

Info-F-106 : Projet d'Informatique
Classification à l'aide de réseaux de neurones
Rapport

Table des matières : Pages

<u>Introduction :</u>	3
<u>Méthodes implémentées :</u>	3
<u>Comparaisons de performance :</u>	4
<u>Images de la nouvelle GUI :</u>	5
<u>Conclusion :</u>	5

Introduction :

L'objectif du projet est de réaliser un réseau de neurones (les réseaux de neurones sont des systèmes de calculs inspirés des réseaux de neurones biologiques qu'on trouve dans le cerveau animal) en Python 3 pour effectuer de la classification de données. Le réseau est entraîné sur un jeu de données, qu'il apprend à classer correctement. Une fois entraîné celui-ci peut alors faire des prédictions pour de nouvelles données. Ce projet a été réalisé dans le cadre du projet d'année de BA1. Ce projet s'exécute à l'aide de la commande `python3 partie3.py` sur le terminal. Il suffit ensuite de choisir le numéro que l'on veut entraîner, le dataset, alpha le paramètre qui divise `dataTrain` (la section de donnée pour entraîner les neurones) et `dataTest` (la section de donnée pour tester les neurones), la probabilité qu'un neurone soit désactivé, epsilon qui est utilisé pour vérifier la convergence de la méthode d'apprentissage, `learning_rate` utilisé pour la mise à jour des poids, `H` qui est la taille de la couche intermédiaire du neurone, Cross-validation représentant le nombre de fois que le neurone sera testé et training qui est le nombre de fois que le neurone sera entraîné. Côté affichage on peut voir l'état du dropout dans le tab neurones et dans le tab Poids on peut voir les pixels qui sont considérés importants pour déterminer à quel chiffre nous avons affaire. On peut également voir sur le terminal la comparaison entre deux fonctions d'activations différentes.

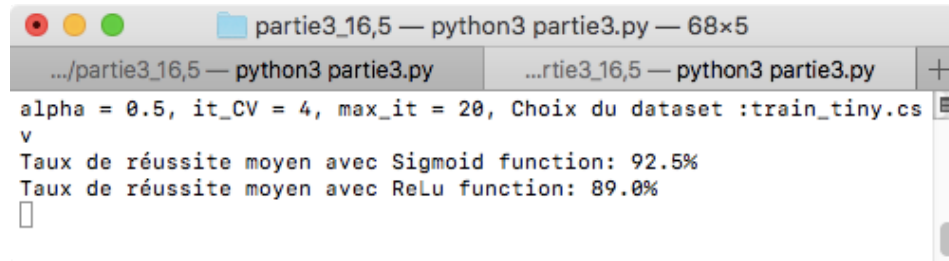
Méthodes implémentées :

Pour cette 4ème partie j'ai décidé d'implémenter la fonction d'activation (fonction mathématique appliquée à un signal en sortie d'un neurone artificiel) ReLu qui est exécutée sur le même `dataTrain` et `dataTest` que la fonction d'activation Sigmoid afin de pouvoir les comparer dans les mêmes conditions. La fonction d'activation ReLu est une fonction qui remplace donc simplement toute valeur négative par 0 à l'aide de la fonction `numpy.maximum(0,x)` et sa dérivée `heaviside(x,1)`. Les résultats des fonctions d'activations sont disponibles sur le terminal.

J'ai également amélioré la GUI en remplaçant les tables des poids par des images où les zones les plus foncées représentent les poids des neurones les plus importants pour la classification. Cela a été implémenté grâce au module `QImage` où je crée une image. Les pixels de cette image sont déterminés par les poids du vecteur poids qui sont d'abord additionnés par la valeur absolue du minimum du vecteur poids pour obtenir des valeurs de 0 à max. La valeur est ensuite multipliée par la division de 255 par la valeur max du vecteur poids. Tout cela pour obtenir des valeurs de 0 à 255 afin de pouvoir déterminer la nuance de gris.

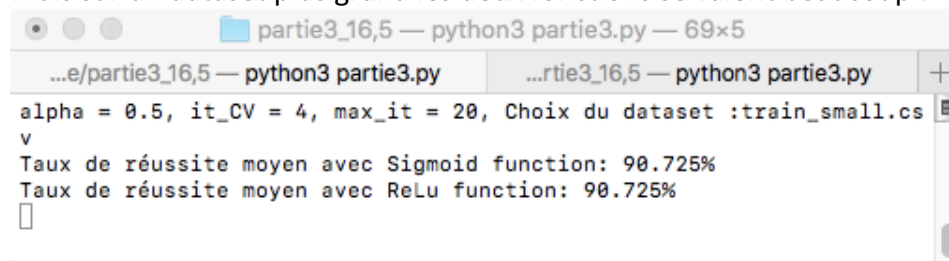
Comparaisons de performance :

Les fonctions d'activation Sigmoid et ReLu ont des résultats proches sur le plus petit dataset mais la fonction Sigmoid reste la plus performante :



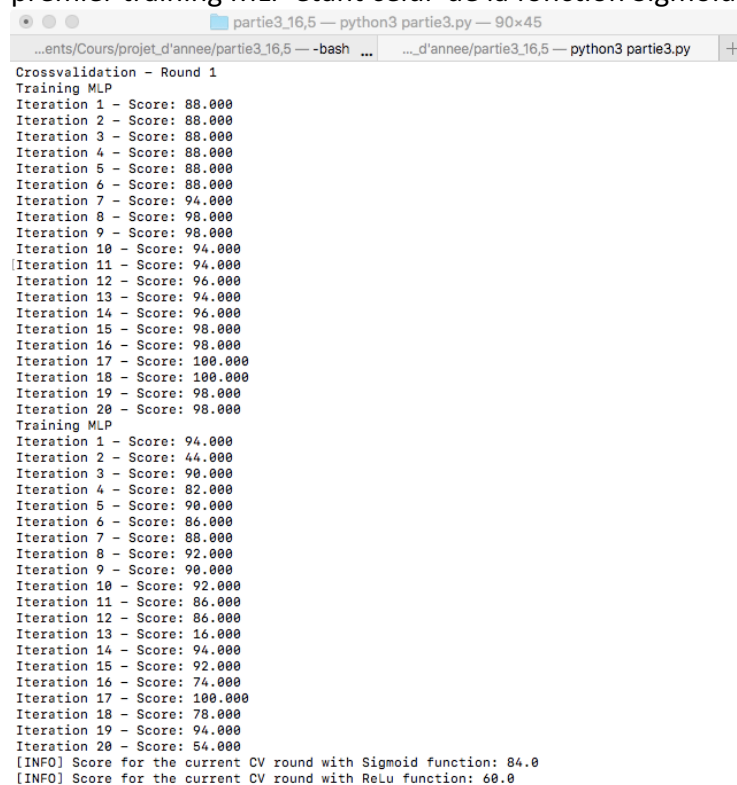
```
partie3_16,5 — python3 partie3.py — 68x5
.../partie3_16,5 — python3 partie3.py
alpha = 0.5, it_cv = 4, max_it = 20, Choix du dataset : train_tiny.csv
Taux de réussite moyen avec Sigmoid function: 92.5%
Taux de réussite moyen avec ReLu function: 89.0%
```

Mais sur un dataset plus grand les deux fonctions se valent beaucoup :



```
partie3_16,5 — python3 partie3.py — 69x5
...e/partie3_16,5 — python3 partie3.py
alpha = 0.5, it_cv = 4, max_it = 20, Choix du dataset : train_small.csv
Taux de réussite moyen avec Sigmoid function: 90.725%
Taux de réussite moyen avec ReLu function: 90.725%
```

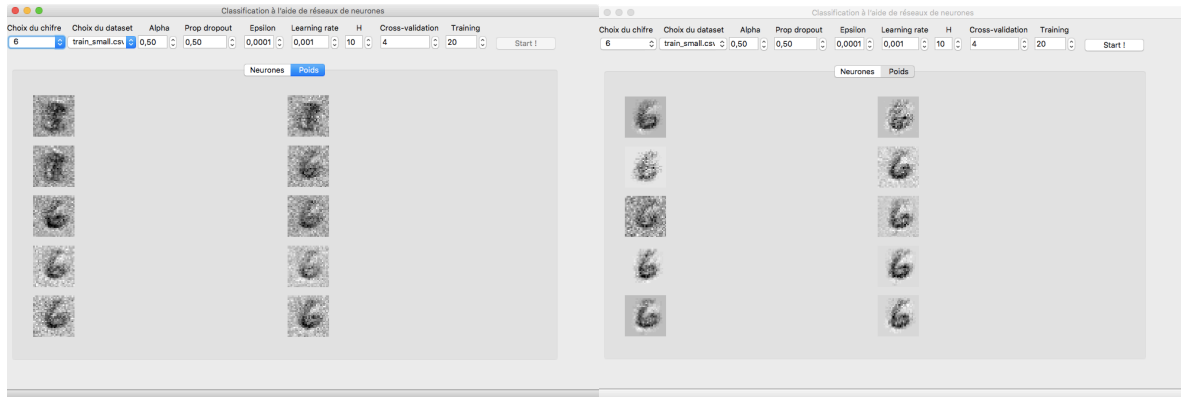
On peut également observer que la fonction ReLu produit des résultats très différents lors de l'entraînement du perceptron (le deuxième training MLP de la photo ci-dessous le premier training MLP étant celui de la fonction Sigmoid):



```
partie3_16,5 — python3 partie3.py — 90x45
...ents/Cours/projet_d'annee/partie3_16,5 — -bash ...
Crossvalidation - Round 1
Training MLP
Iteration 1 - Score: 88.000
Iteration 2 - Score: 88.000
Iteration 3 - Score: 88.000
Iteration 4 - Score: 88.000
Iteration 5 - Score: 88.000
Iteration 6 - Score: 88.000
Iteration 7 - Score: 94.000
Iteration 8 - Score: 98.000
Iteration 9 - Score: 98.000
Iteration 10 - Score: 94.000
Iteration 11 - Score: 94.000
Iteration 12 - Score: 96.000
Iteration 13 - Score: 94.000
Iteration 14 - Score: 96.000
Iteration 15 - Score: 98.000
Iteration 16 - Score: 98.000
Iteration 17 - Score: 100.000
Iteration 18 - Score: 100.000
Iteration 19 - Score: 98.000
Iteration 20 - Score: 98.000
Training MLP
Iteration 1 - Score: 94.000
Iteration 2 - Score: 44.000
Iteration 3 - Score: 90.000
Iteration 4 - Score: 82.000
Iteration 5 - Score: 90.000
Iteration 6 - Score: 86.000
Iteration 7 - Score: 88.000
Iteration 8 - Score: 92.000
Iteration 9 - Score: 90.000
Iteration 10 - Score: 92.000
Iteration 11 - Score: 86.000
Iteration 12 - Score: 86.000
Iteration 13 - Score: 16.000
Iteration 14 - Score: 94.000
Iteration 15 - Score: 92.000
Iteration 16 - Score: 74.000
Iteration 17 - Score: 100.000
Iteration 18 - Score: 78.000
Iteration 19 - Score: 94.000
Iteration 20 - Score: 54.000
[INFO] Score for the current CV round with Sigmoid function: 84.0
[INFO] Score for the current CV round with ReLu function: 60.0
```

Images de la nouvelle GUI:

Voici des exemples de ce que la nouvelles GUI permet de faire (images ou les zones les plus foncées représentent les poids des neurones les plus important pour la classification) à gauche une représentation exécuté à l'aide de la fonction Sigmoides à droite à l'aide de la fonction ReLu.



Conclusion :

En conclusion on peut voir que les résultats du projet sont assez concluant, on obtient des résultats de l'ordre des 90% pour les deux fonctions d'activations. On peut également désormais voir les poids des neurones les plus importants qui nous donne une bonne image des pixels qui sont les plus importants pour déterminer le contenu de l'image.