手寫辨識說明文件

LIU,TZU-CHUN

109 級畢業專題

3/11,2023

簡介

簡介

報告內容主要會用來介紹以 DNN 神經網路做成的手寫辨識程式碼，以及用 CNN 所做成的手寫辨識程式碼。

DNN 版本程式碼結構

Import package

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
```

讀入 MNIST 資料集

```
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Normalization

```
x_train = x_train.reshape(60000, 784)/255
x_test = x_test.reshape(10000, 784)/255
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

將每個數據都拉平成 1*784 的矩陣
p.s. 除以 255 是在做 normalization 的動作（讓數值能介在 0 /和 1 之間）

建構 DNN 神經網路

```
# 以Sequential的方式建構自己的神經網路
model = Sequential()
# model 中加入
（神經網路模式CNN/RNN/DNN（神經元數量,維度（這邊是28*28）,激發函數））
model.add(Dense(100, input_dim=784, activation='relu'))
# 第二層
model.add(Dense(100, activation='relu'))
# 輸出層, 10 代表說最後輸出10個神經元（因為數字10種 1~10）
model.add(Dense(10, activation='softmax'))
# loss 的意思是 loss function, SGD(gradient descent)
model.compile(loss='mse', optimizer=SGD(learning_rate=1),
              metrics=['accuracy'])
```

建構神經網路完成後可以用 model.summary() 查看 model 的樣子

```
# batch size 分批的概念
# epochs 學習幾次的意思
model.fit(x_train, y_train, batch_size=100, epochs=50)
```

訓練模型

```
# 預測
y_predict = np.argmax(model.predict(x_test), axis=-1)

# 測試
n = 9453
print('神經網路預測是:', y_predict[n])
plt.imshow(x_test[n].reshape(28,28), cmap='Greys')
```

開始使用模型預測、並測試

CNN 版本程式碼結構

Import Package

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
```

匯入套件

讀入 MNIST 資料集

```
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Normaliztion

```
# 每張圖就是一個矩陣，所以是 28*28,1
x_train = x_train.reshape(60000, 28, 28, 1) / 255
x_test = x_test.reshape(10000, 28, 28, 1) / 255

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

建構 CNN 神經網路

```
# 以Sequential的方式建構自己的神經網路
model = Sequential()

# 16 represents the number of filters
# 每個 filter 都是 3*3
# padding = 'same' 的意思就是說最後出來的計分板也要是28*28的格式
# input shape 就是輸入的格式
# activation function 是 relu
model.add(Conv2D(16, (3,3), padding='same',
                 input_shape=(28,28,1),
                 activation='relu'))
```

建構 CNN 神經網路

使用 MaxPooling 以及增加 Filter 的方法來優化結果

```
model.add(MaxPooling2D(pool_size=(2,2)))

# filter 增加到32個
model.add(Conv2D(32, (3,3), padding='same',
                 activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

# filter 增加到64個
model.add(Conv2D(64, (3,3), padding='same',
                 activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))
```

建構 CNN 神經網路

```
model.add(Flatten())

# 以全連結層結尾
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='mse', optimizer=SGD(learning_rate=0.1),
              metrics=['accuracy'])
```

開始訓練 Model

```
model.compile(loss='mse', optimizer=SGD(learning_rate=0.1),
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=128, epochs=20)
```

查看模型準確率

```
y_predict = np.argmax(model.predict(x_test), axis=-1)

loss, acc = model.evaluate(x_test, y_test)
```

這邊一樣可以用 model.summary() 查看神經網路的樣子

測試模型

```
n = 2685
print('神經網路預測是:', y_predict[n])
plt.imshow(x_test[n].reshape(28,28), cmap='Greys')aluate(x_test, y_test)
```