Université Cadi Ayyad

Ecole Nationale des sciences appliquées-SAFI

Année Universitaire 2021-2022

Filière : Génie Informatique et Intelligence Artificielle

Projet : Machine learning

Classification de la base de données MNIST avec python

Projet réalisé par :

- Aaba Abdelghani
- El Ouarroudi Monir
- Darfi Oussama

Sous la supervision de :

- Mr Mediafi Mohammed

Introduction Générale

Le Machine Learning est une branche de l'intelligence artificielle qui a pour but de donner la possibilité aux ordinateurs d'apprendre. Un ordinateur n'est pas intelligent, il ne fait qu'exécuter des tâches. On lui décrit sous forme de programmes quoi faire et comment le faire. C'est ce qu'on appelle la programmation.

Le machine Learning traite des sujets complexes où la programmation traditionnelle trouve ses limites. Construire un programme qui conduit une voiture serait très complexe voire impossible. Cela étant dû aux nombres infinis des cas possibles à traiter… ML traite cette problématique différemment. Au lieu de décrire quoi faire, le programme apprendra par lui même comment conduire en "observant" des expérimentations.

Machine Learning : Donner la possibilité à l'ordinateur d'apprendre sans être programmé.

En fonction des données d'expérimentation que prendra l'algorithme d'apprentissage en entrée, il déduira par lui même une hypothèse de fonctionnement. Il utilisera cette dernière pour de nouveaux cas, et affinera son expérience au fil du temps.

CHAPITRE 1: Partie théorique

1.1- Datasets en machine learning :

Le dataset est un outil numérique qui intègre plusieurs données. Il peut s'agir de fichiers vidéo, d'images, de textes, de sons ou même de statistiques. Leur regroupement forme un ensemble. Dans le domaine du machine learning (ou apprentissage automatique), le dataset demeure indispensable pour la création de modèles qui, eux-mêmes, permettent l'expression d'algorithmes à travers différents usages et résultats :

- les fonctions prédictives et les tendances prévisionnelles d'un secteur ;
- les besoins d'une cible ou d'un consommateur ;
- l'analyse d'un fichier image ;
- la gestion des anomalies de l'équipement ;
- la traduction automatique…

Le dataset constitue donc une mécanique essentielle en intelligence artificielle (IA). En apprentissage supervisé ou non supervisé, les champs d'application s'étendent de la cybersécurité à l'économétrie, en passant par la bio-informatique ou même la segmentation d'images, pour ne citer que quelques exemples. On distingue différentes catégories d'outils. Les plus connues demeurent les datasets d'entraînement, de test et de validation.

1.2- La reconnaissance d'images :

La vision par ordinateur, le domaine concernant les machines capables de comprendre des images et des vidéos, est l'un des sujets les plus brûlants de l'industrie technologique. La robotique, les voitures autonomes et la reconnaissance faciale reposent toutes sur la vision par ordinateur pour fonctionner. Au cœur de la vision par ordinateur se trouve *reconnaissance d'images*, la tâche de reconnaître ce que représente une image.

Avant d'effectuer toute tâche liée aux images, il est presque toujours nécessaire de traiter d'abord les images pour les rendre plus appropriées comme données d'entrée. En particulier sur la façon dont nous pouvons convertir des images à partir de fichiers JPEG ou PNG en données utilisables pour nos réseaux de neurones.

1.3- La base de données MNIST :

L'acronyme MNIST (*Modified* ou *Mixed National Institute of Standards and Technology)*, est une base de données de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenue un test standard. Elle regroupe 60,000 images d'apprentissage et 10,000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté

CHAPITRE 2: Partie experimental et résultats

- Chargement des bibliothéques

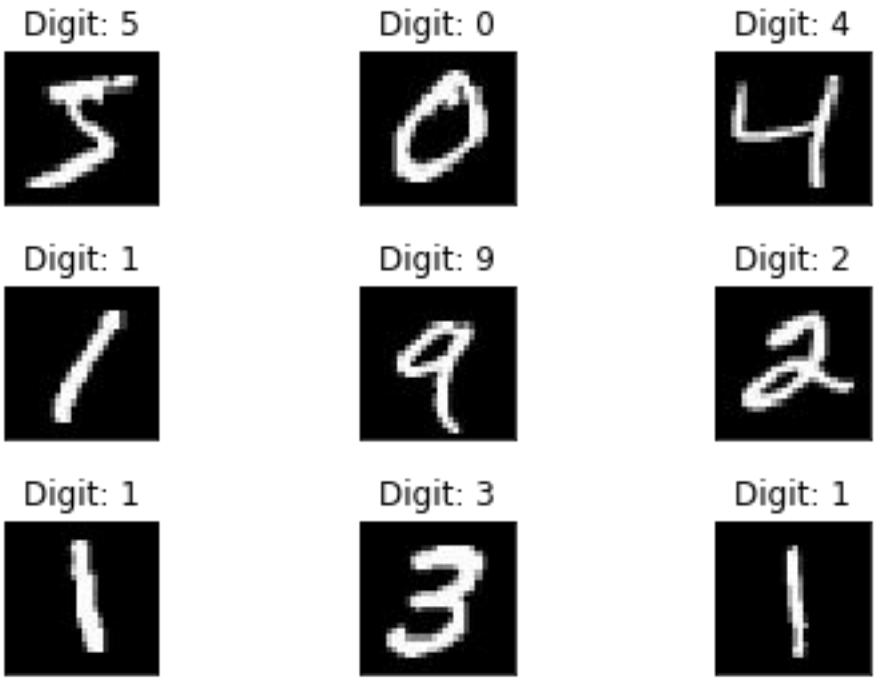
```python
# importer pour le traitement des listes et pour l'affichage
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# keras : importer la base de données and creation de notre reseau
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

- Téléchargement de la base de données et l'affichage des exemples :

```python
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```python
fig = plt.figure()
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.tight_layout()
  plt.imshow(X_train[i], cmap='gray', interpolation='none')
  plt.title("Digit: {}".format(y_train[i]))
  plt.xticks([])
  plt.yticks([])
```



- Adaptation des dimensions des données pour les utiliser :

```
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
print("Matrice d'entrainement", X_train.shape)
print("Matrice d'evaluation", X_test.shape)
```

```
Matrice d'entrainement (60000, 784)
Matrice d'evaluation (10000, 784)
```

```python
# one-hot codage avec numpy pour les utiliser dans le model keras
n_classes = 10
print("Dimensions avant one-hot encoding: ", y_train.shape)
Y_train = np_utils.to_categorical(y_train, n_classes)
Y_test = np_utils.to_categorical(y_test, n_classes)
print("Dimensions avant one-hot encoding: ", Y_train.shape)
```

```
Dimensions avant one-hot encoding:  (60000,)
Dimensions avant one-hot encoding:  (60000, 10)
```

- Création de notre réseau de neurones et spécifier le nombre de neurones pour chaque couche et les fonctions d'activations :

```
# creation d'un reseau de 3 couches qui contient 512, 512, 10 neurones
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))

model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))

model.add(Dense(10))
model.add(Activation('softmax'))
```

- L'entrainement du reseau en utilisant la fonction fit() de bibliothéque keras :

```
# definition des parametres d'optimisation et la fonction de loss
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

# l'entrainement du model
history = model.fit(X_train, Y_train,
          batch_size=128, epochs=15,
          verbose=2,
          validation_data=(X_test, Y_test))
```

```
Epoch 1/15
469/469 - 4s - loss: 0.0444 - accuracy: 0.9855 - val_loss: 0.0648 - val_accuracy: 0.9790 - 4s/epoch - 9ms/step
Epoch 2/15
469/469 - 3s - loss: 0.0410 - accuracy: 0.9866 - val_loss: 0.0663 - val_accuracy: 0.9789 - 3s/epoch - 7ms/step
Epoch 3/15
469/469 - 3s - loss: 0.0371 - accuracy: 0.9879 - val_loss: 0.0626 - val_accuracy: 0.9804 - 3s/epoch - 7ms/step
Epoch 4/15
469/469 - 3s - loss: 0.0364 - accuracy: 0.9885 - val_loss: 0.0675 - val_accuracy: 0.9788 - 3s/epoch - 7ms/step
Epoch 5/15
469/469 - 3s - loss: 0.0327 - accuracy: 0.9896 - val_loss: 0.0732 - val_accuracy: 0.9789 - 3s/epoch - 7ms/step
...
Epoch 14/15
469/469 - 4s - loss: 0.0214 - accuracy: 0.9927 - val_loss: 0.0619 - val_accuracy: 0.9814 - 4s/epoch - 9ms/step
Epoch 15/15
469/469 - 5s - loss: 0.0203 - accuracy: 0.9929 - val_loss: 0.0641 - val_accuracy: 0.9822 - 5s/epoch - 10ms/step
```

- Calculer la précision de notre réseau après l'entrainement :

```python
loss_and_metrics = model.evaluate(X_test, Y_test, verbose=2)

print("Test Loss", loss_and_metrics[0]*100, "%")
print("Test Accuracy", loss_and_metrics[1]*100, "%")
```

```
Test Loss 6.409822404384613 %
Test Accuracy 98.2200026512146 %
```