

IFT780: Travail pratique 1

Réseaux de neurones pleinement connectés

Dans ce devoir, vous devez implanter des réseaux de neurones pleinement connectés. Vous travaillerez tout d'abord avec un réseau linéaire (sans couche cachée) puis un réseau à deux couches cachées. Les parties sont séparées dans des `ipython notebook`, où certaines questions théoriques seront présentes. De plus, le code à implémenter sera indiqué dans les notebooks.

Tel qu'indiqué dans les notebooks, n'oubliez pas de télécharger les datasets en exécutant

```
dataset/get_datasets.sh
```

Les objectifs de ce devoir sont

1. Comprendre le fonctionnement d'un réseau de neurones simple (entraînement et prédiction)
2. Comprendre l'utilisation de données de validation pour ajuster des hyperparamètres.
3. Vectoriser du code afin de manipuler des « mini-batchs » de données sans boucle for.
4. Implanter un classificateur multi-classes de type SVM
5. Implanter un classificateur multi-classes utilisant un Softmax
6. Implanter un réseau de neurones multi-classes à N couches cachées

1. `tp1_hinge.ipynb` (3 points)

Implanter et tester un classifieur linéaire de type SVM avec la fonction de perte « Hinge Loss ».

2. `tp1_softmax.ipynb` (3 points)

Implanter et tester un réseau logistique linéaire ayant un *softmax* et une entropie croisée en sortie.

3. `tp1_simple_neural_net.ipynb` (0 points)

Voyez comment créer un simple réseau de neurones multicouches, comment effectuer une propagation avant, une propagation arrière, effectuer une descente de gradient et l'utilisation d'une « cache ». Bien qu'aucun point n'est attribué à ce notebook, il est primordial d'en comprendre le contenu avant de vous attaquer au 4e notebook.

4. `tp1_neural_net.ipynb` (4 points)

Le but de ce notebook est d'enchâsser dans des classes les différents éléments vus au point 3. Comme ailleurs, le code de la propagation avant et de la rétropropagation de la couche dense et du calcul de la perte doit être vectorisé. Donc attention aux boucles for!