**Multilayer Perceptrons**

**Machine Learning**

**Aiden Koknat**

Data Representation:

The points were stored in a multidimensional array, first reading in the initial value of each line of data as the target value, and the rest of it as the points to be referenced. This ended up resulting in a (12665, 785) array for all the training data, and (2115, 785) for the test data.

Functions used:

- read_in_data(file):
  - Takes a file name and separates the lines of data points into returned array values suitable for the program. Used for the training and testing data
- prepare_data(row):
  - Takes a row of data and separates it into a returned class value and the rest of the data points.
- sigmoid(x):
  - Takes a value and returns the sigmoid activator function with x as the variable.
- neural_network(inputs, input_weights, hidden_weights, hidden_bias, output_bias):
  - Sets up the neural network, returns the interior (hidden) and exterior output(s), recorded as input_g and output.
- backprop(inputs, class_value, output, input_g, hidden_weights, output_bias, hidden_bias):
  - Backpropagates the data from the neural network by creating output_delta and hidden_delta values. It returns the updated weights and biases.

After defining these functions, it begins by first reading in the training and test data using read_in_data. Next, it sets up the output bias and alpha value. It then sets up the size of the network.I chose to have 28 nodes in the hidden layer, with an output size of 1:

```
74  # setting up size of network
75  input_size = len(train_data[0]) - 1  # was len(data_example[0])   input_size: 784
76  hidden_size = 28   hidden_size: 28
77  output_size = 1   output_size: 1
78  hidden_bias = np.empty([input_size, 1])   hidden_bias: [[0.99978087 0.99999939 1.0
```

Afterwards, it initializes the weights and values of the hidden nodes:

```
81      # Initializing random weights
82      input_weights = np.random.randn(input_size, hidden_size)   input_weig
83      hidden_weights = np.array(np.random.randn(hidden_size, output_size))
84      hidden_values = np.array([range(hidden_size)])   hidden_values: [[ 0
```

After this, training begins. I set a max number of epochs at 100, but once the accuracy of the data reaches 99.5%, it will stop training and save the weights. Training entails iterating through each line of values (each image represented in color values)

```
86  # Training
87  max_epoch_num = 100   max_epoch_num: 100
88
89  num_total = len(train_data)   num_total: 12665
90  for i in range(max_epoch_num):   i: 0
91      num_correct = 0   num_correct: 11111
92      for index in range(len(train_data)):   index: 12664
93          class_value_example, data_example = prepare_data(train_data[index])   class_value_example: [1]    data_ex
94          class_value_score = int(class_value_example)   class_value_score: 1
95
96          # feedforward pass
97          output, input_g = neural_network(data_example, input_weights, hidden_weights, hidden_bias, output_bias)
98
99          if class_value_score == 0:
100             if output < .5:
101                 num_correct = num_correct + 1
102
103         if class_value_score == 1:
104             if output >= .5:
105                 num_correct = num_correct + 1
106
107         # weight updates/backprop
108         updated_input_weights, updated_hidden_weights, updated_output_bias, updated_hidden_bias = backprop(data_
109
110         # weight/bias updates
111         input_weights = updated_input_weights
112         hidden_weights = updated_hidden_weights
113         output_bias = updated_output_bias
114         hidden_bias = updated_hidden_bias
115
116     accuracy_percentage = (num_correct / num_total) * 100
117     print("Accuracy of training set " + str(i) + ": " + str(accuracy_percentage) + "%")
118     if accuracy_percentage > 99.5:
119         break
```

The training usually reaches the accuracy benchmark around 30 epochs:

```
Accuracy of training set 0: 89.0248716936439%
Accuracy of training set 1: 95.70469798657719%
Accuracy of training set 2: 96.85748124753258%
Accuracy of training set 3: 97.22858270825108%
Accuracy of training set 4: 97.88393209632846%
Accuracy of training set 5: 98.38136596920647%
Accuracy of training set 6: 98.43663639952625%
Accuracy of training set 7: 98.53138570864589%
```

Finally, the test set will run through, using the weights from the training set that got an accuracy above 99.5%. if it fails to the reach the 99.5% accuracy benchmark after 100 epochs, it will continue with the test set with the weights from the 100th epoch.

```
Accuracy of training set 94: 99.17094354520331%
Accuracy of training set 95: 99.18673509672325%
Accuracy of training set 96: 99.19463087248323%
Accuracy of training set 97: 99.2025266482432%
Accuracy of training set 98: 99.19463087248323%
Accuracy of training set 99: 99.17883932096329%
Accuracy of test: 99.66903073286052%

Process finished with exit code 0
```