Handwritten Digit Recognition Report without TF

Himanshu Gangwal

July 3, 2023

1 Introduction

This report presents the implementation and analysis of a handwritten digit recognition system using a neural network. The code provided utilizes the TensorFlow and NumPy libraries for building and training the neural network model.

2 Data Preprocessing

The MNIST dataset is used for training and testing the model. It consists of images of handwritten digits along with their corresponding labels. The dataset is divided into training and testing sets. The pixel values of the images are normalized to the range [0, 1] by dividing each value by 256.

3 Neural Network Architecture

The neural network used in this implementation consists of one hidden layer with 32 neurons and an output layer with 10 neurons, representing the possibilities for the digit classification. The weights and biases are initialized randomly.

4 Mathematics of Forward and Backpropagation

We will now discuss the mathematical calculations involved in the forward and backpropagation steps of our neural network.

4.1 Forward Propagation

During forward propagation, the neural network processes the input data and computes the output. Let's consider a single training pattern and denote it as *pattern*. We calculate the output as follows:

$$y_1 = b_1 + w_1 \cdot pattern$$

Here, y_1 represents the input to the hidden layer, b_1 is the bias, and w_1 is the weight matrix connecting the input to the hidden layer.

Next, we apply the activation function (sigmoid) to y_1 to obtain the output of the hidden layer, denoted as $y_{1_{box}}$:

$$y_{1_{box}} = \frac{1}{1 + e^{-y_1}}$$

Now, we calculate the output of the network's final layer as follows:

$$y_2 = b_2 + w_2 \cdot y_{1_{box}}$$

Here, y_2 represents the input to the output layer, b_2 is the bias, and w_2 is the weight matrix connecting the hidden layer to the output layer.

Finally, we apply the activation function (sigmoid) to y_2 to obtain the output of the network, denoted as *out*:

$$out = \frac{1}{1 + e^{-y_2}}$$

4.2 Backpropagation

During backpropagation, the network adjusts the weights and biases based on the calculated error. Let's denote the desired output (label) for the given pattern as *number*.

We calculate the error as follows:

$$error = \frac{1}{len(out)} \sum_{i=1}^{len(out)} (out_i - number_i)^2$$

To update the weights and biases, we calculate the delta values for the output layer as follows:

```
delta_y2 = out - number
w2 += -e * np.matmul(np.transpose(delta_y2[np.newaxis]), y1_box[np.newaxis])
b2 += -e * delta_y2
```

Next, we calculate the delta values for the hidden layer as follows:

$$delta\_y_1 = np.transpose(w_2) \cdot delta\_y_2 \cdot (y_{1_{box}} \cdot (1 - y_{1_{box}}))$$

Finally, we update w_1 and b_1 using the delta values:

```
delta_y1 = np.transpose(w2) @ delta_y2 * (y1_box * (1 - y1_box))
w1 += -e * np.matmul(np.transpose(delta_y1[np.newaxis]), pattern[np.newaxis])
b1 += -e * delta_y1
```

Here, e represents the learning rate, and `np.newaxis` is used to adjust the shape of the arrays for matrix multiplication.

This process is repeated for each training pattern for a specified number of epochs. The weights and biases are gradually adjusted to minimize the error and improve the network's accuracy.

The forward propagation and backpropagation steps are crucial in training the neural network and improving its performance.

5 Training and Evaluation

The training process involves iterating over the training set for a specified number of epochs. For each training pattern, the forward propagation and backpropagation steps are performed to update the weights and biases. The accuracy of the model is calculated by comparing the predicted output with the actual label during the forward propagation step.

After training, the model is evaluated on the testing set. The forward propagation step is performed on each testing pattern, and the accuracy is calculated based on the number of correctly classified samples.

6 Conclusion

In this code, we implemented a neural network and discussed the mathematics involved in the forward and backpropagation steps. Here are the key points to take away:

1. Forward propagation: During forward propagation, the input pattern is processed through the network to compute the output. This involves calculating the inputs to the hidden and output layers, applying the sigmoid activation function, and obtaining the final network output.

2. Backpropagation: During backpropagation, the network adjusts the weights and biases based on the calculated error. The error is computed by comparing the network output with the desired output for a given pattern. The weights and biases are updated using the delta values, which are obtained by propagating the error backwards through the layers.

3. Training process: The forward and backpropagation steps are repeated for each training pattern for a specified number of epochs. The weights and biases are gradually adjusted to minimize the error and improve the network's accuracy.

4. Learning rate: The learning rate, denoted as e, controls the step size of weight and bias updates. It is an important hyperparameter that affects the convergence and training speed of the neural network.

By understanding the mathematics behind forward and backpropagation, we gain insights into how the network learns from the data and how the weights and biases are adjusted to optimize performance. This is an introductory example for getting the essence of forward and backpropagation.