Intelligence Artificielle - Deep Learning
Feuille 2

Réseaux de neurones (II)
`tensorflow` et `keras`

Présentation

Dans ce TD, nous allons essentiellement utiliser `tensorflow` [1] et `keras` [2].

Le corpus que nous allons utiliser est le même que pour le TD1. Il s'agit de `mnist` [3].

Le but est de tester des architectures différentes (réseaux de neurones classiques, CNN), des hyper-paramètres différents (nombre de couches, taux d'apprentissage, etc) et de distinguer clairement les différentes étapes (training, validation, test).

Nous allons donc essentiellement reprendre les mêmes exercices que le TD1. Nous allons juste tout développer en utilisant `keras`.

Rappelons que le corpus `mnist` est composé de 70000 images de chiffres manuscrits. Les images sont de taille 28×28 pixels. Chaque pixel est codé par un nombre entre 0 et 255 correspondant au niveau de gris.

Commencez donc par télécharger le fichier `lab2_skeleton.py` disponible à l'adresse du cours :

`https://masterinfo.emi.u-bordeaux.fr/wiki/doku.php?id=ia`.

Vous devez également activer l'environnement `tensorflow` en exécutant l'instruction

```
source /net/ens/DeepLearning/python3/tensorflow2/bin/activate
```

Quelques rappels.

Rappelons qu'un réseau de neurones est composé d'une suite de couches successives. Chaque suite est un ensemble de neurones formels. Les architectures de base considèrent que les signaux ne circulent que d'une couche vers la suivante.

Chaque neurone dans la $k^{ème}$ couche est connecté à tous les neurones de la $(k-1)^{ème}$ couche et les neurones d'une couche forment un ensemble indépendant.

Un réseau de neurones est composé de :

- une couche d'entrée x
- un (éventuel) nombre arbitraire de couches cachées
- une couche de sortie y
- un ensemble de poids W et de biais b
- un ensemble de fonctions d'activation, une par couche.

1. `https://www.tensorflow.org`
2. `https://keras.io`
3. `http://yann.lecun.com/exdb/mnist/`

Exercice 1. Un premier réseau de neurones

Dans cet exercice, nous allons définir un réseau avec une couche de sortie avec deux neurones. Le but est toujours de faire une classification binaire et arriver à faire la différence entre le chiffre 5 et les autres chiffres.

1. Modifier le fichier pour implémenter ce réseau de neurones. Quelles fonctions d'activation utiliseriez-vous pour la couche cachée ? Pour la couche de sortie ?
2. Entrainer le réseau avec juste 10 `epchos`, le tester et mesurer sa précision. Peut-on améliorer encore la précision sans changer l'architecture du réseau ?
3. Modifier le réseau pour rajouter une couche cachée avec un seul neurone. Qu'observez-vous ?

Exercice 2. Un réseau pour reconnaitre tous les chiffres

À présent, nous allons adapter le réseau précédent pour reconnaire tous les 10 chiffres.

1. Expliquer, avec une figure, l'architecture du nouveau réseau.
2. Redéfinir, dans le programme, la nouvelle architecture.
3. Entrainer et tester le nouveau réseau. Refaire les mêmes experimentations que dans l'exercice 1. Changer d'algorithme d'optimisation.

Exercice 3. Un réseau CNN pour reconnaitre tous les chiffres

Même exercice mais en utilisant une architecture en CNN.

1. Définir un réseau de neurones à convolution avec :
   - une couche de convolution
   - une couche de maxpooling
   - un réseau "fully connected" à plusieurs couches
2. Entrainer le modèle et le tester.
3. Choisir les bons hyper-paramètres pour atteindre une précision d'au moins 99%.