Summary of a simple deep learning model for stock price prediction using TensorFlow by Sebastian Heinz

Jody Shu

March 9, 2020

This article introduces a high level approach to predict stock price using TensorFlow package in Python. The author has a link to his dataset, but unfortunately, the link is no longer available, but since the purpose of this summary is to understand this paper and how he coded the Python, I will ignore it and use my own data in the future for application.
The dataset is also spiltted into training and test data set which 80% *being training and* 20% for test data.
As similar to other papers I summaried for homework 6, the author uses MinMaxScaler package to scale the training set and test set separately. Moreover, there are four hidden layers, and started out with 1024 neurons and half the size in the subsequent layer until the 4th layer which is 128 neurons. Basically that is each two input data produces one output, and use that output as input for the subsequent layer. And then the author continues to talk about the setup of the bias and weight. For example, the second dimension (argument) of the previous layer is the first dimension of the current layer for weight matrices. In addition, the biases argument is the second argument of the current layer's weight matrix.

Next, the author discusses about the hidden layer activation functions.

There are many activation functions, and in this tutorial, the rectified linear unit (ReLU) is used in the example. The network architecture is a feedforward network. Further more, the author used mean squared error function between the true and prediction as the cost function and to minimize the cost function, the Adaptive Moment Estimation (Adam) optimizer is used. The whole procedure starts training the data by batches feed through the network until it reaches the output layer, and TensorFlow compares the output against the true, and then optimized and updates the weights and biases accordingly, and the procedure continues until all batches have been presented to the network which is called an epoch. The algorithm stops when the maximum number of epochs is reached. Once training set is done, the model can be evaluated by the test set. The result for the final test MSE is 0.00078 which is very low and the mean absolute percentage error of the forecast is 5.31% which is pretty good.