cs577 Assignment 2

Jorge Gonzalez Lopez
A20474413
Department of Computer Science
Illinois Institute of Technology
February 9, 2021

Abstract

A multi-class classification neural network formed by 2 hidden layers has been created from scratch. This neural network has been tried on two different datasets in order to assess its behaviour and performance with real datasets.

Problem statement:

A simple neural network has to be created from scratch to be able to perform a multi-class classification problem. The networks specifications are the following:

- It has to include a loss function and an evaluation function.
- A different class for each type of node (simple mathematical operation by which a computational graph is formed) with forward and backwards computing methods.
- Forward and Backward traversal of the computation graph where data batches are pushed forward, and gradient loss values are pushed backward.
- Stochastic gradient descent (SGD) with a learning rate and decay parameters for updating the weights and the biases of the model

Proposed Solution:

As one of the requirements demanded, a different class for each type of node has been created:

1. **Linear Node:** These types of nodes just perform the basic operation for the forward pass which consists of a matrix multiplication of the weights and the input plus the bias, Figure 1.

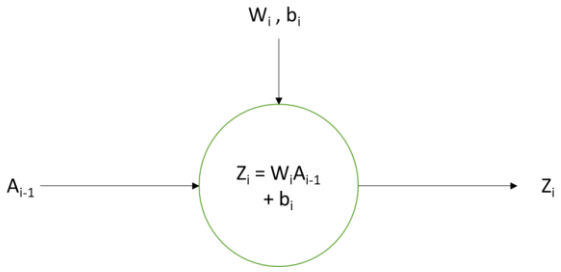

Figure 1: Basic functioning of a Linear Node.

Therefore, for the backward propagation, they compute the derivatives of Z with respect of the weights, the bias and the inputs and return their product with the previous derivatives.

2. **Sigmoid Node:** There are two Sigmoid nodes in the model for the activations of the hidden layers. The sigmoid function in the forward propagation is as follows:

$$\sigma(Z) = \frac{1}{1 + e^{-Z}}$$

And its derivative for the backward propagations corresponds to the following equation:

$$\frac{d}{dZ}\sigma(Z) = \sigma(Z)(1 - \sigma(Z))$$

3. **Softmax Node:** There is only one of these nodes in the network. It is the activation function of the output layer. The softmax function in the forward propagation returns a vector with a length equal to the total number of classes as follows:

$$softmax(Z) = \frac{e^{Z_i}}{\sum_{j=0}^{K} e^{Z_j}}$$

At the same time, as it has been seen in class, the backward propagation of the softmax function will be done jointly with the loss function for simplicity. Hence, in this node, the output for the forward propagation will be the same as the input.

4. **Loss Node:** There is only one node at the end of the network. In the forward propagation, this node provides the performance of the model with the current weights and biases. The Loss function used is the categorical cross-entropy and the evaluation function computed is the model's accuracy.
At the same time, for the backward propagation, the computed derivative is with respect the input of the softmax function as follows:

$$\frac{d\,E}{dZ} = \hat{Y} - Y$$

At the same time, another class has been created for the overall Model. To create a model, the dimensions of the network have to be passed as an argument:

(Input_features, neurons_layer1, neurons_layer2, neuron_layer3)

Then, a model with those dimensions is created and its weights initialized randomly with low values.
At the same time, different functions have been created to train and test the model:

- **Forward**: this function computes the forward propagation of all the layers with their respective parameters and returns the value of the Loss and accuracy.

- **Backward**: This function computes the backward propagation of all the layers and stores all the derivatives of the loss with respect the parameters.

- **Params_update:** This function updates the values of the parameters by subtracting them the derivatives of the backward propagation multiplied by the learning rate.

- **Fit**: It is the function that trains the model. To do so, it needs 6 arguments: the input training data (X_t) with its respective output labels (Y_t), the input validation data (X_v) with its respective output labels (Y_v), the learning rate and the total number of iterations. Then, the function performs the forward pass, backward pass and parameters update a total number of times as the number of iterations defined and stores the loss, validation loss, accuracy, and validation accuracy for all of them.

- **Print_results:** this function prints the loss and accuracy evolution for the test and validation datasets throughout the training.

- **Test**: This function computes the Loss and the accuracy of a given inputs with the current values of the weights of the model.

Results and Discussion:

The model has been tested on two different datasets: the Iris dataset and the Wine datasets. Both of them found in the following address: https://archive.ics.uci.edu/ml/index.php.

1. IRIS Dataset:

This dataset consists of 4 features that measure (in cm) the sepal length, the sepal width, the petal length, and the petal width to try to discern between three different varieties of Iris: Iris Setosa, Iris Versicolour and Iris Virginica. The data contains 150 samples (50 in each of three classes).

The preprocessing carried out is as follows:

a) The data is read as a Pandas dataframe.
b) The dataframe is split into the inputs (X) and the target (y), knowing that the target corresponds to the last column.
c) Then, the X and y arrays are shuffle and divided into three different datasets: training, validation, and test datasets with a split ratio of 7:1.5:1.5, respectively.

Finally, as the target is categorical, a function called get_dummies is used to One-Hot encode the target data into vector of '0's and '1's.

Finally, a Neural Network is designed with the following design:

- First hidden layer with 8 neurons.
- Second hidden layer with 16 neurons.
- Output Layer with 3 units (number of classes).

At the same time, a learning rate of 1e-2, and 1000 epochs have been used.

As it can be seen in Figure 2, the model's validation error is smaller than the training one. Therefore, an increase of the complexity of the network is advisable as the model suffers from a little of high bias.

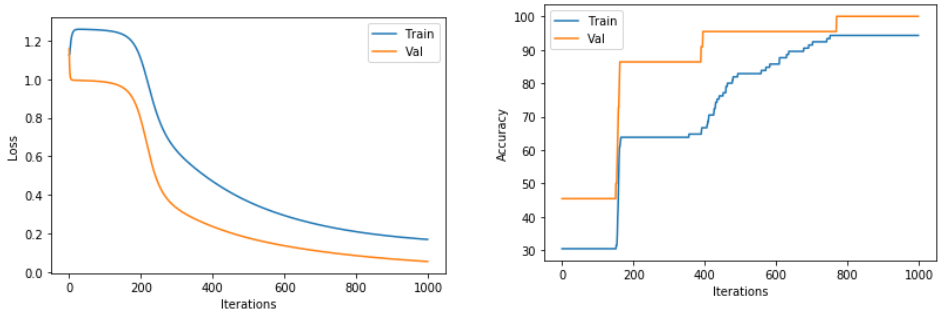

Figure 2: Model results with 8 and 16 neurons in the hidden layers.

However, even though the complexity of the network is increased to 32 and 64 neurons in the hidden layers, the problem persists, Figure 3. Therefore, the real problem could be the scarce number of samples in each dataset (training, validation and testing) as there are only a total of 150 samples.

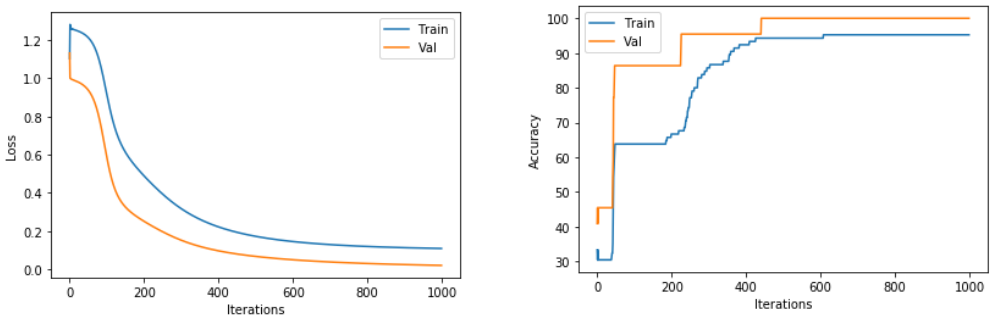

Figure 3: Model results with 32 and 64 neurons in the hidden layers.

With this model, the results over the test dataset are the following:

loss: 0.1329 - accuracy: 95.65217 %

2. Wine Quality Dataset:

This dataset consists of 11 features that measure some important features (acidity, density, pH, alcohol…) of 4898 different wines to try to get the wine's quality between 0 and 10. Therefore, this problem can be solved either with a regression or a classification model. Hence, in this case, a classification approach has been used with just 3 out of all possible quality values of the wines: 4, 5 and 6.

The same preprocessing as with the IRIS dataset has been followed, ending up with three datasets: the training, the validation, and the testing datasets.

Finally, a Neural Network is designed with the following design:

- First hidden layer with 8 neurons.
- Second hidden layer with 16 neurons.
- Output Layer with 3 units (number of classes).

At the same time, a learning rate of 1e-2, and 500 epochs have been used.

As it can be seen in Figure 4, the model's error and accuracy gets stable too fast and then, they are constantly oscillating. Therefore, a reduction of the learning rate and of the complexity of the network will be applied so that the model can get to the optimum and do not diverged.

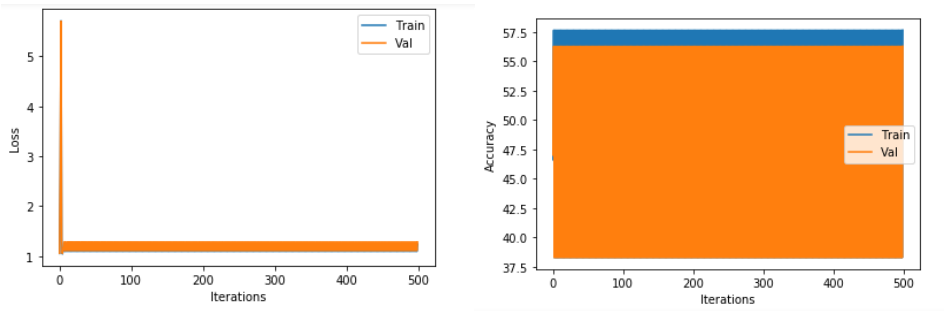

Figure 4: Model results with 8 and 16 neurons in the hidden layers.

Finally, the complexity of the network is reduced to 2 and 4 neurons in the hidden layers with a learning rate value of 0.0001, Figure 5. In this case, the model needs 100 iterations to reach the lowest error (as the learning rate is much smaller and the model is simpler) and does not fluctuate any more.

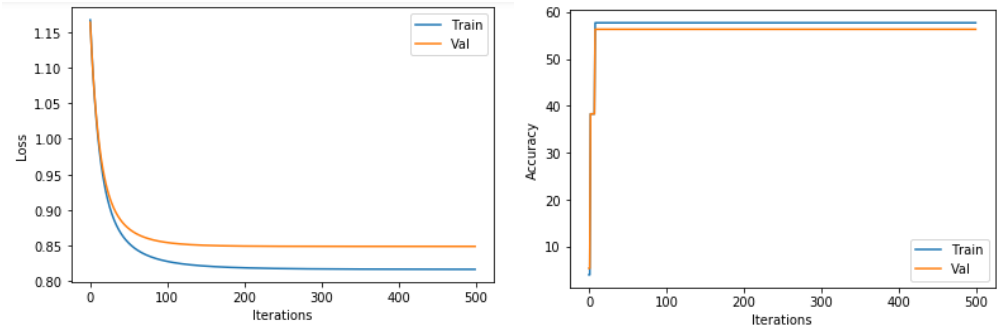

Figure 5: Model results with 2 and 4 neurons in the hidden layers.

With this model, the results over the test dataset are the following:

loss: 0.81684 - accuracy: 58.3624 %