**Chapter 3**

**Implementation:**

Implementation contains the details description of the implementation procedure of our system.

Introduction:

In this chapter we will discuss details about how we implemented our stock market prediction system using gated recurrent unit (GRU) neural networks. This chapter also contains a brief description about the experimental tools we used to implement our project.

Software:

The programming language we used here is Python 3.7 [python]. Python is a powerful high-level, object-oriented programming language created by Guido van Rossum. It has simple easy-to-use syntax, making it the perfect language for someone trying to develop something creative as it comes with lots of modern packages.

For data analysis and data structures purpose we use Pandas which is an open source BSD-licensed library. We use NumpPy library for adding support for large, multi-dimensional arrays, matrices and for scientific calculation along with a large collection of high level mathematical functions to operate on these arrays [NumPy]. For collecting historical data at real time we use Pandas-datareader. For feature scaling we use Scikit-learn which is a machine learning library [Scikit-learn]. For developing our deep GRU neural networks we use Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow [Keras]. To visualize the result we use Matplotlib which is a Python 2D plotting library which produces quality figures in a variety of interactive environments across platforms [Matplotlib].

Implementation Real Time Data Collection:

We collect historical data from Yahoo Finance using pandas\_datareader library at real time. The DataReader method requires four parameters:

* The 1st parameter is for the ticker that represents the stock for a company.
* The 2nd parameter designates the online data repository from which historical prices are fetched.
* The third and fourth parameters denote, respectively, the start date and end date through which to collect historical data.

We then save our data into CSV file. We collect both training data set and testing data set.

Preparing the Inputs and Outputs:

After data collection, our task is to prepare the inputs and outputs. Our inputs are opening price, closing price, highest value and lowest value at time t and output is opening/closing values at time t+1. To do that we shifted our output one timestamp and remove the row containing NAN value.

Implementation of Feature Scaling:

The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.

Min-max normalization:

To scale the values between [0, 1], we use min-max normalization.

We use Scikit-learn MinMaxScaler to scale our data.

Reshaping the Inputs:

Reshaping is about changing the format of inputs. Before reshaping our input was a two dimensional array containing number of observations and features and we have four features. After reshaping our input data is a three dimensional array as time step is also included. We implemented this by using NumpPy library.

Initializing the GRU Neural Networks:

We called our object regressor and use sequential class as our data has a continuous sequence.

Creating the Inputs and Hidden Layers:

We changed our hidden layers from traditional GRUs by using sigmoid activation function instead of hyperbolic tangent.

The sigmoid function curve looks like a S-shape and the main reason why we use sigmoid function is because it exists between the range zero to one.

The equation for sigmoid is:

We use Keras to create this model with four input features and one time step using sigmoid activation function.

Creating the outputs Layers:

We used Keras, Densed class to create the output layers which is opening/closing prices at next time step.

Implementation of Compiling GRU Neural Networks:

We use ADAM optimization algorithm that can used instead of the classical stochastic gradient descent algorithm. Our loss function is mean squared error (MSE).

Implementation of Fitting the GRU to the training set:

Here we import our inputs and outputs at the neural networks. As we use mini-batch gradient descent, we tested our neural networks on several batch size and find the best result at batch size 32. For good result and good fitting we iterated our networks 200 times and find very low lost function which is better for testing.

Implementation of predicting result from test set:

We first applied feature scaling at our inputs. Our predicted results is the opening/closing prices of the next day.

Comparing the results with real values:

First we applied inverse feature scaling on our predicted results. We use Scikit-learn inverse transform method to get the values. After that we compare our results with real values.

Visualizing the results:

We use pyplot module from matplotlib to plot the real socks price and predicted stocks and visualize the results.

Evaluating the results:

Our model is a regression model and regression model is generally evaluated by calculating root mean square error (RMSE).

If is a vector of predictions generated from a sample of *n* data points on all variables, and is a vector of real values of the variable being predicted, then the with in sample root mean squared error (RMSE) of the predictor is computed as

We used Scikit-learn and math library to calculate the root mean square error (RMSE) between the real stock price and the predicted stock price.

Conclusion:

In this chapter we will discuss about implementation process related to this project. In the next chapter we discuss about the experimental results of this project.