Licence Informatique 3e année

Apprentissage artificiel

Projet

Le projet consiste à écrire un programme de réseau de neurones multicouche avec apprentissage par descente de gradient. On peut se limiter à des neurones sigmoïdes. Vous êtes libre d'utiliser un apprentissage batch ou inline, d'utiliser la validation croisée ou non, etc. Le choix du langage de programmation est libre. *Il ne s'agit pas d'utiliser une librairie existante, mais d'écrire soi-même un programme.*

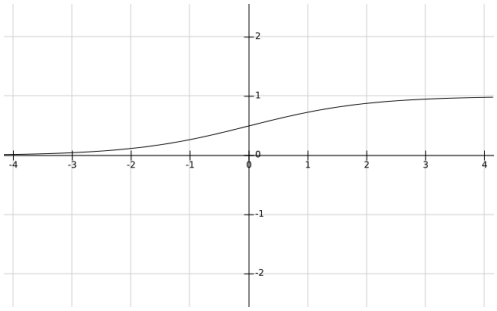
Le travail est à réaliser en binome. Il faudra présenter le travail réalisé durant la dernière semaine de TP (juste avant le début des stages). En plus du code, il faudra rendre un petit rapport (2 ou 3 pages) qui explique comment vous avez choisi de coder votre réseau (en quelques lignes, expliquez la structure du programme, éventuellement avec un diagramme), quel mode d'apprentissage vous avez choisi (stochastique ou batch ou mini-batch), quel processus de validation (10% des données pour validation, ou validation croisée), quelles données vous avez traitées, quels tests vous avez menés et quels résultats vous avez obtenus.

Tester votre programme sur des exemples simples. On peut par exemple utiliser des données d'apprentissage issues du portail européen des données ouvertes www.europeandataportal.eu/data/, du portail français www.data.gouv.fr/, du site de l'université de Californie archive.ics.uci.edu/ml/, du site de l'université Stanford snap.stanford.edu/data/, du site Kaggle www.kaggle.com/datasets, ou d'autres.

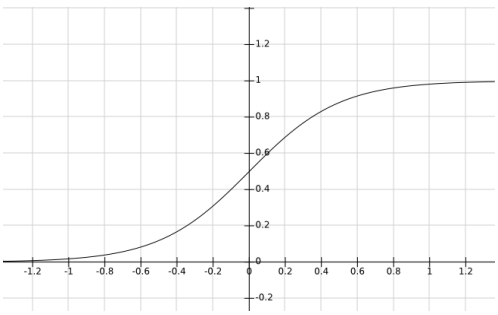
Quelques conseils pratiques

1. **Normaliser** les entrées : les données d'entrée doivent avoir des valeurs dans l'intervalle [0,1]. Pour les caractères, vous pouvez diviser leur code ASCII par la valeur du plus grand caractère présent (attention aux caractères non alphanumériques comme les émoticônes). Dans le cas des images, les couleurs de pixel doivent être ramenées entre 0 et 1. De même pour toutes les données numériques.

2. Le **paramètre** de la fonction sigmoïde peut être ajusté pour accélérer l'apprentissage.



$$\frac{1}{1+e^{-x}}$$

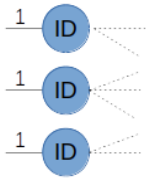


$$\frac{1}{1+e^{-4x}}$$

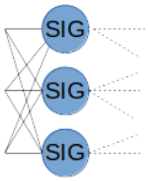
Rappel : si $y=\frac{1}{1+e^{-\lambda x}}$ alors $\frac{dy}{dx}=\lambda . y .(1-y)$

3. Si le **nombre de couches** est trop grand, lors de l'apprentissage, l'erreur va diminuer au fur et à mesure de la rétropropagation et sera quasiment nulle lors des modifications de poids sur les premières couches. Il faut donc se limiter à 4 ou 5 couches maximum.

4. Sur la **première couche**, ne pas appliquer les entrées chacune sur un neurone, sinon les sorties seront quasi nulles en général (un neurone qui n'a qu'une seule valeur d'entrée, multipliée par un poids faible, aura forcément une sortie faible).



Solution A : les neurones de la couche d'entrée n'ont qu'une seule entrée, avec des poids à 1 et ont pour fonction de transfert l'identité (la sortie est égale à l'entrée). Ces neurones ne font que dispatcher les valeurs d'entrées dans la couche suivante, sans modifier les valeurs.



Solution B : la première couche est comme les autres constituée de neurones sigmoïdes, avec des entrées pondérées par des poids variables, mais ces neurones prennent en entrée toutes les valeurs. En faisant la somme de pas mal de valeurs, on évite d'avoir toujours des valeurs très faibles en sortie de cette couche (et donc des couches suivantes).

5. Ne pas être trop gourmand sur l'**erreur acceptable**, pour éviter le sur-apprentissage. En classification, il suffit que le réseau discrimine correctement les exemples.