MNIST Neural Network

June 26, 2019

In [17]:
```
import time
import mnist

from keras.utils import np_utils
from keras import Sequential
from keras import layers

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

import numpy as np
```

In [11]:
```
# getting data
train = mnist.train_images()
label_train = mnist.train_labels()
test = mnist.test_images()
label_test = mnist.test_labels()

n_classes = 10
```

In [12]:
```
# preprocessing data for the neural network
nsamples, nx, ny = train.shape
train = train.reshape((nsamples,nx*ny))

nsamples, nx, ny = test.shape
test = test.reshape((nsamples,nx*ny))

train = train.astype('float32')/255
test = test.astype('float32')/255

label_train = np.array(label_train)
label_test = np.array(label_test)

label_train = np_utils.to_categorical(label_train, n_classes)
label_test = np_utils.to_categorical(label_test, n_classes)
```

In [13]:
```
# training the neural network
start = time.time()
```

```
        model = Sequential()
        model.add(layers.Dense(512,  activation = 'relu', input_shape=(784,)))
        model.add(layers.Dense(512, activation = 'relu'))
        model.add(layers.Dense(10, activation = 'softmax'))

        model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
        model.fit(train, label_train, batch_size=128, epochs=10)

        end = time.time()
```

```
Epoch 1/10
60000/60000 [==============================] - 12s 205us/step - loss: 0.2162 - acc: 0.9369
Epoch 2/10
60000/60000 [==============================] - 13s 216us/step - loss: 0.0793 - acc: 0.9763
Epoch 3/10
60000/60000 [==============================] - 12s 203us/step - loss: 0.0503 - acc: 0.9841
Epoch 4/10
60000/60000 [==============================] - 14s 237us/step - loss: 0.0366 - acc: 0.9883
Epoch 5/10
60000/60000 [==============================] - 15s 254us/step - loss: 0.0271 - acc: 0.9909
Epoch 6/10
60000/60000 [==============================] - 15s 246us/step - loss: 0.0205 - acc: 0.9933
Epoch 7/10
60000/60000 [==============================] - 18s 297us/step - loss: 0.0209 - acc: 0.9929
Epoch 8/10
60000/60000 [==============================] - 16s 266us/step - loss: 0.0183 - acc: 0.9941
Epoch 9/10
60000/60000 [==============================] - 15s 248us/step - loss: 0.0149 - acc: 0.9951
Epoch 10/10
60000/60000 [==============================] - 16s 268us/step - loss: 0.0140 - acc: 0.9953
```

In [14]:
```
        # predicting
        pred = model.predict(test)
        pred = np.argmax(pred, axis=1)
        label_test = np.argmax(label_test, axis = 1)
```

In [15]:
```
        # function to calculate the metrics
        def metrics(cm, cls, size):
            cm = np.array(cm)
            tp = cm[cls][cls]
            fp = sum(cm[x, cls] for x in range(10))-cm[cls][cls]
            fn = sum(cm[cls, x] for x in range(10))-cm[cls][cls]
            tn = size - tp - fp - fn
            precision = tp/(tp+fp)
            recall = tp/(tp+fn)
            fmeasure = 2*(precision*recall)/(precision + recall)
```

```
            accuracy = (tp + tn)/size

            return precision, recall, fmeasure, accuracy
```

In [18]:
```
# Rows: Actual
# Cols: Predicted
cm = confusion_matrix(label_test, pred)
print("Confusion Matrix:\n ")
print(cm)
```

```
Confusion Matrix:

[[ 969    1    1    0    1    1    3    0    1    3]
 [   0 1129    1    0    0    0    2    1    2    0]
 [   2    0 1006    1    3    0    2   14    4    0]
 [   0    1    7  976    0   10    0    7    6    3]
 [   1    0    2    0  966    0    4    1    0    8]
 [   2    0    0    3    3  874    5    0    3    2]
 [   4    2    2    0    2    1  945    0    2    0]
 [   2    1    2    0    2    0    1 1016    1    3]
 [   5    1    3    2    3    4    0    7  946    3]
 [   1    4    0    0    5    4    1    7    1  986]]
```

In [19]:
```
# Class 0
precision0, recall0, f0, acc0 = metrics(cm, 0, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 0: ", round(precision0, 3), "  ", round(recall0, 3), \
      " ", round(f0, 3), "   ", round(acc0,3))
```

```
        Precision Recall F-measure Accuracy
Class 0:  0.983    0.989   0.986     0.997
```

In [20]:
```
# Class 1
precision1, recall1, f1, acc1 = metrics(cm, 1, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 1: ", round(precision1, 3), "  ", round(recall1, 3), \
      " ", round(f1, 3), "   ", round(acc1,3))
```

```
        Precision Recall F-measure Accuracy
Class 1:  0.991    0.995   0.993     0.998
```

In [21]:
```
# Class 2
precision2, recall2, f2, acc2 = metrics(cm, 2, len(test))
print("        Precision Recall F-measure Accuracy")
print("Class 2: ", round(precision2, 3), "  ", round(recall2, 3), \
      " ", round(f2, 3), "   ", round(acc2,3))
```

```
        Precision Recall F-measure Accuracy
Class 2:  0.982    0.975   0.979     0.996
```

```
In [22]: # Class 3
         precision3, recall3, f3, acc3 = metrics(cm, 3, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 3: ", round(precision3, 3), "  ", round(recall3, 3), \
               " ", round(f3, 3), "   ", round(acc3,3))
```

```
        Precision Recall F-measure Accuracy
Class 3:  0.994    0.966   0.98     0.996
```

```
In [23]: # Class 4
         precision4, recall4, f4, acc4 = metrics(cm, 4, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 4: ", round(precision4, 3), "  ", round(recall4, 3), \
               " ", round(f4, 3), "   ", round(acc4,3))
```

```
        Precision Recall F-measure Accuracy
Class 4:  0.981    0.984   0.982     0.996
```

```
In [24]: # Class 5
         precision5, recall5, f5, acc5 = metrics(cm, 5, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 5: ", round(precision5, 3), "  ", round(recall5, 3), \
               " ", round(f5, 3), "   ", round(acc5,3))
```

```
        Precision Recall F-measure Accuracy
Class 5:  0.978    0.98   0.979     0.996
```

```
In [25]: # Class 6
         precision6, recall6, f6, acc6 = metrics(cm, 6, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 6: ", round(precision6, 3), "  ", round(recall6, 3), \
               " ", round(f6, 3), "   ", round(acc6,3))
```

```
        Precision Recall F-measure Accuracy
Class 6:  0.981    0.986   0.984     0.997
```

```
In [26]: # Class 7
         precision7, recall7, f7, acc7 = metrics(cm, 7, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 7: ", round(precision0, 3), "  ", round(recall7, 3), \
               " ", round(f7, 3), "   ", round(acc7,3))
```

```
        Precision Recall F-measure Accuracy
Class 7:  0.983    0.988   0.976     0.995
```

```
In [27]: # Class 8
         precision8, recall8, f8, acc8 = metrics(cm, 8, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 8: ", round(precision8, 3), "  ", round(recall8, 3), \
               " ", round(f8, 3), "   ", round(acc8,3))
```

```
        Precision Recall F-measure Accuracy
Class 8:  0.979    0.971   0.975     0.995
```

```
In [29]: # Class 9
         precision9, recall9, f9, acc9 = metrics(cm, 9, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 9: ", round(precision9, 3), "  ", round(recall9, 3), \
               " ", round(f9, 3), "   ", round(acc9,3))
```

```
        Precision Recall F-measure Accuracy
Class 9:  0.978    0.977   0.978     0.996
```

```
In [30]: # number of instances classified correctly
         acc_score = accuracy_score(pred, label_test)
         print("Accuracy_score: ", round(acc_score, 5))
```

```
Accuracy_score:  0.9813
```

```
In [31]: # training time
         print("Training Time: %s secs" % round(end - start, 5))
```

```
Training Time: 146.72545 secs
```