digitbaseline

November 4, 2022

[1]:
```
# Baseline MLP for MNIST dataset
#import tensorflow as tf
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

[2]:
```
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

[ ]:
```
img = X_test[0]
y_test[0]
```

[ ]:
```
y_test.shape
```

[ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
imgplot = plt.imshow(img)
```

[3]:
```
# flatten 28*28 images to a 784 vector for each image
num_pixels = X_train.shape[1] * X_train.shape[2]
X_train = X_train.reshape((X_train.shape[0], num_pixels)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], num_pixels)).astype('float32')
```

[ ]:
```
X_train.shape
```

[ ]:
```
y_train.shape
```

[4]:
```
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255
```

[ ]:
```
y_train[1]
```

[5]:
```
# one hot encode outputs
y_test_org = y_test
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

[ ]:
```
y_test.shape
```

[ ]:
```
img = X_train[30].reshape(28,28)
```

[ ]:
```
y_train[30]
```

[ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
imgplot = plt.imshow(img)
```

[6]:
```
# define baseline model
def baseline_model():
        # create model
        model = Sequential()
        model.add(Dense(num_pixels, input_dim=num_pixels,activation='relu'))
        model.add(Dense(num_classes,  activation='softmax'))
        # Compile model
        model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
        return model
```

[7]:
```
# build the model
model = baseline_model()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10,␣
 ↪batch_size=200, verbose=1)
# Final evaluation of the model
```

```
Epoch 1/10
300/300 [==============================] - 31s 25ms/step - loss: 0.4834 -
accuracy: 0.8629 - val_loss: 0.1387 - val_accuracy: 0.9593
Epoch 2/10
300/300 [==============================] - 8s 26ms/step - loss: 0.1177 -
accuracy: 0.9667 - val_loss: 0.0964 - val_accuracy: 0.9725 0s - loss: 0.1185 -
accuracy: 0. - ETA: 0s - loss: 0.118
Epoch 3/10
300/300 [==============================] - 7s 24ms/step - loss: 0.0714 -
accuracy: 0.9790 - val_loss: 0.0830 - val_accuracy: 0.9762
Epoch 4/10
300/300 [==============================] - 7s 24ms/step - loss: 0.0488 -
accuracy: 0.9865 - val_loss: 0.0718 - val_accuracy: 0.9776
Epoch 5/10
```

```
300/300 [==============================] - 7s 24ms/step - loss: 0.0350 -
accuracy: 0.9913 - val_loss: 0.0686 - val_accuracy: 0.9787
Epoch 6/10
300/300 [==============================] - 7s 23ms/step - loss: 0.0254 -
accuracy: 0.9932 - val_loss: 0.0692 - val_accuracy: 0.9794
Epoch 7/10
300/300 [==============================] - 7s 24ms/step - loss: 0.0197 -
accuracy: 0.9949 - val_loss: 0.0588 - val_accuracy: 0.9824
Epoch 8/10
300/300 [==============================] - 7s 22ms/step - loss: 0.0133 -
accuracy: 0.9975 - val_loss: 0.0598 - val_accuracy: 0.9815
Epoch 9/10
300/300 [==============================] - 7s 22ms/step - loss: 0.0106 -
accuracy: 0.9978 - val_loss: 0.0641 - val_accuracy: 0.9816
Epoch 10/10
300/300 [==============================] - 6s 21ms/step - loss: 0.0086 -
accuracy: 0.9983 - val_loss: 0.0629 - val_accuracy: 0.9820
```

[7]:
```
<keras.callbacks.History at 0x1f023034430>
```

[8]:
```
scores, accuracy = model.evaluate(X_test, y_test, verbose=0)
#print("Baseline Error: %.2f%%" % (100-scores[1]*100))
print("Baseline Accuracy: %.2f%%" % (accuracy*100))
```

```
Baseline Accuracy: 98.20%
```

[9]:
```
y_predicted = model.predict_classes(X_test)
```

```
C:\Users\Think\anaconda3\envs\tensorflowEnv\lib\site-
packages\keras\engine\sequential.py:450: UserWarning: `model.predict_classes()`
is deprecated and will be removed after 2021-01-01. Please use instead:*
`np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class
classification   (e.g. if it uses a `softmax` last-layer activation).*
`(model.predict(x) > 0.5).astype("int32")`,   if your model does binary
classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

[10]:
```
confuse_matrix = confusion_matrix(y_test_org, y_predicted)
print(confuse_matrix)
```

```
[[ 972    1    1    1    1    0    1    1    2    0]
 [   0 1126    3    1    0    0    1    1    3    0]
 [   2    2 1015    0    1    0    2    3    7    0]
 [   0    0    4  994    0    3    0    1    5    3]
 [   1    0    3    0  966    0    3    1    0    8]
 [   2    0    0    9    3  867    3    1    6    1]
 [   5    3    2    1    4    3  939    0    1    0]
 [   1    5    9    1    1    0    0 1006    2    3]
 [   2    0    4    2    6    0    0    4  953    3]
```

```
[   2    4    0    2   13    3    0    3    0  982]]
```

[11]:
```
result = classification_report(y_test_org, y_predicted)
print(result)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.98      0.98      1010
           4       0.97      0.98      0.98       982
           5       0.99      0.97      0.98       892
           6       0.99      0.98      0.98       958
           7       0.99      0.98      0.98      1028
           8       0.97      0.98      0.98       974
           9       0.98      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

[ ]:
```
y_predicted[0]
```

[ ]:
```
img = X_test[0].reshape(28,28)
%matplotlib inline
import matplotlib.pyplot as plt
imgplot = plt.imshow(img)
```

[ ]:
```

```