TRAVAUX PRATIQUE

INTRODUCTION AU DEEP LEARNING

Théo Lopès-Quintas

BPCE Payment Services,
Université Paris Dauphine

Janvier 2024

SÉANCE 1 - RÉSEAUX DENSES

Séance 1 - Réseaux denses

Consignes

Quelques consignes et conseils pour l'ensemble des sujets proposés :

- Pour tous les réseaux de neurones que vous définirez, essayez de calculer à la main le nombre de paramètres de votre réseau de neurones puis contrôler avec la méthode *summary*.
- Ne négligez pas la qualité de code ! Le travail avec les réseaux de neurones nécessite de rédiger plus de code que les algorithmes de Machine Learning classique, cela peut rendre le notebook plus difficile à lire et à retravailler.
- Vous pouvez choisir votre propre sujet à condition de le faire valider en amont

Sauf mention contraire, on utilisera le dataset MNIST pour entraîner les réseaux de neurones denses.

Séance 1 - Réseaux denses

A - Hyper paramètres

Nous souhaitons mesurer l'impact du choix des hyper paramètres sur les performances d'un réseau de neurones dense. Pour cela, choisir un ou deux hyper paramètres à tester parmi :

- **Learning rate** : comment se comporte l'entraînement du réseau de neurones selon différentes valeurs du learning rate ?
- **Profondeur** : comment se comporte l'entraînement du réseau de neurones selon différentes profondeur du réseau de neurones ?
- **Époques** : comment se comporte l'entraînement du réseau de neurones sur des temps d'entraînement long ?

On peut trouver d'autres questions, il faut les faire valider pour pouvoir les traiter.

Consignes

- Annoncer avant de comparer ce à quoi on s'attend
- Produire des résultats simplement interprétable et lisible
- Identifier et présenter les limites de la comparaison réalisée

Nous souhaitons mesurer l'impact de la distribution des valeurs initiales des poids sur l'entraînement du réseau de neurones dense. Pour cela, on se propose de comparer plusieurs distribution :

- **Glorot** : Normal et uniforme
- **He** : Normal et uniforme
- **Random** : Normal et uniforme

Pour être complet, on réalisera l'étude pour la fonction d'activation ReLU et, au choix, sigmoid ou tanh.

Consignes

- Annoncer avant de comparer ce à quoi on s'attend
- Produire des résultats simplement interprétable et lisible
- Identifier et présenter les limites de la comparaison réalisée

Les réseaux de neurones sont particulièrement sensible au sur-apprentissage à cause de la très forte flexibilité offerte par les nombreux paramètres. Une première manière de régulariser le réseau de neurones est d'exploiter une régularisation $\mathcal{L}_2$ comme pour une régression linéaire. Cela consiste à modifier la fonction de perte :

$$\mathcal{L}_\lambda(w) = \mathcal{L}(w) + \frac{\lambda}{2}\|w\|_2^2 \quad \text{avec } \lambda \geqslant 0$$

Pour l'appliquer il faut, couche par couche, choisir la valeur de λ que l'on veut appliquer. On souhaite mesurer l'impact de la régularisation sur l'apprentissage d'un réseau de neurones : proposer une étude répondant au problème.

Consignes

- Annoncer avant de comparer ce à quoi on s'attend
- Produire des résultats simplement interprétable et lisible
- Identifier et présenter les limites de la comparaison réalisée

Nous souhaitons entraîner un réseau de neurones pour résoudre un problème de régression. Pour cela, nous considérerons le dataset sklearn.datasets.fetch_california_housing. Après avoir prit connaissance du problème que l'on cherche à résoudre, et réalisé un rapide modèle Machine Learning classique, on entraînera un réseau de neurones. A noter qu'il faudra modifier :

- **Couche input** : l'input n'a plus besoin d'être aplati, donc il faudra renseigner la dimension de l'input dans la première couche
- **Couche output** : l'output n'a pas 10 classes, donc il faudra ne placer qu'un seul neurone en sortie
- **Loss** : la fonction de perte doit être une fonction de perte de régression, par exemple la MSE

Consignes

- Réaliser une étude rapide et claire du dataset utilisé
- Entraîner avec des performances correctes un algorithme de Machine Learning classique
- Entraîner avec des performances correctes le réseau défini

SÉANCE 2 - RÉSEAUX CONVOLUTIONNEL

Séance 2 - Réseaux convolutionnel

Consignes

Quelques consignes et conseils pour l'ensemble des sujets proposés :

- Pour l'ensemble des réseaux de neurones que vous définirez, essayez de calculer à la main le nombre de paramètre de votre réseau de neurones puis contrôler avec la méthode *summary*.
- Ne négligez pas la qualité de code ! Le travail avec les réseaux de neurones nécessite de rédiger plus de code que les algorithmes de Machine Learning classique, cela peut rendre le notebook plus difficile à lire et à retravailler.
- Vous pouvez choisir votre propre sujet à condition de le faire valider en amont

Sauf mention contraire, on utilisera le dataset Fashion MNIST pour entraîner les réseaux de neurones denses.

Séance 2 - Réseaux convolutionnel

A - Hyper paramètres

Nous souhaitons mesurer l'impact du choix des hyper paramètres sur les performances d'un réseau de neurones convolutionnel. Pour cela, choisir un ou deux hyper paramètres à tester parmi :

- **Taille de convolution** : comment se comporte l'entraînement du réseau de neurones selon différentes tailles de filtre de convolution ?
- **Fonction d'activation** : comment se comporte l'entraînement du réseau de neurones selon différentes fonction d'activation ?
- **Pooling** : comment se comporte l'entraînement du réseau de neurones selon différentes type de pooling ?

On peut trouver d'autres questions, il faut les faire valider pour pouvoir les traiter.

Consignes

- Annoncer avant de comparer ce à quoi on s'attend
- Produire des résultats simplement interprétable et lisible
- Identifier et présenter les limites de la comparaison réalisée

L'article *Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift* écrit par Sergey Ioffe et Christian Szegedy introduit une nouvelle régularisation pour un réseau de neurones. On se propose dans ce sujet de l'étudier pour mieux comprendre son fonctionnement. On pourra faire une liste de l'ensemble des *shift* existant en Machine Learning, présenter des exemples et expliquer comment on peut s'en prémunir, si c'est possible.

Consignes

- Lire l'article et expliquer le fonctionnement de cette couche
- Produire des résultats simplement interprétable et lisible
- Identifier et présenter les limites de la comparaison réalisée

On souhaiterai pouvoir définir nous-mêmes un réseau de neurones convolutionnel résiduel. Pour cela, on s'appuiera sur l'approche fonctionnel de Keras pour définir soi-même un réseau de neurones résiduels.
Egalement, on utilisera la couche de BatchNormalization après l'avoir étudié d'après le papier présenté au sujet précédent. On expliquera rapidement le fonctionnement.

Consignes

- Ne pas négliger la qualité de code
- Essayer de généraliser le plus possible la création d'un ResNet
- Entraîner avec des performances correctes le réseau défini

Suite à thread Twitter en 2020, l'idée de remplacer une couche de convolution part une couche de similarité cosinus mise à la puissance p a été discuté puis implémenté. Voici les points majeur que l'on connait pour le moment :

1. On ne gagne pas forcément en performance : vitesse de calcul ou précision
2. On gagne énormément en taille de modèle : un modèle avec SCS a beaucoup moins de paramètres qu'un réseau convolutionnel classique
3. Il faut plutôt utiliser un couche de pooling qui travaille en valeur absolue

Il y a bien sûr d'autre point que l'on connait, on souhaite se focaliser sur ces trois-là.

Consignes

- Expliquer mathématiquement comment fonctionne SCS et faire le lien avec les points 1 et 3
- Comprendre comment ajouter ces nouvelles couches qui ne sont pas implémentés pour le moment sur Keras
- Entraîner avec des performances correctes un réseau avec SCS