Mayank Gupta
SR No.- 17112
M.Tech AI

Assignment-1 Report

For training the neural network for Software 2.0 part of the assignment, run the command ‘python train.py’, to store the weights uncomment the last line in train.py. For using the trained network run the file main.py, this file will load the weights from the file ‘software2’ from the model folder and will write the output of test data in files ‘Software1.txt’ and ‘Software 2.txt’.

Approach:

Softare1.0

Used simple logic to classify inputs into one of the categories.

Result:- 100 percent accuracy on the dataset.

Software2.0:

Methods and hyperparameters used:

1) Converted the inputs to its 10-bit binary representations as this helps network to learn easily.

2) Modelled this problem as a classification problem with 4 categorical outputs, where output categories represent: Fizz, Buzz, FizzBuzz and input number.

3) Used neural network with 2 hidden layers with 15, 8 neurons respectively. Both layers use relu non-linearity as activation.

4) For training the network I used Adam optimizer with constant learning rate of 0.01 and weight decay of 0.001 to prevent overfitting, Cross-entropy loss with weight parameter to weight the loss of different classes accordingly. This is done because of class imbalance present in the data.

5) I am using 3000 epoch and batch size of 64, and I am shuffling the data in each epoch.

Result: Neural network takes around 1 minute to train(on CPU) on my laptop with intel i5-8^{th} gen. Network gives 96 percent test set accuracy. Accuracy on the training set is around 97 percent. Accuray of fizz, buzz, fizzbuzz is 97 percent, 100 percent, 99 percent respectively on the test set.

Analysis:

Some findings when I was experimenting with hyperparameters:

1) I used network of smaller size with 8 and 6 neurons in the two hidden layer. In this case the model was very sensitive to the hyperparameters and required few random initialization to get good results. This may be because the capacity of the networks was less for the problem.

2) Even after increasing the size of network, sometimes algorithm will get stuck in local minima or saddle point and the loss will be high. To remedy this, run the algorithm again with different initialization.

3) Some common points-

Learning rate matters a lot. So I tried a wide range to get faster convergence and good results.

If regularization(weight decay) is increased then network performance decreases sharply.

If batch size is too small, then the program is slower and convergence is also slower.

Using relu as activation and Adam optimizer, convergence is faster than other methods.