CS577 Assignment 2: Final report

Theo Guidroz
Department of Computer Science
Illinois Institute of Technology

February 12, 2021

Abstract

This is a report for Assignment 2 of CS 577. The assignment served to gain a deeper understanding of neural networks.

Problem Statement

Build a three-layer neural network from scratch without using a GPU framework for a 3 class-classification. Use categorical entropy for the loss, sigmoid activation for the hidden layer, and softmax for the output layer.

Implementation details

This implementation question was divided into 4 main tasks:

1. Creating each node of the layer
2. Creating a forward and backward propagation
3. Creating a function to test the accuracy of the data
4. Test the data on 2 different sets

1. Creating each node of the layer

To create each node, it is necessary to know the formula for addition, multiplication, sigmoid, softmax as well as their derivative with respect to the inputs.
Below are the code used for each function:
Addition:

Function : `z = np.add(x,y)`
Derivative : `return [dz,dz]`

Multiplication:
Function : `z = np.dot(x,y)`
Derivative :
```
dx = self.y*dz
dy = self.x*dz
return [dx,dy]
```

Sigmoid:
Function : `z = 1/(1 + np.exp(-x))`
Derivative : `return x * (1 - x)`

Softmax:
Function :
```
exps = np.exp(x - np.max(x, axis=1, keepdims=True))
return exps/np.sum(exps, axis=1, keepdims=True)
```
Derivative: Because of the chain rule, the derivative of softmax and the derivative of loss with respect to cross entropy cancel out so there is no need to compute it.

Cross Entropy:
Function:
```
l = -np.log(x[0,y])
loss = np.sum(l)
```
Derivative:
```
n_samples = y.shape[0]
res = x - y
return res/n_samples
```
The functions will be used in the forward pass and the derivative in the backward pass.

2. Creating a forward and backward propagation

A class called modeled had been defined to hardcode the structure of this neural network. Then, forward and backward methods were implemented to calculate the loss and the updated weights. The forward method used the forward() implemented during step 1 and the backward method used the backward(). The chain rule was used to update the weights and biases after each training example (Stochastic gradient). The class model allows the user to change the number of neurons in the hidden layer. This useful to test the accuracy of the model with different structures of the neural network until the accuracy is satisfactory.

3. Creating a function to test the accuracy of the data

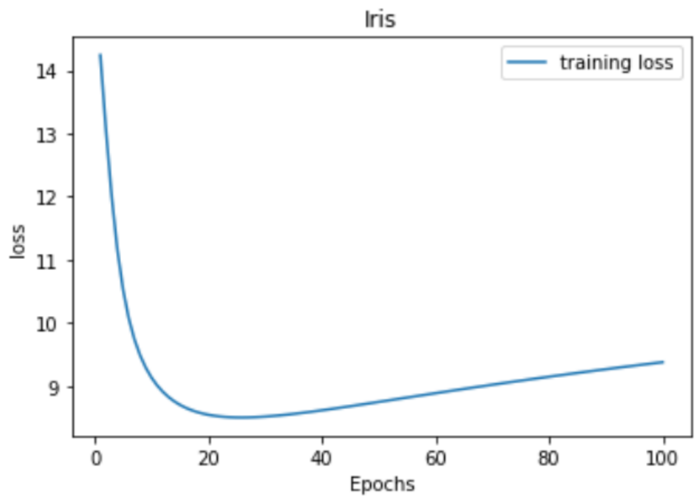
A predict method was implemented. It would return the class that had the highest probability after the forward pass. Then, an accuracy function was created. This would check the prediction made by the model and compare it to the real class given in the dataset. It would divide the number of good predictions by the number of rows in the dataset and multiply by 100 to find the percentage accuracy.

4. Test the data on 2 different sets

Initially, the goal was to download the datasets from the UCI database but unfortunately, there were some issues formatting the dataset for the neural network. Therefore, an alternative was to use datasets provided by Sklearn. Iris and a random classification dataset generator were used to test the model.

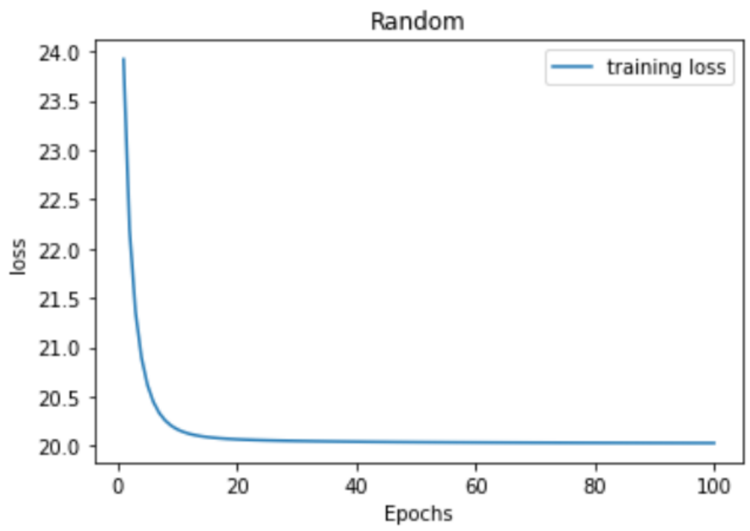
Results and discussion

Once the dataset was loaded, the model was trained and the number of neurons was increased until the model reached an accuracy of >90% or the model was trained more than 6 times. Below are the results for Iris and the random dataset.
Iris:



```
Training accuracy:  94.28571428571428
Test accuracy:  95.55555555555556
The number of neurons in the hidden layer:  104
```

Random:



```
Training accuracy:  57.142857142857146
Test accuracy:  56.666666666666664
The number of neurons in the hidden layer:  139
```

The model trained really well on the Iris dataset and with 104 neurons, the neural network had an accuracy of 95%. However, the model struggled to train on the random dataset. This might be due to the lack of correlation between the features and the label.
In conclusion, the model performs well on datasets where there is a correlation between the features and the labels.

References

Professor’s slides
https://scikit-learn.org/
pandas.pydata.org