Neural Network Without Pytorch & Iris Dataset

Elijah Booth, Luke Morrison, Adam Worthington

MAP4112: Mathematics For Machine Learning

Dr. Gerrit Welper

December 7, 2023

Introduction:

Although the implemented project is on the Iris dataset, the challenge here was to construct a neural network without the assistance of Pytorch or software that could shortcut any steps. As such we needed to manually implement a network class, a layer class, and a node class. Each played a critical part in the neural network; the network includes the backpropagation and forward pass that solves the minimizing issue, the layer class worked the specifics of the nodes in each layer allowing for forward and backward function, and the node class maintains the fully dense structure.

Discussion:

The network class works to solve the minimizing issue using least squares method and acts as the construct for the entire network, defining the amount of layers in which defines the nodes. It was initially given too much responsibility working with the partial derivative in our backpropagation algorithm; this was later delegated to the layer class. The network class now loops through the layers, calling the layer class's back propagation function instead, thus ensuring the shapes of the layers' input vectors and weight matrices are correct. Regarding forward pass, it first ensures appropriate data is passed into the function, and similar to the backpropagation function it loops through the layers calling the layer class's forward function. The loss function (MSE) is then where the output given by the forward_pass function is passed into the backpropagation function, minimizing the least squares error and finishing the computation. The last function called is update_weights, performing the gradient descent and updating each layer' weights and bias.

The layer class acts on each layer of the neural network, identifying it’s status and performing computations. It first defines a layer by it’s status as the input layer, a hidden layer, or an output layer, and a list containing all the nodes inside the layer. It’s goal is then to simplify the computations using embedded weights and bias attributes, and their current corresponding gradients using the forward and backward functions. The forward function takes the layers corresponding weights matrix, input vector, and bias vector, computes the z value (before activation) and then determines if an activation should be applied; in this case it is the ReLU activation. In turn, the backward function will construct the weights gradient and bias via the partial derivative at each layer.

The final class, the node class, is rather simple, containing an attribute called connections which contains a list of node objects. This list represents every node the node is connected to and allows for manipulation of the density of the structure; in this case it is fully connected. It was also initially designed to maintain the value and weights for each node, this responsibility was passed onto the layer class.

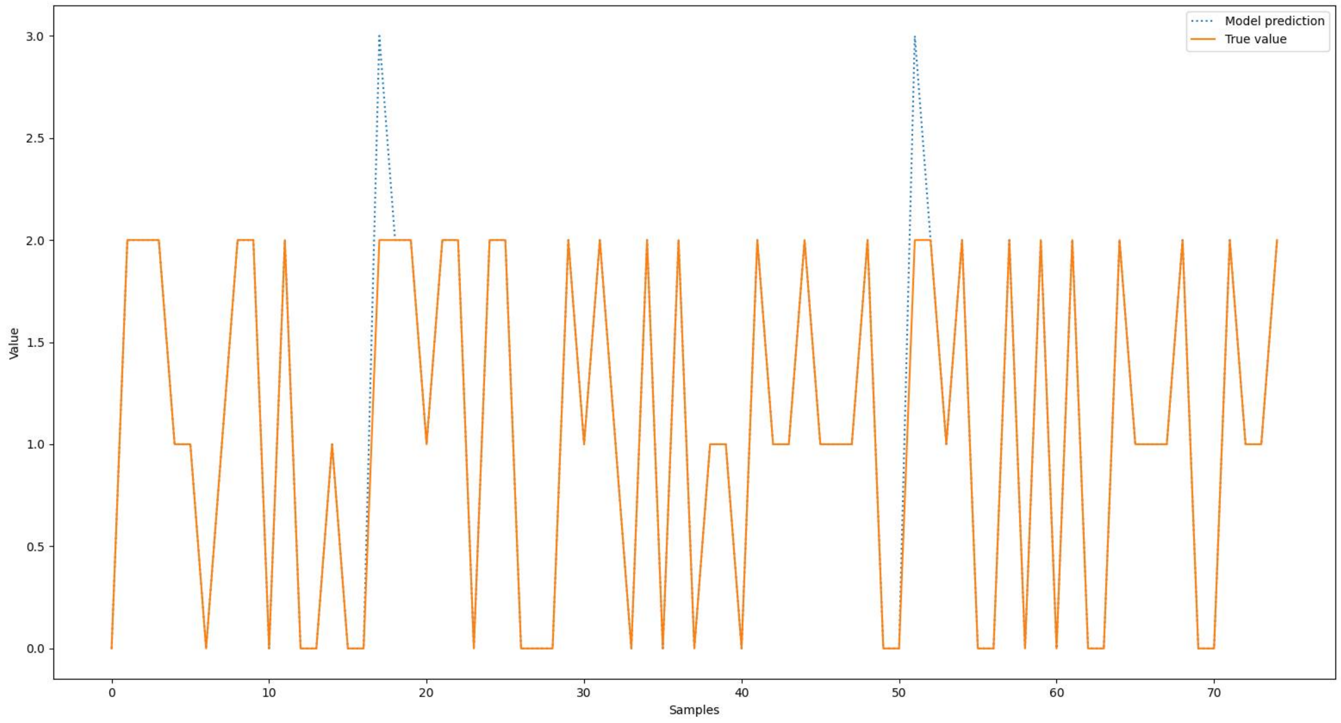
Implementation:

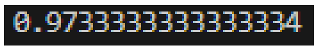
The neural network was implemented for the Iris dataset, containing data on 4 features of plants resulting in 3 classes of 50 instances per class. To prepare the data it first normalizes its range and splits it into training and test sets.

The Code then defines the number of epochs to train the network, creating an instance of the network class, fully connecting the network, and randomizing the biases; ensuring the weights and biases are the correct shape. Next it calls the train_network function which inputs the network, the number of epochs, the datasets, and loops through all epochs thus training the network. It then performs the test on the constructed neural network, receives the error, and plots the true values alongside the predictive values.

Results & Conclusion:

Here is a resultant graph from executing the implemented code and the corresponding accuracy:



Testing Accuracy: 

```
0.9733333333333334
```

In the test set of 75 samples 73 were predicted correctly, and so an effective neural network that can classify samples from the iris dataset was constructed. It should be mentioned that this network is effective for classification problems where the number of features is relatively low; this is because we run into a zeroing issue where occasionally every forward pass produces zero. Though identification of the issue has not been completed, one can suspect it has something to do with the ReLU activation function as it makes all values which are negative zero. That is, by creating a more complex model with more inputs, it has more chances to zero out. Nevertheless, this project has produced an accurate model for a unique classification dataset and the team behind the project intends to use this network as a building block for more difficult projects in the future.