

Introduction à l'apprentissage profond

Mathieu Lefort

18 et 19 septembre 2023

Ce TP peut être fait en binôme (vous indiquerez alors la répartition des tâches que vous avez adopté). Vous rendrez un compte-rendu incluant votre code commenté et vos réponses aux questions à la fin de chacune des 2 séances de TPs et la version finale pour le 8 octobre. Pensez à indiquer votre(vos) nom(s) dans le fichier déposé.

1 Objectif

L'objectif de ce projet est d'implémenter les modèles du MLP et du CNN vus en cours, de comprendre leur fonctionnement et de prendre en main le framework PyTorch.


2 Installation

Vous devez installer python, pytorch et numpy (et matplotlib si vous voulez faire des affichages). Pour l'installation de PyTorch regardez <https://pytorch.org/get-started/locally/>. Pour vérifier l'installation, lancez une console python et tapez l'instruction `import torch`.

3 PyTorch

Vous trouverez une liste des fonctions disponibles ici : <https://pytorch.org/docs/stable/index.html>. Regardez ce micro tutoriel : http://pytorch.org/tutorials/beginner/pytorch_with_examples.html. Les concepts clés sont présentés ici : <https://pytorch.org/tutorials/beginner/basics/intro.html> (les chapitres 3 et 7 peuvent être passés).

4 Données

Vous avez à votre disposition le jeux de données MNIST qui contient des images (de taille 28*28) de chiffres manuscrits (par exemple une des images de 5 de la base : ). La base de données est structurée comme suit : ((tableau_image_apprentissage, tableau_label_apprentissage), (tableau_image_test, tableau_label_test)). Les images sont stockées sous la forme d'un vecteur de 28*28 valeurs et les labels sont sous la forme d'un codage *one-hot* (i.e. si le chiffre représenté sur l'image est un 5, son label sera : [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]).

L'entrée de chaque réseau aura donc 784 entrées (28×28 pixels) et 10 sorties (la i^{ème} sortie représentant à quel point le réseau reconnaît le chiffre i dans l'image).

5 Partie 1 : Perceptron

Vous avez à votre disposition 2 fichiers d'implémentation du (même) perceptron :

1. PERCEPTRON_PYTORCH.PY qui utilise uniquement les tenseurs
2. PERCEPTRON_PYTORCH_DATA_AUTO_LAYER_OPTIM.PY qui utilise la plupart des outils de Pytorch (les *dataloaders*, la différenciation automatique, les couches et les optimiseurs)

- Indiquer et expliquer la taille de chaque tenseur dans le fichier `PERCEPTRON_PYTORCH.PY` fourni.

6 Partie 2 : *Shallow network*

Dans cette partie vous implémenterez l'algorithme du perceptron multi-couches avec une seule couche cachée et une sortie linéaire.

- En vous appuyant sur le fichier `PERCEPTRON_PYTORCH_DATA_AUTO_LAYER_OPTIM.PY` fourni, implémenter ce réseau en utilisant les outils de Pytorch.
- Trouver des hyperparamètres (η et le nombre de neurones de la couche cachée) permettant d'avoir une bonne performance. Décrire précisément la méthodologie que vous avez utilisé et l'influence observée de chaque hyperparamètre sur la performance.

7 Partie 3 : *Deep network*

Maintenant que vous maîtrisez Pytorch, vous allez pouvoir l'utiliser pour de l'apprentissage profond.

- Implémenter un réseau de neurones profond (i.e. avec au moins 2 couches cachées) en utilisant les outils fournis par Pytorch.
- Trouver des hyperparamètres (η , le nombre de couches cachées et de neurones par couche cachée et la taille des mini batches) permettant d'avoir une bonne performance. Décrire précisément la méthodologie que vous avez utilisé et l'influence observée de chaque hyperparamètre sur la performance.

8 Partie 4 : *CNN*

Maintenant que vous maîtrisez les MLP, vous allez pouvoir passer aux réseaux convolutifs (mieux adaptés aux images).

- Implémenter un réseau de neurones convolutif simple (en prenant comme base LeNet5) en utilisant les outils fournis par Pytorch.

9 Partie 5 : Pour aller plus loin (optionnel)

Si vous avez le temps et l'envie vous pouvez chercher à améliorer les performances de toute ou partie des sections précédentes en modifiant les paramètres suivants :

- Utiliser la fonction de coût cross entropie.
- Tester d'autres fonctions d'activation pour les neurones (ReLU en particulier).
- Utiliser une méthode de gradient plus efficace que la SGD comme Adam ou Adagrad.
- Utiliser d'autres méthodes d'initialisation des poids (poids gaussien, initialisation Xavier par exemple)
- Utiliser un ResNet50 pré entraîné