# coding: utf-8

import numpy as np

import matplotlib.pyplot as plt

from keras.utils import np\_utils

from keras.datasets import mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

for i in range(0,5): ## 劃出數字

plt.subplot(1, 5, i+1)

img = x\_train[i]

plt.imshow(img, cmap="gray")

print(y\_train[:5]) ## show 出前5筆 y

print(x\_train.shape)

x\_train\_reshape = x\_train.reshape(60000, 784).astype('float32')

print(x\_train\_reshape.shape)

x\_train\_normalized = x\_train\_reshape / 255

print(x\_train\_normalized[:5])

print(y\_train[:5])

y\_train\_onehot = np\_utils.to\_categorical(y\_train) ## 轉成 10個0/1 碼

print(y\_train\_onehot[:5]) ## show 出前5筆 y

######################################################

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(units=256, input\_dim=784, kernel\_initializer="normal", activation="relu"))#也可以有不同模型

model.add(Dense(units=10, kernel\_initializer="normal", activation="softmax"))

print(model.summary())

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])

train\_history = model.fit(x=x\_train\_normalized, y=y\_train\_onehot, validation\_split=0.2, epochs=10, batch\_size=200, verbose=2)

get\_ipython().magic('matplotlib inline')

## IPython有一組預先定義好的所謂的魔法函數（Magic Functions），你可以通過命令列的語法形式來訪問它們。

import matplotlib.pyplot as plt

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.show()

show\_train\_history(train\_history, "acc", "val\_acc") ## 訓練正確率圖

show\_train\_history(train\_history, "loss", "val\_loss") ## 訓練誤差圖

print(len(y\_test))

x\_test\_reshape = x\_test.reshape(10000, 784).astype("float32")

x\_test\_normalized = x\_test\_reshape / 255

y\_test\_onehot = np\_utils.to\_categorical(y\_test)

scores = model.evaluate(x\_test\_normalized, y\_test\_onehot)

print("Accuracy: {}%".format(scores[1]))

import itertools

def plot\_confusion\_matrix(cm, classes, normalize=False, title="Confusion Matrix", cmap=plt.cm.Blues):

plt.figure()

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

thresh = cm.max() / 2

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, cm[i, j], horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

results = model.predict\_classes(x\_test\_reshape)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, results)

plot\_confusion\_matrix(cm, range(0, 9)) ## 彩色混屯矩陣

incorrect = np.where(y\_test != results)[0] ## 抓出錯誤的樣本資料

print(incorrect[:5])

for i in range(0, 9): ## 劃出數字

plt.subplot(3, 3, i+1)

idx = incorrect[i]

img = x\_test[idx]

plt.imshow(img, cmap="gray")

plt.title("{}/{}".format(y\_test[idx], results[idx]))

correct = np.where(y\_test == results)[0]

for i in range(0, 9):

plt.subplot(3, 3, i+1)

idx = correct[i]

img = x\_test[idx]

plt.imshow(img, cmap="gray")

plt.title("{}".format(y\_test[idx]))