Project1.1report

Rajat Nagpal
M.Tech SE
rajatnagpal@iisc.ac.in

Abstract

In this project, different neural network architecture (no. of layers, no of units per layer), hyper-parameters such as the learning rate, number of epochs, loss function, regularizer, etc. are analyzed. Since, outputs are discrete, cross-entropy loss function worked well for this task. Evaluation of performance for different settings for the architecture and the hyper-parameters has been done.

1 Introduction

Neural networks are the core of many deep learning algorithms. Neural networks are nothing but multi-layer perceptron. The effectiveness of any neural network depends on its architecture and the choice of hyper-parameters.

In this project, we are going to look at different aspects of the neural network architecture and the choice of hyper-parameters affecting the accuracy of the model. We will encounter the problem of over-fitting and under-fitting and we will try to overcome these problems by playing with neural network architecture and its hyper-parameters.

2 Analysis

2.1 Software 1.0

In software 1.0, the logic based implementation for the "fizzbuzz" problem has been done and the results are not surprising at all. The model is 100 percent accurate.

2.2 Software 2.0

Firstly, the neural network architecture with one hidden layer was used in this model but it was found that the architecture was so naive that it was not learning the data as desired. So, one more hidden layer was introduced in the architecture. The performance of the model was not encouraging. So, hidden layer dimensions were increased to 15 and 8 respectively. The gradient was shooting because the learning rate was set too high for the model. So, learning rate was decreased from 0.1 to 0.001 but then the number of epochs had to be increased as the model was learning at a very slow rate. So, learning rate was set to 0.01.

The model did not perform well and it was not giving the desired accuracy so the number of epochs were increased to 50,000, just to be sure. The performance of the model started to get better and the model was giving good training accuracy but the test accuracy was pretty bad.

It was found that the model was learning the exact orientation of the data and it was not performing good on the test data.Therefore, to make the model more robust, the data was shuffled after every 100 epochs and then fed into the neural network architecture. After that, batch-wise learning was used for the model and the batch-size was set to 32. At each iteration, a batch of 32 is taken from training data, randomly.

The regularizer factor is set to be 0.0001 to prevent the model from over-fitting on training data. Finally, the test accuracy of the model is found to be 93 percent.

3 Experiments and Results

The input is converted to 10 dimensional binary encoding and the output is converted to one hot vector encoding. The data is classified as four classes named as fizz, buzz, fizzbuzz and nota. Here, nota refers to "none of the above class".

In my final report, 2 hidden layers were used with sigmoid activation function applied to each layer. First hidden layer of dimension 15 and second hidden layer of dimension 8 is used. Output is 4 dimensional softmax probability distribution. Cross-entropy loss is used in the model.

Hyper-parameters and methods	
Learning rate	0.01
Batch-size	32
Epochs	50,000
Regularization factor	0.00001
Hidden layers	2
Optimizer	Adam
Output	Softmax
Loss	Cross-entropy
Activation function	Sigmoid
Neurons in HL1	15
Neurons in HL2	8

Table 1: Experimental conditions used for the final Software 2.0 model

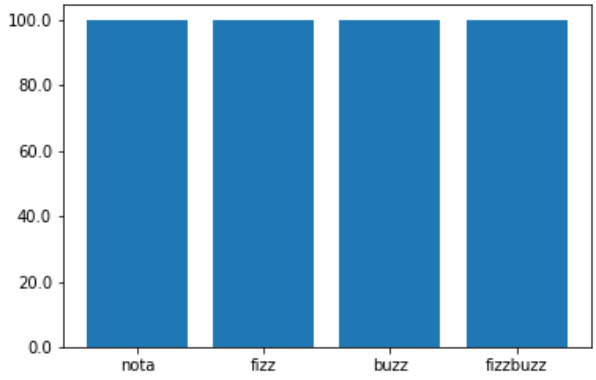

Figure 1: Software1 individual accuracy scores. Here, nota refers to "none of the above". Overall Accuracy is found to be **100** percent.

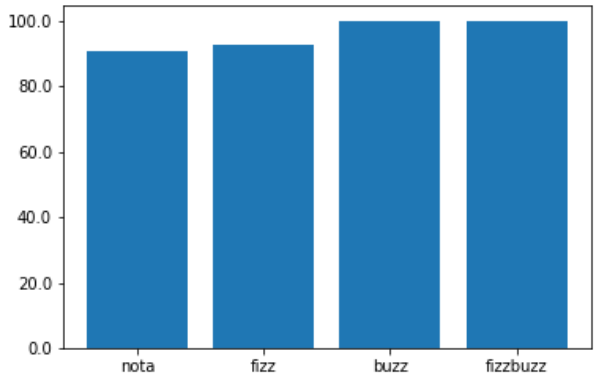

Figure 2: Software2 individual accuracy scores. Here, nota refers to "none of the above" Overall Accuracy is found to be **93** percent.

4 Conclusion

In software 2.0, it is easy to get better results with more data available with high variance. If the data is oriented in a certain way, it is more prone to over-fit to that orientation of the data. The distribution of the data is very important in training neural network architecture models.

Hyper-parameters play a vital role in enhancing the performance of neural network models. It is so important to tune the hyper-parameters correctly otherwise the model starts to show weird behaviour.

Software 1.0 is better than software 2.0 for this problem in space as well as time complexity. Also, the accuracy is better for software 1.0.