TP2 - Reconnaissance de chiffres manuscrits

masco programmation

4 mars 2020

1 Objectif

L'objectif de ce projet est de programmer un réseau convolutionnel permettant de reconnaître des chiffres manuscrits.

2 Données

Les données permettant d'entraîner le modèle sont composées de couples (x_i, y_i) où x_i est une image de 28×28 pixels représentant chacune un chiffre de 0 à 9 et y_i est le chiffre correspondant à l'image. Chaque pixel est représenté par un nombre compris entre 0 et 255 qui indique le niveau de gris du pixel. La valeur 0 correspond à la couleur blanche et la valeur 255 au noir.

Les données sont divisées en *données d'apprentissage* qui vont servir à entraîner le modèle et en *données de test*, qui serviront à évaluer le modèle sur les données non vues lors de l'apprentissage.

On dispose de 60 000 couples (x, y) pour les données d'apprentissage et de 10 000 couples pour les données de test. Les données se trouvent dans quatre fichiers :

- `train_image_file.txt`
- `train_label_file.txt`
- `test_image_file.txt`
- `test_label_file.txt`

Les deux premiers correspondent aux données d'apprentissage, avec, dans le premier fichier les images et, dans le second, les catégories correspondant (les chiffres entre 0 et 9). Ces fichiers sont *alignés* : la première ligne du fichier `train_image_file.txt` correspond à une image (une séquence de 784 nombres) et la première ligne de `train_label_file.txt` au chiffre correspondant.

Vous pourrez télécharger les données à partir de la page du cours.

3 Modèle

Le modèle utilisé dans ce projet est un réseau convolutionnel dont le niveau d'entrée est une matrice de dimension 28×28 et le niveau de sortie un vecteur de dimension 10, chaque dimension corespondant à une des 10 classes de sortie.

L'implémentation pourra être réalisée à l'aide du logiciel Keras, sur la plateforme colab. Mais vous pouvez aussi utiliser d'autres outils si vous le souhaitez.

Le réseau utilisé est un réseau de type `Sequential` composé d'une succession de couches de convolution et de pooling. La dernière couche est une couche dense qui produit le niveau de sortie.

3.1 Couche convolutionnelle

Une couche de convolution bi-dimensionnelle est créée à l'aide de la méthode `Conv2D`.

Vous trouverez dans la documentation de keras la liste complète des arguments de cette méthode, en voici les principales :

- `filters:` nombre de filtres de la couche de convolution.
- `kernel_size:` taille du noyau, représenté sous la forme d'un tuple ou d'une liste de deux entiers représentant la hauteur et la largeur du noyau. Si un seul entier est indiqué, le noyau est un carré dont cette valeur est le côté.
- `strides:` taille du pas, représenté sous la forme d'un tuple ou d'une liste de deux entiers représentant la valeur verticale et horizontale du pas.
- `activation:` fonction d'activation.

La première couche convolutionnelle d'un réseau doit indiquer de plus le format du niveau d'entrée. Cela se fait par l'intermédiaire de l'argument `input_shape`, qui prend pour valeur un tuple de trois entiers. Une valeur de `input_shape` qui vaut `(28, 28, 1)` correspond à des images de 28×28 pixels où la couleur de chaque pixel est un entier (le niveau de gris, par exemple).

3.2 Couche de pooling

Une couche de max pooling est réalisée à l'aide de la fonction `MaxPooling2D`, dont le paramètre principal est `pool_size` qui est un tuple de deux entier indiquant la fenêtre de pooling.

3.3 Exemple

Voici un exemple simple de réseau convolutionnel bi-dimensionnel :

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D

model.add(Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1),
                 activation='relu',
                 input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

La couche `flatten` a pour seul but d'applatir un niveau afin de le représenter sous la forme d'un vecteur.

4 Ce qu'il faut faire

1. Ouvrir et exécuter le notebook `MNIST_CONV_squelette.ipynb` qui contient le squelette d'un notebook.
2. Déterminer avec l'enseignant une étude à mener autour de ce réseau et de ces données (voir la section ci-dessous)
3. Réaliser l'étude sous la forme d'un notebook, à rendre pour le 16 mars 2020.

4.1 Quelques pistes à creuser

Voici quelques pistes possibles pour une étude à mener autour de ce réseau. Elles ne sont pas exclusives, vous pouvez choisir d'en explorer plusieurs :

— Hyperparamètres. Quelle est l'influence des différents hyperparamètres sur les performances ?
— Augmentation de données. Il est possible de manipuler les données d'apprentissage de manière à générer de nouveaux exemples, en réalisant, par exemple, des translations des images, vers le haut, le bas, la gauche ou la droite. Quelle est l'influence de cette augmentation des données sur les images initiales ? sur les images ayant subi une translation ?
— Niveaux de gris. Les images qui constituent les données indiquent pour chaque pixel, son niveau de gris, une valeur comprise entre 0 et 255. Est-il nécessaire d'avoir une telle gradation du niveau de gris ? que se passe-t-il lorsque diminue les valeurs possibles de niveau de gris ?
— Bruit. On peut de manière aléatoire modifier la valeur de certain pixels dans l'image. Comment est ce que les performances décroissent en fonction du taux de pixels modifiés. Que se passe-t-il lorsqu'on ajoute du bruit dans les données d'apprentissage ?
— Caractères inconnus. Que se passe-t-il lorsqu'on présente au réseau une entrée qui ne correspond pas à un chiffre ? serait-il possible de créer une nouvelle classe qui corresponde aux images qui ne sont pas des chiffres ?