoid(+/-infini)=0/1 et Tanh(+/-infini)=1/-1.
Pour rendre l'apprentissage possible, nous avons introduit une normalisation de 28*28 ce qui permet à la somme calculée d'être dans l'intervalle [-1,1] et de ce fait permettra à la fonction d'activation de voir la différence entre chacune des entrées, ce qui rendra l'ajustement des poids plus efficace.

— Choix de la fonction d'activation : Le choix de la fonction d'activation influence le taux d'erreur obtenu. On obtient les meilleurs résultats (présentés ci-dessous) avec une fonction logistique.

— Taux d'apprentissage : Ce paramètre influence le nombre d'itération nécessaire pour assurer la convergence du taux d'erreur. Avec un petit eta le réseau nécessitera plus d'itérations pour converger. Nous avons gardé le taux prédéfini de 0,01 et avons effectué des tests avec un taux de 0,1.

1.2 Résultats

— Réseau monocouche : Le meilleur résultat est obtenu avec une configuration (28*28/Sigmoid).
Au bout des 10 itérations sur les données des 60000 exemples, nous atteignons une précision de 91,3%, ce qui est un résultat plutôt satisfaisant compte tenu du fait qu'on utilise une architecture avec une seule couche d'apprentissage (la couche d'entrée ne contribue pas, elle sert de lecteur des pixels de l'image).

— Réseau à trois couches : Nous avons fait plusieurs tests avec des configurations différentes en changeant à chaque fois la normalisation, les fonctions d'activation et

le nombre de couches cachées. Le meilleur résultat est obtenu avec une configuration (128 neurons cachés/28*28/Sigmoid-Sigmoid). La précision atteint est de 95,7%, le meilleur résultat affiché sur le site de la base de données MNIST est obtenu avec une configuration de [2-layer NN, 300 hidden units, mean square error], le taux d'erreur est de 4,7% ce qui n'est pas loin du résultat que nous avons obtenu avec seulement 48 neurons cachés.

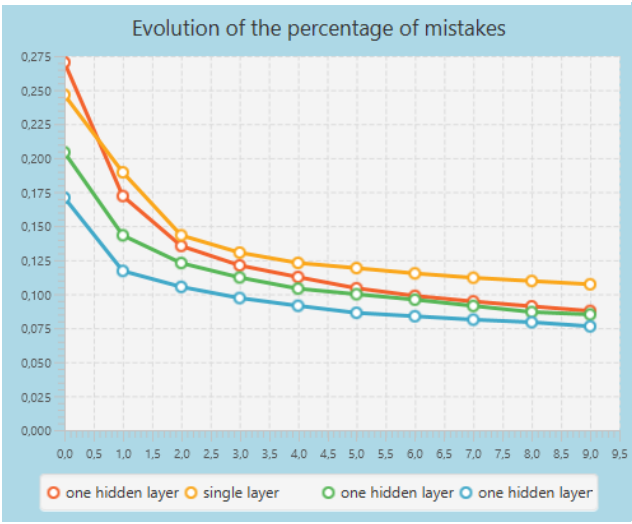

FIGURE 1 – Résultats obtenus avec les différentes configurations & eta=0,01

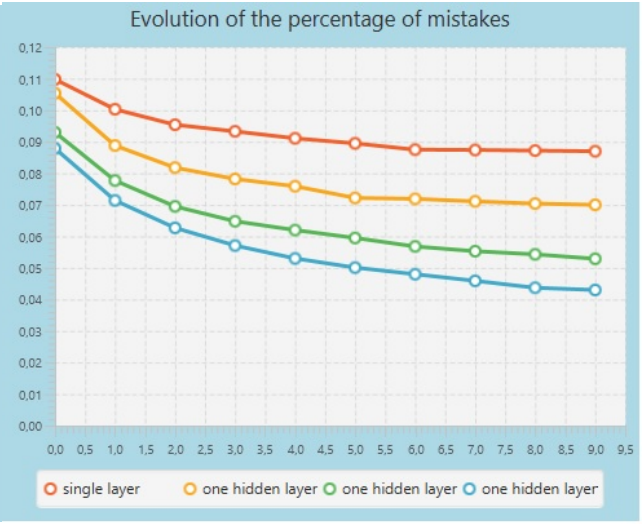

FIGURE 2 – Résultats obtenus avec les différentes configurations & eta=0,1

Courbe	Compilation	n Hidden	Normalisation	Activation	Taux d'erreur
Orange | SingleLayer_0.01	4 minutes	-	28*28	Sigmoid	0,1072
Rouge | HiddenleLayer_0.01	9 minutes	24	28*28	Sig - Sig	0.0876
Verte | HiddenleLayer_0.01	20 minutes	48	28*28	Sig - Sig	0.085
Bleue| HiddenleLayer_0.01	65 minutes	128	28*28	Sig - Sig	0.0763
-	-	-	-	-	-
Rouge | SingleLayer_0.1	4 minutes	-	28*28	Sigmoid	0,087
Orange | HiddenleLayer_0.1	7 minutes	12	28*28	Sig - Sig	0.07
Verte | HiddenleLayer_0.1	9 minutes	24	28*28	Sig - Sig	0.0529
Bleue| HiddenleLayer_0.1	22 minutes	48	28*28	Sig - Sig	0.043

2 Difficultés d'implémentation

La phase cruciale de la réussite du projet était la bonne compréhension de l'algorithme de descente du gradient et de ses différentes étapes. L'étape la plus difficile à comprendre et à implémenter était celle du 'Back propagation'.

— Single Layer :
Dans ce cas il suffisait de comprendre l'architecture déjà donnée et de savoir manipuler les différentes instances de chacun des objets pour pouvoir implémenter l'algorithme de descente du gradient. Le tutoriel [1] était d'une aide précieuse.

— One Hidden Layer :
Dans ce cas, l'application de l'algorithme était plus difficile. Plus particulierement, l'implémentation de l'étape FeedBackWard (qui permet de propager l'erreur dans le réseau afin de mettre à jour les poids) dans le cas des neurons cachés était plus délicate.
La difficulté provenait d'une part de la complixité de la formule de mise à jour (nous nous sommes basées sur [2] et [3] afin de comprendre les étapes de l'algorithme) et d'autre part de l'architecture imposée (le neuron caché ne pouvait pas 'voir' directement les poids des connexions avec la couche de sortie).

3 Niveau de difficulté du projet

Etant donné que ce projet est considéré comme le "Hello World" du Machine learning, on trouve qu'il était intéressant de le faire même s'il ne traitait pas les problèmes rencontrées en pratiques (en particulier le 'nettoyage' de la base de donnée), il permetait bien saisir la notion du réseau de neuron et d'appliquer l'algorithme de descente du gradient.
Le projet était d'une difficulté correcte dans sa première partie et plus difficile dans la deuxième. Le temps imparti pour sa réalisation était suffisant.
L'architecture choisie rendait la mise en place des algorithmes plus compliquée. En effet, cela aurait été beaucoup plus simple et 'naturel' de manipuler des matrices et vecteurs.

Références

[1] Matt Lind *Machine Learning in C : Simple 1-Layer Neural Network for MNIST Handwriting Recognition*
https ://mmlind.github.io/Simple_1-Layer_Neural_Network_for_MNIST_Handwriting_Recogni

[2] Keith Downing *Advanced Artificial Intelligence - Gradient Descent Training Rule : The Details*
http ://www.idi.ntnu.no/ keithd/classes/advai/lectures/backprop.pdf.

[3] Institute of Technology Blanchardstown *Neural Networks and Back Propagation Algorithm*
http ://www.dataminingmasters.com/uploads/studentProjects/NeuralNetworks.pdf.