201600282 엄기산

In [1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell. ast_node_interactivity = "all"
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as mp
%matplotlib inline
import scipy.special
```

In [4]:
```
class neuralNetwork:
    def __init__(self, inputnodes, hiddennodes, outputnodes, learningrate):
        self.inodes = inputnodes
        self.hnodes = hiddennodes
        self.onodes = outputnodes
        self.lr = learningrate
        self.wih = np.random.normal(0.0,pow(self.hnodes,-0.5),(self.hnodes,self.inode
        self.who = np.random.normal(0.0,pow(self.onodes,-0.5),(self.onodes,self.hnode
        self.activation_function = lambda x : scipy.special.expit(x)
        pass

    def train(self, inputs_lists, targets_lists):
        inputs = np.array(inputs_lists, ndmin=2).T
        targets = np.array(targets_lists, ndmin=2).T

        hidden_inputs = np.dot(self.wih,inputs)
        hidden_outputs = self.activation_function(hidden_inputs)

        final_inputs = np.dot(self.who, hidden_outputs)
        final_outputs = self.activation_function(final_inputs)

        output_errors = targets - final_outputs
        hidden_errors = np.dot(self.who.T, output_errors)

        self.who += self.lr * np.dot((output_errors * final_outputs * (1.0 - final_ou

        self.wih += self.lr * np.dot((hidden_errors * hidden_outputs * (1.0 - hidden_
        pass

    def query(self, inputs_lists):
        inputs = np.array(inputs_lists, ndmin=2).T
        hidden_inputs = np.dot(self.wih,inputs)
        hidden_outputs = self.activation_function(hidden_inputs)

        final_inputs = np.dot(self.who, hidden_outputs)
        final_outputs = self.activation_function(final_inputs)

        return final_outputs

input_nodes = 784
hidden_nodes = 100
output_nodes = 10
learning_rate = 0.1

n = neuralNetwork(input_nodes,hidden_nodes,output_nodes,learning_rate)

training_data_file = open("mnist_dataset/mnist_train.csv",'r')
training_data_list = training_data_file.readlines()
training_data_file.close()

epochs = 5
```

```
for e in range(epochs):
    for record in training_data_list:
        all_values =record.split(',')
        inputs = (np.asfarray(all_values[1:])/255.0*0.99)+0.01
        targets = np.zeros(output_nodes) + 0.01
        targets[int(all_values[0])] = 0.99
        n.train(inputs, targets)
        pass
pass
```

In [5]:
```
test_data_file = open("mnist_dataset/mnist_test.csv",'r')
test_data_list = test_data_file.readlines()
test_data_file.close()
```

In [6]:
```
scorecard = []

for record in test_data_list:
    all_values = record.split(',')
    correct_label = int(all_values[0])
    inputs = (np.asfarray(all_values[1:])/255.0*0.99)+0.01
    outpus = n.query(inputs)
    label = np.argmax(outpus)

    if(label == correct_label):
        scorecard.append(1)
    else:
        scorecard.append(0)
        pass
    pass
```

In [7]:
```
scorecard_array = np.asarray(scorecard)
print("performance = ", scorecard_array.sum() / scorecard_array.size)
```

```
performance =  0.9684
```

hidden 계층 1개 추가

In [8]:
```
class neuralNetwork2:
    def __init__(self, inputnodes, hiddennodes, hiddennodes2,outputnodes, learningrat
        self.inodes = inputnodes
        self.h1nodes = hiddennodes
        self.h2nodes = hiddennodes2
        self.onodes = outputnodes

        self.lr = learningrate
        self.wih1 = np.random.normal(0.0,pow(self.h1nodes,-0.5),(self.h1nodes,self.ir
        self.wh1h2 = np.random.normal(0.0,pow(self.h2nodes,-0.5),(self.h2nodes,self.h
        self.wh2o = np.random.normal(0.0,pow(self.onodes,-0.5),(self.onodes,self.h2nc

        self.activation_function = lambda x : scipy.special.expit(x)
        pass

    def train(self, inputs_lists, targets_lists):
        inputs = np.array(inputs_lists, ndmin=2).T
        targets = np.array(targets_lists, ndmin=2).T

        hidden1_inputs = np.dot(self.wih1,inputs)
        hidden1_outputs = self.activation_function(hidden1_inputs)

        hidden2_inputs = np.dot(self.wh1h2,hidden1_outputs)
```

```
        hidden2_outputs = self.activation_function(hidden2_inputs)

        final_inputs = np.dot(self.wh2o, hidden2_outputs)
        final_outputs = self.activation_function(final_inputs)

        output_errors = targets - final_outputs
        hidden2_errors = np.dot(self.wh2o.T, output_errors)
        hidden1_errors = np.dot(self.wh1h2.T, hidden2_errors)

        self.wh2o += self.lr * np.dot((output_errors * final_outputs * (1.0 - final_

        self.wh1h2 += self.lr * np.dot((hidden2_errors * hidden2_outputs * (1.0 - hi

        self.wih1 += self.lr * np.dot((hidden1_errors * hidden1_outputs * (1.0 - hid
        pass

    def query(self, inputs_lists):
        inputs = np.array(inputs_lists, ndmin=2).T

        hidden1_inputs = np.dot(self.wih1,inputs)
        hidden1_outputs = self.activation_function(hidden1_inputs)

        hidden2_inputs = np.dot(self.wh1h2,hidden1_outputs)
        hidden2_outputs = self.activation_function(hidden2_inputs)

        final_inputs = np.dot(self.wh2o, hidden2_outputs)
        final_outputs = self.activation_function(final_inputs)

        return final_outputs

input_nodes = 784
hidden1_nodes = 100
hidden2_nodes = 100
output_nodes = 10
learning_rate = 0.1

n2 = neuralNetwork(input_nodes,hidden_nodes,output_nodes,learning_rate)

epochs = 5

for e in range(epochs):
    for record in training_data_list:
        all_values =record.split(',')
        inputs = (np.asfarray(all_values[1:])/255.0*0.99)+0.01
        targets = np.zeros(output_nodes) + 0.01
        targets[int(all_values[0])] = 0.99
        n2.train(inputs, targets)
        pass
pass
```

In [9]:

```
scorecard2 = []

for record in test_data_list:
    all_values = record.split(',')
    correct_label = int(all_values[0])
    inputs = (np.asfarray(all_values[1:])/255.0*0.99)+0.01
    outpus = n2.query(inputs)
    label = np.argmax(outpus)

    if(label == correct_label):
        scorecard2.append(1)
    else:
        scorecard2.append(0)
```

```
        pass
    pass
```

In [10]:
```
scorecard_array2 = np.asarray(scorecard2)
print("performance2 = ", scorecard_array2.sum() / scorecard_array2.size)
```

```
performance2 =  0.9655
```

201600282 엄기산