Introduction à l'apprentissage profond

Mathieu Lefort

10 & 12 septembre 2024

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 13 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 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 jeu 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 784 (=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). Il s'agit donc d'un problème de classification, il faudra choisir une fonction de coût adaptée.

5 Partie 1 : Perceptron

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

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.

— Implémentez ce réseau en utilisant les outils de PyTorch.

— Décrivez très précisément la méthodologie à utiliser (pensez en particulier à faire un jeu de validation pour éviter le sur-apprentissage et rappelez vous que les poids initiaux sont aléatoires) pour trouver les bons hyperparamètres (η, le nombre de neurones de la couche cachée et la taille des mini batchs) pour avoir la meilleure performance.

— Trouvez des hyperparamètres permettant d'avoir une bonne performance (vous pouvez adapter en pratique le nombre de tests réalisés prévu par votre méthodologie à votre puissance de calcul disponible, en expliquant/justifiant sur quels tests vous vous êtes concentrés) et décrire l'influence observée de chaque hyperparamètre sur la performance.

7 Partie 3 : *Deep network*

Maintenant que vous maitrisez PyTorch, vous allez pouvoir l'utiliser pour de l'apprentissage profond.

— Implémentez un réseau de neurones profond (i.e. avec au moins 2 couches cachées) en utilisant les outils fournis par PyTorch.

— Trouvez des hyperparamètres (η, le nombre de couches cachées et de neurones par couche cachée et la taille des mini batchs) permettant d'avoir une bonne performance (vous pouvez adapter en pratique le nombre de tests réalisés prévu par votre méthodologie à votre puissance de calcul disponible, en expliquant/justifiant sur quels tests vous vous êtes concentrés) et décrire l'influence observée de chaque hyperparamètre sur la performance.

8 Partie 4 : *CNN*

Maintenant que vous maitrisez les MLP, vous allez pouvoir passer aux réseaux convolutifs mieux adaptés aux images (attention les données sont sous forme d'un vecteur de taille 784, il faut leur redonner la taille d'une image 28×28).

— Implémentez un réseau de neurones convolutif (en utilisant les outils fournis par PyTorch) permettant d'avoir de bonnes performances. Vous pouvez par exemple vous appuyer sur un réseau simple comme LeNet5 ou faire un *fine tuning* d'un réseau pré entraîné comme un ResNet. Expliquez et justifiez votre démarche (adaptée à votre puissance de calcul).