mnist-cnn

一、 程式碼

1. 匯入資料集、轉換數據及類型

```
from keras.datasets import mnist
from keras.utils import to_categorical  # Corrected import
statement
import numpy as np

np.random.seed(15)

# Read MNIST data
(X_Train, y_Train), (X_Test, y_Test) = mnist.load_data()

# Translation of data
X_Train4D = X_Train.reshape(X_Train.shape[0], 28, 28,
1).astype('float32')
X_Test4D = X_Test.reshape(X_Test.shape[0], 28, 28,
1).astype('float32')
```

2. 將資料標準化

```
# Standardize feature data
X_Train4D_norm = X_Train4D / 255
X_Test4D_norm = X_Test4D /255
```

3. 將 MNIST 資料集的標籤轉換為 one-hot 編碼格式

```
from keras.utils import to_categorical

# Label Onehot-encoding
y_TrainOneHot = to_categorical(y_Train)
y_TestOneHot = to_categorical(y_Test)
```

4. 建構 CNN(卷積神經網路)，建立卷積層、池化層。

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPool2D
model = Sequential()
# Create CN layer 1
model.add(Conv2D(filters=16,
                 kernel_size=(5,5),
                 padding='same',
                 input_shape=(28,28,1),
                 activation='relu',
                 name='conv2d_1'))
# Create Max-Pool 1
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_1'))

# Create CN layer 2
model.add(Conv2D(filters=36,
                 kernel_size=(5,5),
                 padding='same',
                 input_shape=(28,28,1),
                 activation='relu',
                 name='conv2d_2'))
# Create Max-Pool 2
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_2'))

# Create CN layer 3
model.add(Conv2D(filters=48,
                 kernel_size=(5,5),
                 padding='same',
                 input_shape=(28,28,1),
                 activation='relu',
                 name='conv2d_3'))
# Create Max-Pool 3
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_3'))

# Add Dropout layer
model.add(Dropout(0.25, name='dropout_1'))
```

- 本來只有兩層卷積，多加一層，變成三層卷積，測試準確率是否有提升。

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPool2D
model = Sequential()
# Create CN layer 1
model.add(Conv2D(filters=16,
                                 kernel_size=(5,5),
                                 padding='same',
                                 input_shape=(28,28,1),
                                 activation='relu',
                                 name='conv2d_1'))
# Create Max-Pool 1
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_1'))

# Create CN layer 2
model.add(Conv2D(filters=36,
                                 kernel_size=(5,5),
                                 padding='same',
                                 input_shape=(28,28,1),
                                 activation='relu',
                                 name='conv2d_2'))
# Create Max-Pool 2
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_2'))

# Create CN layer 3
model.add(Conv2D(filters=48,
                                 kernel_size=(5,5),
                                 padding='same',
                                 input_shape=(28,28,1),
                                 activation='relu',
                                 name='conv2d_3'))
# Create Max-Pool 3
model.add(MaxPool2D(pool_size=(2,2), name='max_pooling2d_3'))

# Add Dropout layer
model.add(Dropout(0.25, name='dropout_1'))
```

5. 建立神經網路，並查看模型。

```
model.add(Flatten(name='flatten_1'))
model.add(Dense(128, activation='relu', name='dense_1'))
model.add(Dropout(0.5, name='dropout_2'))
model.add(Dense(10, activation='softmax', name='dense_2'))
model.summary()
print("")
```

6. 定義訓練並進行訓練

```
# 定義訓練方式
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

```
# 開始訓練
train_history = model.fit(x=X_Train4D_norm,
                          y=y_TrainOneHot, validation_split=0.2,
                          epochs=10, batch_size=500, verbose=2)
```

- 將每批訓練改為 500 個樣本，設置為 2 表示輸出訓練過程的詳情（進度條等）

```
# 定義訓練方式
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 開始訓練
train_history = model.fit(x=X_Train4D_norm,
                          y=y_TrainOneHot, validation_split=0.2,
                          epochs=10, batch_size=500, verbose=2)
```

7. 利用可視覺化圖形顯示研究過程及結果

```
import matplotlib.pyplot as plt
def plot_image(image):
    fig = plt.gcf()
    fig.set_size_inches(2,2)
    plt.imshow(image, cmap='binary')
    plt.show()

def plot_images_labels_predict(images, labels, prediction, idx,
num=15):
    fig = plt.gcf()
    fig.set_size_inches(12, 14)
    if num > 25: num = 25
    for i in range(0, num):
        ax=plt.subplot(5,5, 1+i)
        ax.imshow(images[idx], cmap='binary')
        title = "l=" + str(labels[idx])
        if len(prediction) > 0:
            title = "l={},p={}".format(str(labels[idx]),
str(prediction[idx]))
        else:
            title = "l={}".format(str(labels[idx]))
        ax.set_title(title, fontsize=10)
        ax.set_xticks([]); ax.set_yticks([])
        idx+=1
```

```
    plt.show()

def show_train_history(train_history, train, validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
#使用函數 show_train_history 顯示 accuracy 在 train 與 evaluation 的差異與 loss 在 train 與 evaluation 的差異如下:

show_train_history(train_history,'accuracy','val_accuracy')
show_train_history(train_history, 'loss', 'val_loss')
```

8. 評估模型準確度及進行預測

```
scores = model.evaluate(X_Test4D_norm, y_TestOneHot)
print()
print("\t[Info] Accuracy of testing data =
{:2.1f}%".format(scores[1]*100.0))
print("\t[Info] Making prediction of X_Test4D_norm")
prediction = model.predict(X_Test4D_norm)# Making prediction and
save result to prediction
prediction = np.argmax(prediction,axis=1)
print()
print("\t[Info] Show 15 prediction result (From 220):")
print("%s\n" % (prediction[220:235]))
```

- 顯示 220 開始的前 15 筆資料

結論：更改了一些變數及新增卷積層後，準確率並沒有顯著的變化。