# 顔認証したいな～
# (手書き数字の判別)

bv18076　森 大樹

2018/11/2

# 目次

# 1　研究背景

創るの授業 (顔認証で出席確認ができたら授業開始時の出席確認に時間をとることなく済むから顔認証をやろうという思って授業を進めていました) で興味を持ったち, 完成していないので, できれば自分で何か形として作ることがことができたらいいと思い, やろうと思いました.

# 2　研究方針

python を使い, 静止画像での顔認証をするためにこつこつと進めていく. 手書き数字の判別をやり, 関数がどのような動きをしているか学んでいき、数字の判別は一番似ているものに判断するのでそれを元に誰の顔に一番にているかをやろうと考えていて、これは顔認証とは違い本来の目的から離れているが段々近づくためにやっていきたい. そして, 顔認証に取り組んでいき学習データを少なくして顔認証ができたらいいと思っています.

# 3　実行結果

| 回数 | スコア |
|---|---|
| 1 | 0.90100 |
| 2 | 0.90785 |
| 3 | 0.88085 |
| 4 | 0.92128 |
| 5 | 0.91842 |
| 6 | 0.90914 |
| 7 | 0.91657 |
| 8 | 0.86585 |
| 9 | 0.92028 |
| 10 | 0.92628 |

10 回の平均は 0.90675 となった. このプログラムは 0.997 の精度が出たことがあったらしい. 自分はただ動かしていただけなので次のソースコード理解をやっているうちに何かわかることがあるかもしれないので次に進みたい.

# 4　ソースコードの理解

## 4.1　[1]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns

np.random.seed(2)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau

sns.set(style='white', context='notebook', palette='deep')
```

import：モジュールを使用できるようにします.as を使うことで使用できるようになったモジュールを別名で呼べるようにします.
from･･･ import･･･：これも import と同じようにモジュール使用できるようにするんですが import ではモジュールのクラスを使うとプログラムを長くかかなくてはいけなくなるので,from を使うことで短い文にできます．np.random.seed(2):固定された乱数を発生させてます.
sns.set:図などの表示の設定などを行い, 役に立ちます.

## 4.2　[2]

```
# Load the data
train = pd.read_csv("･･･ \input\train.csv")
test = pd.read_csv("･･･ \input\test.csv")
```

pd.read_csv:csv の形で与えられたデータの読み込みをします.

## 4.3 [3]

```
Y_train = train["label"]

# Drop 'label' column
X_train = train.drop(labels = ["label"],axis = 1)

# free some space
del train

g = sns.countplot(Y_train)

Y_train.value_counts()
```

Y_train = train["label"]:これは train に読み込んだデータの label の行のデータを Y_train に読み込ませました.
X_train = train.drop(labels = ["label"],axis = 1):train に読み込んだデータの label の行以外のデータを読み込ませました.
del train:train データを削除しました.
g = sns.countplot(Y_train):データの個数をグラフとして可視化してみれるようにします.
Y_train.value_counts():データの個数を可視化します.

## 4.4 [4]

```
# Check the data
X_train.isnull().any().describe()
```

## 4.5 [5]

```
test.isnull().any().describe()
```

··· .isnull().any().describe():行列ごとに一つでも欠損値があれば True と表示させ, なければ False と表示させます.

### 4.6 [6]

```
# Normalize the data
X_train = X_train / 255.0
test = test / 255.0
```

各ピクセルに 0 以上 255 以下の整数になっているので 255.0 で割って 0 から 1 の間に収めます.

### 4.7 [7]

```
# Reshape image in 3 dimensions (height = 28px, width = 28px , canal = 1)
X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
```

データを $28 \times 28$ の行列に変換します.

### 4.8 [8]

```
# Encode labels to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
Y_train = to_categorical(Y_train, num_classes = 10)
```

0 から 9 までの数字を一つのベクトルにするそうです.

### 4.9 [9]

```
# Set the random seed
random_seed = 2
```

random_seed に 2 という値を代入します.

### 4.10 [10]

```
# Split the train and the validation set for the fitting
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, ran-
dom_state=random_seed)
```

train_test_split:この関数を使ってデータを分割します.

## 4.11 [11]

```
# Some examples
g = plt.imshow(X_train[0][:,:,0])
```

画像を表示させます.

## 4.12 [12]

```
# Set the CNN model
# my CNN architechture is In -> [[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 ->
Flatten -> Dense -> Dropout -> Out

model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation ='relu',
input_shape = (28,28,1)))
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))
```

Sequential モデルの作成をしています．.add でレイヤーを積み重ねるそうです.

## 4.13 [13]

```
# Define the optimizer
optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
```

オプティマイザ (最適化アルゴリズム) はモデルをコンパイルする際に必要となるパラメータの一つだそうです.

### 4.14 [14]

```
# Compile the model
model.compile(optimizer=optimizer ,loss="categorical_crossentropy",metrics=["accuracy"])
```

モデルの学習をします. 3 つの引数でどのような画像処理を行うかを設定します.

### 4.15 [15]

```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc', patience=3, verbose=1,
factor=0.5, min_lr=0.00001)
```

評価値の改善が止まったときに学習率を減らします.

### 4.16 [16]

```
epochs = 1 # Turn epochs to 30 to get 0.9967 accuracy
batch_size = 86
```

## 4.17 [17]

```
# Without data augmentation i obtained an accuracy of 0.98114
#history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
# validation_data = (X_val, Y_val), verbose = 2)
# With data augmentation to prevent overfitting (accuracy 0.99286)

datagen = ImageDataGenerator(
featurewise_center=False, # set input mean to 0 over the dataset
samplewise_center=False, # set each sample mean to 0
featurewise_std_normalization=False, # divide inputs by std of the dataset
samplewise_std_normalization=False, # divide each input by its std
zca_whitening=False, # apply ZCA whitening
rotation_range=10, # randomly rotate images in the range (degrees, 0 to 180)
zoom_range = 0.1, # Randomly zoom image
width_shift_range=0.1, # randomly shift images horizontally (fraction of total width)
height_shift_range=0.1, # randomly shift images vertically (fraction of total height)
horizontal_flip=False, # randomly flip images
vertical_flip=False) # randomly flip images

datagen.fit(X_train)
```

データを回転やズームなどをさせてトレーニング用のデータを拡張しました.

## 4.18 [18]

```
# Fit the model
history = model.fit_generator(datagen.flow(X_train,Y_train, batch_size=batch_size),
epochs = epochs, validation_data = (X_val,Y_val),
verbose = 2, steps_per_epoch=X_train.shape[0] // batch_size
, callbacks=[learning_rate_reduction])
```

トレーニング用のデータを学習させました. 精度が得られます.

## 4.19 [19]

```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```

このプログラムは損失曲線と精度曲線をプロットするそうです．私が行ったときはプロットされませんでした．

## 4.20 [20]

```
# Look at confusion matrix
def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues):
"""
This function prints and plots the confusion matrix.
Normalization can be applied by setting `normalize=True`.
"""
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')
# Predict the values from the validation dataset
Y_pred = model.predict(X_val)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_val,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(10))
```

predicted label と True lubel の関係の表を表します.

## 4.21 [21]

```
# Display some error results
# Errors are difference between predicted labels and true labels
errors = (Y_pred_classes - Y_true != 0)
Y_pred_classes_errors = Y_pred_classes[errors]
Y_pred_errors = Y_pred[errors]
Y_true_errors = Y_true[errors]
X_val_errors = X_val[errors]
def display_errors(errors_index,img_errors,pred_errors, obs_errors):
""" This function shows 6 images with their predicted and real labels"""
n = 0
nrows = 2
ncols = 3
fig, ax = plt.subplots(nrows,ncols,sharex=True,sharey=True)
for row in range(nrows):
for col in range(ncols):
error = errors_index[n]
ax[row,col].imshow((img_errors[error]).reshape((28,28)))
ax[row,col].set_title("Predicted label :\nTrue label :".for-
mat(pred_errors[error],obs_errors[error]))
n += 1
# Probabilities of the wrong predicted numbers
Y_pred_errors_prob = np.max(Y_pred_errors,axis = 1)
# Predicted probabilities of the true values in the error set
true_prob_errors = np.diagonal(np.take(Y_pred_errors, Y_true_errors, axis=1))
# Difference between the probability of the predicted label and the true label
delta_pred_true_errors = Y_pred_errors_prob - true_prob_errors
# Sorted list of the delta prob errors
sorted_dela_errors = np.argsort(delta_pred_true_errors)
# Top 6 errors
most_important_errors = sorted_dela_errors[-6:]
# Show the top 6 errors
display_errors(most_important_errors, X_val_errors, Y_pred_classes_errors, Y_true_errors)
```

6 つの間違えて判断を行ったものを表示させます.

### 4.22 [22]

```
# predict results
results = model.predict(test)
# select the indix with the maximum probability
results = np.argmax(results,axis = 1)
results = pd.Series(results,name="Label")
```

予測結果を Numpy 配列でだします.

### 4.23 [23]

```
submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)
submission.to_csv("cnn_mnist_datagen.csv",index=False)
```

提出するファイルを保存します.

## 5 今後の課題

似ている顔の判断以降のことをやっていきます. まだプログラミング技術がないのでこれから理解しつつ自分で考え自分でしっかりとプログラミングしていけるようになるようにします.

## 参考文献


[1] Python でデータサイエンス,https://pythondatascience.plavox.info/numpy/,2018/11/1

[2] note.nkmk.me,https://note.nkmk.me/python-pandas-drop/,2018/11/1

[3] note.nkmk.me,https://note.nkmk.me/python-pandas-nan-judge-count/,2018/11/1

[4]【Python 入門】import・from でモジュールを読み込む方法 ,http://programming-study.com/technology/python-import/,2018/10/30

[5] 無から始める keras 第 5 回,https://qiita.com/Ishotihadus/items/b171272b954147976bfc,2018/11/1

[6] train_test_split 関数でデータ分割,https://docs.pyq.jp/python/machine_learning/tips/train_test_split.html,20

[7] keras モデル構築　コンパイル記述,http://kenbo.hatenablog.com/entry/2017/07/24/142935,2018/11/1

[8] コールバックの使い方,https://keras.io/ja/callbacks/,2018/11/1

[9] Model クラス API,https://keras.io/ja/models/model/,2018/11/1

[10] クジラ飛行机,『Python によるスクレイピング& 機械学習 開発テクニック BeautigulSoup、scikit-learn、 TensorFlow を使ってみよう』, ソシム株式会社,2016

[11] Introduction-to-cnn-keras-0.997(top-6),https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6 ,2018/11/1