Creating a Neural Network for MNIST Classification Using NumPy

Badger Code

July 29, 2024

Abstract

This paper details the implementation of a neural network from scratch using only NumPy to classify handwritten digits from the MNIST dataset. We cover the forward pass using ReLU and softmax functions, the computation of the categorical cross-entropy loss, and backpropagation for gradient descent optimization.

1 Introduction

Neural networks have become a fundamental tool in the field of machine learning, particularly for tasks involving image recognition. In this paper, we demonstrate how to implement a simple neural network using only NumPy to classify digits from the MNIST dataset. This implementation includes the forward pass, loss computation, and backpropagation.

2 Data Preparation

We use the MNIST dataset, which contains 70,000 images of handwritten digits (0-9). Each image is 28x28 pixels. The dataset is normalized and split into training and test sets. Labels are one-hot encoded.

3 Neural Network Architecture

Our neural network consists of an input layer with 784 neurons (28x28 pixels), a hidden layer with 128 neurons, and an output layer with 10 neurons (one for each digit).

3.1 Initialization

Weights and biases are initialized as follows:

$$\mathbf{W}_1 \sim \mathcal{N}(0, 0.01), \qquad \mathbf{b}_1 = \mathbf{0} \tag{1}$$

$$\mathbf{W}_2 \sim \mathcal{N}(0, 0.01), \qquad \mathbf{b}_2 = \mathbf{0} \tag{2}$$

4 Forward Pass

The forward pass consists of computing the activations for each layer using the ReLU activation function for the hidden layer and the softmax function for the output layer.

4.1 ReLU Activation Function

The ReLU activation function is defined as:

$$\text{ReLU}(z) = \max(0, z) \tag{3}$$

4.2 Softmax Function

The softmax function is used to compute the probabilities of each class:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4}$$

4.3 Forward Propagation Equations

The forward propagation through the network is given by:

$$\mathbf{Z}_1 = \mathbf{X}\mathbf{W}_1 + \mathbf{b}_1 \tag{5}$$

$$\mathbf{A}_1 = \text{ReLU}(\mathbf{Z}_1) \tag{6}$$

$$\mathbf{Z}_2 = \mathbf{A}_1\mathbf{W}_2 + \mathbf{b}_2 \tag{7}$$

$$\mathbf{A}_2 = \text{softmax}(\mathbf{Z}_2) \tag{8}$$

5 Loss Function

The categorical cross-entropy loss function measures the performance of the classification model:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{K} y_{ij} \log(\hat{y}_{ij}) \tag{9}$$

where $\mathbf{y}$ is the true label, $\hat{\mathbf{y}}$ is the predicted probability, m is the number of samples, and K is the number of classes.

6 Backpropagation

Backpropagation is used to update the weights and biases by computing the gradients of the loss function with respect to the parameters.

6.1 Gradient Computation

The gradients are computed as follows:

$$\mathbf{dZ}_2 = \mathbf{A}_2 - \mathbf{Y} \tag{10}$$

$$\mathbf{dW}_2 = \frac{1}{m}\mathbf{A}_1^\top \mathbf{dZ}_2 \tag{11}$$

$$\mathbf{db}_2 = \frac{1}{m}\sum_{i=1}^{m} \mathbf{dZ}_2 \tag{12}$$

$$\mathbf{dA}_1 = \mathbf{dZ}_2 \mathbf{W}_2^\top \tag{13}$$

$$\mathbf{dZ}_1 = \mathbf{dA}_1 \odot \mathrm{ReLU}'(\mathbf{Z}_1) \tag{14}$$

$$\mathbf{dW}_1 = \frac{1}{m}\mathbf{X}^\top \mathbf{dZ}_1 \tag{15}$$

$$\mathbf{db}_1 = \frac{1}{m}\sum_{i=1}^{m} \mathbf{dZ}_1 \tag{16}$$

6.2 Parameter Update

The parameters are updated using gradient descent:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \alpha \mathbf{dW}_1 \tag{17}$$

$$\mathbf{b}_1 \leftarrow \mathbf{b}_1 - \alpha \mathbf{db}_1 \tag{18}$$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \alpha \mathbf{dW}_2 \tag{19}$$

$$\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \alpha \mathbf{db}_2 \tag{20}$$

where α is the learning rate.

7 Training the Neural Network

We train the neural network on the training data using a specified number of epochs and learning rate. The loss is printed every 100 epochs to monitor the training process.

8 Evaluation

After training, we evaluate the model on the test set and compute the accuracy:

$$\text{accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100 \tag{21}$$

9 Visualization

We visualize some predictions to see how well the model performs on individual samples.

10 Conclusion

This paper demonstrated the implementation of a neural network using only NumPy to classify MNIST digits. The network was trained using gradient descent and evaluated on the test set to measure its accuracy.