Machine Learning Report of Pro1.1

Haorui Dong
UBIT number: 50291149
Department of Computer Science and Engineering
University at Buffalo
haoruido@buffalo.edu

Abstract

After the fizzbuzz project, I have a preliminary understanding of machine learning. I knew about how to set up a training model, why we need to encode the original data. I learned about pandas and numpy, especially a lot of knowledge about tensorflow. I read the tensorflow edition of fizzbuzz, tensorflow is a very powerful package for machine learning with python. It can set up a neural network which we can define the number of layers and neurons.

1 Introduction of Tensorflow

1.1 Dataflow graph

First, at tensorflow website, I look at this graph. This graph has a specific description of tensorflow. We can understand the data flow clearly. The data get start at input, then after some hidden layers which has some bias and weight, the original data has been edited. Then they will pass through activation function, and optimizer. Tensorflow offer several kinds of activation functions and optimizers. Each of them has their feature and they are used in different condition of machine learning.

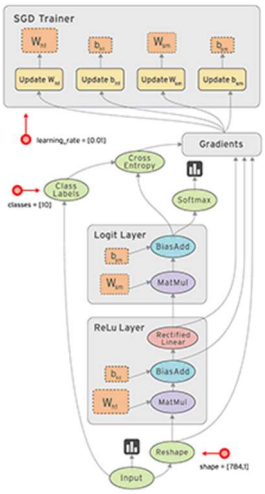

Figure1: Dataflow Graph

1.2 Activation functions and optimizer

The activation functions provide different types of nonlinearities for use in neural networks. There are several activation functions,:

- `tf.nn.relu`
- `tf.nn.sigmoid`

- `tf.nn.tanh`

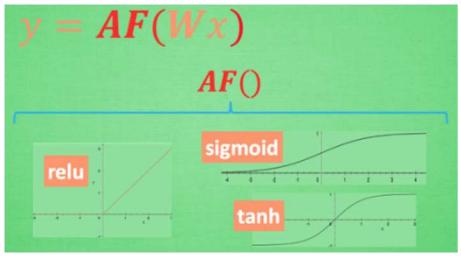

Figure2: Activation Function

In project1.1, we use GradientDescentOptimizer, this optimizer implements the gradient descent algorithm. There are also some other optimizers.

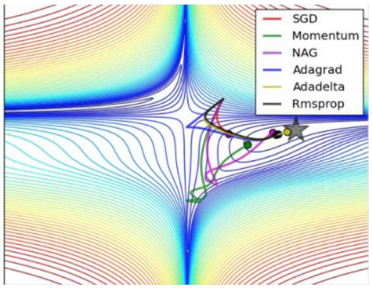

Figure3: Different Optimizers

As a beginner, I have a general concept of tensorflow and machine learning after read many resources of tensorflow. We use tensorflow to create a neural network and use lots of data to train this network. During the training, the program modifies bias and weight of neurons of hidden layers. With the training data grow up, the accuracy of model will grow up obviously. However, there will be some special conditions which will cause some mistake of machine learning and the model cannot set up correct. For example, overfitting.

1.3 Important functions in tensorflow

- `tf.placeholder(dtype, shape=None, name=None)`

This function gives user ability to set a variable later. We can use `feed_dict` to offer data for what we need.

- `tf.nn.softmax_cross_entropy_with_logits_v2(logits,labels,name=None)`

This function computes softmax cross entropy between `logits` and `labels`. Then we will use `tf.reduce_mean` to define a `error_function`. This `error_function` will be used in optimizer.

- `tf.Session()`
- `sess = tf.Session()`
- `sess.run()`

We often use these functions together to running TensorFlow operations.

2 Adjust parameters

2.1 Number of hidden neurons

I adjusted Tensorflow Model Definition. There are two parameters at the beginning of definition. The number of neurons and learning rate.

First, I keep the learning rate, and modify the `NUM_HIDDEN_NEURONS_LAYER_1`, this is how many neurons will be setup in layer 1. I modify it from 100 to 200. Before I modified, the accuracy is 90%

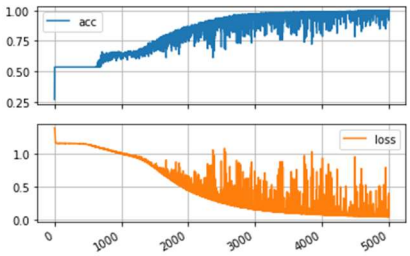

Figure4: Plot of Original Parameters

I reckon that more neurons will have a higher accuracy of test. Because it will have a more suitable bias and weight. After I modified, obviously the accuracy is higher than original. It is 96% now.

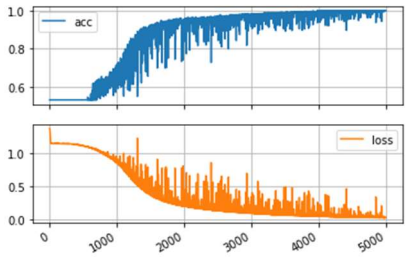

Figure5: Plot of 200 Neurons

Then I tried lots of different number to modify this parameter. I found that as this number get bigger, the training time will be longer. Before it finish, I think more neurons will get higher accuracy. However, it only has a little influence to accuracy. When I tried 1000 neurons, it takes a very long time to training. I think this must because there are too much neurons should be training in every EPOCH. But I get a very high performance of the result. It reached at 100% accuracy at last!

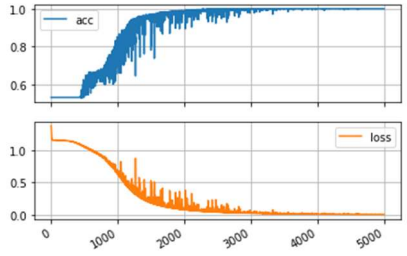

Figure6: Plot of 1000 Neurons

2.2 Learning rate

Then I keep neurons as 100, and try different number about learning rate. First, I make learning rate to 0.025.

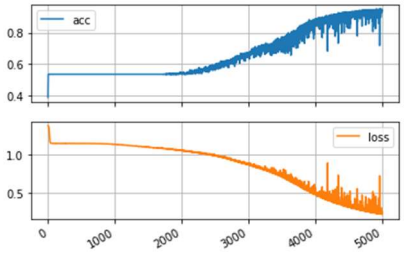

Figure7: Plot of 0.025 Learning rate

We can discover that the accuracy is more slowly to reach the highest number obviously. Then I try the 0.01 learning rate to maximize the effect of learning rate. But the result is weird. The accuracy is only 53%~54%. And the plot is also weird. Someone told me that may be overfitting.

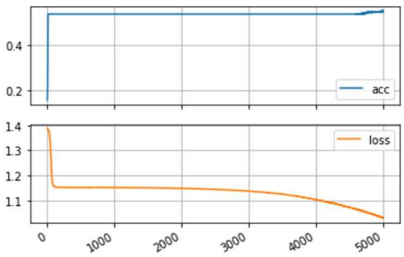

Figure8: Plot of 0.01 Learning rate

I set learning rate to 0.9. And the curve get a very high speed to the highest number. But the accuracy at last is not very satisfactory. It is only 84%.

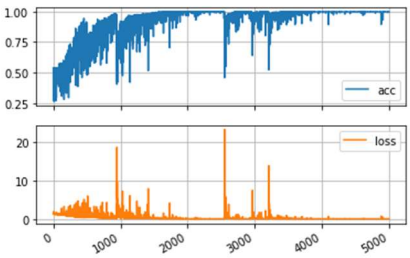

Figure9: Plot of 0.9 Learning rate

2.3 Activation Function

The original code use `tf.nn.relu()`, I tried many different functions under the condition of 0.05 learning rate and 200 neurons.

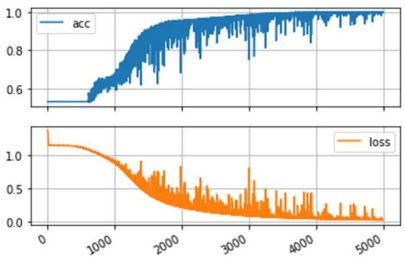

Figure10: Plot of `tf.nn.relu6`

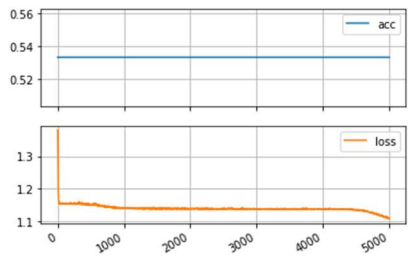

Figure11: Plot of `tf.nn.elu`

I try some other activation functions, but many of them look like Figure11, they have a same accuracy 53%. Only the relu6 can instead the relu. It has a better performance.

3 The result

At last, I keep the number of neuron as 500, and learning rate as 0.05. I use the relu6() as my activation function. I find the result under this condition is good enough. The test accuracy reached at 98%.

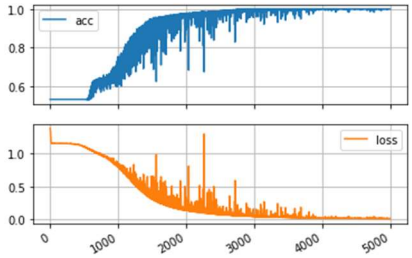

Figure12: Final result of training

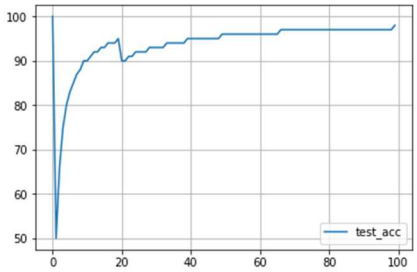

Figure13: Final result of test accuracy