Deep Learning Applications on Stock Market Data

Sushma Adari[1,3,⋆]

[1] *Computer Science + Astronomy, Astronomy Department, 1002 W Green St, Urbana, IL 61801, United States*

Abstract. This paper aims to outline the different methods of stock market analysis through the use of Neural Networks using the Keras Deep Learning Library for Theano and Tensorflow. The concentration will be on using a type of Recurrent Neural Network (RNN) known as the Long Short Term Memory (LSTM) and the Multilayer Perceptron (MLP) Neural Network.

1 Introduction

The recent years have shown an increase in interest in Artificial Intelligence (AI) in the science and computing community. A specific subset of AI known as deep learning, which uses neural networks, has gained particular attention as it is a powerful tool that is used across various disciplines including astronomy, physics, finance, mathematics and more. Neural Networks are a computational method based on the way the human brain recognizes and solves problems. The neural network, much like how the brain works, uses a knitted system of simple operational elements and artificial neurons that work in parallel to determine an outcome that is influenced by various pieces of information. The purpose of a neural network is to 'learn' patterns and rules from observational data and replicate the function of groups of neurons and axions within the brain. Neural networks then apply what they have learned on new sets of data in order to identity/predict the new data's characteristics. Formally a neural network can be defined as follows:

"Specifically, a neural network is a system composed of many simple processors - fully, locally, or sparsely connected - whose function is determined by the connec- tion topology and strengths. "

- DARPA Neural Network Study 1988

"Neural Networks are universal function approximators, they are powerful methods for pattern recognition, classification, and forecasting. Neural Networks are less sensitive to noise, chaotic components and, heavy tails better than most other methods."

-Designing a Neural Network for Forecasting Financial and Economic Time-Series

While Neural Networks are a powerful tool, it remains that they are incredibly complex as there are many factors that need to be taken into consideration before they are suitable for real world application. Often, their solutions are similar to black boxes since the parameters to create a successful Neural Network are generated in an experimental manner.

The rest of this paper will be written as follows: First a general description of a Neural Network will be provided which will also discuss important features that should be taken into consideration when building a neural network. Next a description of the Multilayer Perceptron Neural Network will be provided along with a brief overview of the backpropagation algorithm. The RNN and LSTM methods for building Neural Network will also be described. Finally an overview of Keras Deep Learning library Sequential() method will be described and its application to the stock market dataset will be outlined.

2 Artificial Neural Networks

2.1 Artificial Neurons

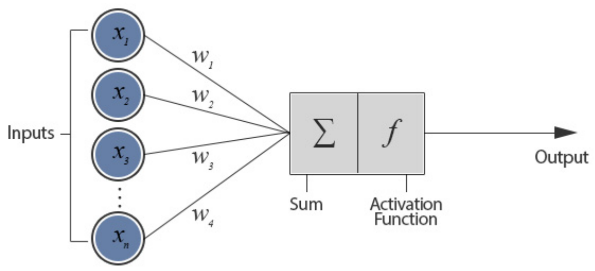


Figure 1: Artificial Neuron

The functionality of an artificial neuron is based off a typical neuron found within the human brain. Consider how the human brain makes a decision, the brain receives information and then takes into account the importance of each piece before coming to a final conclusion. This process is completed with complex chemical signals that the artificial neuron aims to replicate.

A simplified functionality of an artificial neuron receives N different input parameters, $x_1, x_2...x_N$. Each input is assigned a "weight", $w_1, w_2...w_N$ such that the weight of

⋆e-mail: sadari2@illinois.edu

the input is used to calculate the final output of the neuron. The neurons output is determined by whether or not $\sum_i^N x_i * w_i$ is greater or less than some threshold value. A negative value in weights is labeled an inhibitory connection and a positive weight value is labeled a excitatory connection.
There are various activation functions used to calculate the final output including, Gaussian, sigmoid, binary threshold and more.

2.2 Artificial Neural Network

A single artificial neuron is not capable of computing many different boolean functions. This is solved by connecting the output of some neurons to the input of other neurons, thus creating an artificial neural network. Neural networks are organized through layers, in the figure below, the leftmost most neurons are a part of the input layer and the rightmost neurons are a part of the output layer. The neurons located between the output and input layer is called the hidden layer. Artificial Neural Networks containing more than one hidden layer are referred to as multilayer networks. A neural network that uses outputs from one layer as an input the next layer are known as feedforawrd neural networks in which all of the information passes through the network in a single direction an example of this is a multilayer perceptron neural network (MLP). Neural networks that have connections within the current layer and/or to the previous layer are known as Recurrent Neural Networks (RNN).

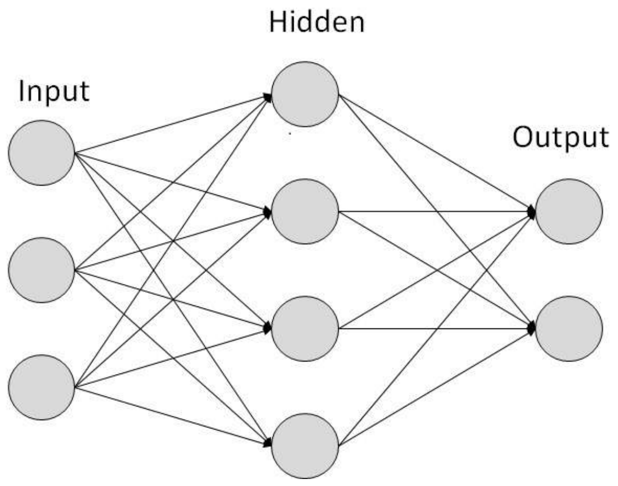


Figure 2: Artificial Neuron Work

3 Multilayer Perceptron Neural Network

Multilayer perception neural networks are feedforward networks that involve one or more hidden layers. Each layer is fully connected to the next layer such that each neuron, excluding input neurons, has a non-linear activation function.
(NEED TO CHANGE FIGURES HERE, CREATE OWN FIRST)
The training algorithm used with multilayer perception neural networks is a supervised learning algorithm known as backpropagation. The back propagation method is a generalization of the Least Mean Squares method which focuses on minimizing the error function.

3.1 Backpropagation

Backpropagation is the widley used algorithm for supervised learning learning with MLP neural networks. The goal of backpropagation is to compute each of the weights of the neural network with respect to an error function (also referred to as a cost function or loss function) through partial derivatives. In the case of backpropagation, the error function refers to the difference between the training input example and the expected out after the training input has been propagated through the network.
In order for backpropagation to be successful there are two assumptions that need to be made about the error function. The first, is that the error function can be written as an average $E = \frac{1}{n} \sum_x E_x$ over all unique training examples x. The reason behind this is so the backpropagation algorithm can compute the partial derivatives for each training example x and then be able to generalize it to the total error function.
The second assumption that needs to be made about the error function is the, is that the error function can be written as a function of outputs from the neural network, such that $E = f(a)$ where a is the outputs of the neural network.
These assumptions allow for the successful use of the backpropagation algorithm. The algorithm itself can be divided into to two phases. A forward pass and the backward pass.
The forward pass starts with the insertion of a vector x_N into the input layer of the neural network, and the outputs are passed through as forward propagations through the network until the estimated final output of the neural network is generated.
The backward pass begins after the estimated final output of the neural network is generated and are provided with the expected output. Starting at the final layer the expected output is moves back through the neural net where δ_N, the difference between the estimated output and the expected output is calculated for all hidden and output neurons while simultaneously updating the weights of each in order for the network to provide an estimated output that is closer to the expected value.

4 Recurrent Neural Network

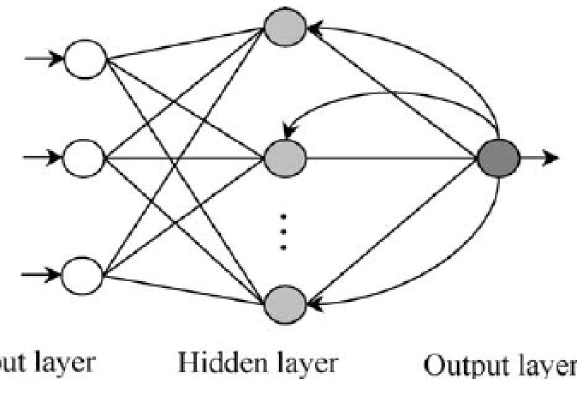


Figure 3: Recurrent Neural Network

Recurrent Neural Networks have become popular amongst scientists as they have become extremely successful when applied to various research subjects. Unlike MLP neural netowrks, RNNs are not feedforward and can contain connections within the current layer and to previous layers. Within a traditional feedforward network the input vector, the output vector and the amount of computational steps is a fixed value. However, within a recurrent neural network, computation can be computed on sequences of the input vector or output vector. As such recurrent neural networks are more powerful than a feedforward network. In a feedforward neural network the output vector is influenced by the input vector, in a RNN the network is influenced by not only the input vector but also all of the history of previous inputs. Like MLP neural networks, RNNs also rely on backpropagation but they use a variation known as backpropagation through time (BPTT), the key difference between the standard backpropagation algorithm and BPTT is that gradients are summed up over the layers since RNNs are not feedforward and can have dependencies on the current layer and/ previous layers. BPTT ensures that all changes in weights are accounted for.
One of the advantages of using an RNN is the ability to look back to previous information and use it in the current task. RNNS are capable of gathering relevant information however only to a certain limit. If the information the RNN needs to compute is relatively near the current, the space needed to hold that information is also small. In cases, where the RNN needs to reach much farther back to get information, the RNN, in practice, is no longer able to learn to connect to the previous relevant information.
This is known as the vanishing gradient problem. In a recurrent neural network information passes through the neural net undergoes multiple stages of multiplication as a result when gradients are computed the values become too small for the network to learn on.
Due to the vanishing gradient problem, a variation of the RNN known as Long Short Term Memory (LSTM) was developed so that neural nets can learn long term dependencies.

4.1 Long Short Term Memory

LSTMs are a special variation of the RNN designed to avoid the long-term dependency problem, by remembering long sequences of information without having to learn them.
LSTMs are built with memory gates with a unique feature known as a gated cell which regulates what information can be stored, written, or read through gate openings and closings. The gates are made of a sigmoid neural net layer which outputs values between zero and one through pointwise multiplication operations, indicating how much information should be passed through. Zero corresponding to let no information pass and one corresponding to let all the information pass.

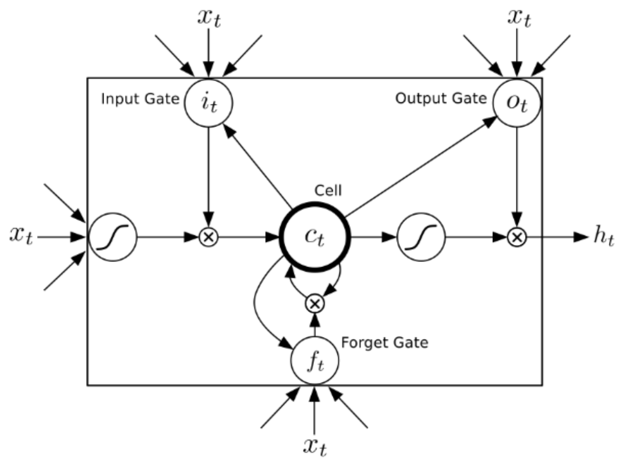


Figure 4: Long Short Term Memory

In order to build a successful LSTM, a combination of gates are used to determine what information is relevant to the current problem. First within the forget gate layer, the LSTM neural network decides whether or not to keep the information in the current cell state. This layer computes a value between zero and one in order to determine how much information should be remembered in the current state. The next layer is the input gate layer which determines which new information, if at all, should be included in the cell state, once again a value between zero and one is computed to determine the extent of new information that should be remembered. Finally, the output gate layer determines which information in the current state should be outputted.
LSTMs are used far more frequently than the general recurrent neural network since they account for the vanishing gradient problem and are capable of solving a larger variety of problems.

5 Keras

"Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."
-Keras Documentation
The implementation of the MLP neural network and LSTM neural network for predicting trends in stock market data was completed through Keras. In this section, a brief outline of the Keras Sequential model will be discussed followed by an brief explanation of the implementation of the MLP and LSTM neural networks and the factors that were taken into consideration upon building them. The sequential model is Keras is a linear stack of layers. First construct a Sequential model and add the layers of computation in the desired order. The first add will correspond to the input layer in which the the input dimension should be defined, this is also where the type of network can be be define such as LSTM. In Keras, fully connected layers are defined by the Dense class in which the first parameter is the number of neurons in the layer, the second is the initialization method, and the third is the activation

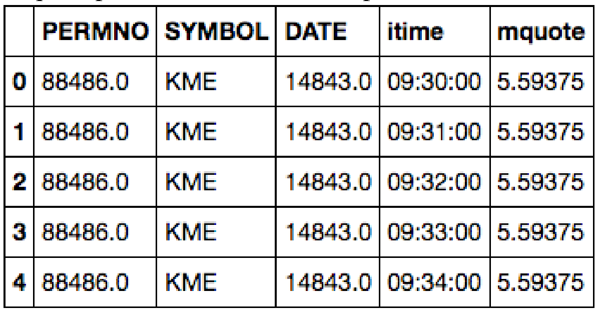
method. Often, the hidden layers and the input layer use the rectifier activation function in order to have better performance and the output layer activation function is chosen based on what format the intended predictions should take. A full list of the initializations and activations can be found in the Keras Documentation.
After defining the layers of the neural network using the Sequential model, the learning process of the model will be configured using the compile method. Compilation is mandatory as Keras will determine the most effective way to represent the neural network for training. Keras finds the best representation of the neural network such that it runs on the appropriate hardware such as CPUs and GPUs. In order to compile the model, a number of variables need to be defined that should be specific to training the network. A error function, defined as loss function on the documentation, and an optimizer should be defined along side any of the other optional metrics. A full list of available optimizers and loss functions can be found in Keras Documentation.
The next step is to train the network with the training set using the fit function. The fit method will run through the dataset in a fixed number of iterations known as epochs defined by the nb_epoch argument. Each epoch can be further divided into input output pairs defined by the batches argument which defines the number of evaluations that occur before the weights in the network are updated.
The next step is the evaluate the model. Performance can be either evaluated on the same dataset that training was done on or on another dataset. Performing evaluation on the same dataset does not provide any information on how well the algorithm performs on new data, therefore a dataset that the model has not seen before will better represent how accurate the model performs with new data. the evaluate function takes in the same input and output parameters passed as when the fit function was called. A list of evaluation metric will be returned after the model computes the loss across all the evaluations as well as any additional metrics that were defined with the model.
Finally, after determining that the model is accurate it can be used for prediction using the predict function. This is simply done by calling predict on the model using a new input set of data and the result will be returned as defined by the output layer of the network.
This is the general outline to be followed when creating a Keras Sequential model, often defining the layers and the parameters to include are found through trial and error as there is no completely accurate way of defining the what parameters should be chosen when building the model.

6 Stock Market Data Applications

This section will describe the dataset, the preprocessing steps taken to use the dataset, and finally there will be a brief outline of the Keras LSTM and MLP neural network implementation.

6.1 Data Format

The dataset consists of minute level stock data spanned from the years 1993 to 2010, the dataset is organized in five columns names PERMNO, SYMBOL, DATE, itime, and mquote. PERMNO corresponds to a unique identification number of each stock, SYMBOL corresponds to the stock's ticker symbol, DATE is the date in Julian date format, itime is the time between 9:30:00 AM to 4:30:00 PM and mquote corresponds to midquote price of the stock at the specified date and time.



	PERMNO	SYMBOL	DATE	itime	mquote
0	88486.0	KME	14843.0	09:30:00	5.59375
1	88486.0	KME	14843.0	09:31:00	5.59375
2	88486.0	KME	14843.0	09:32:00	5.59375
3	88486.0	KME	14843.0	09:33:00	5.59375
4	88486.0	KME	14843.0	09:34:00	5.59375

Figure 5: Dataframe of Stock KME

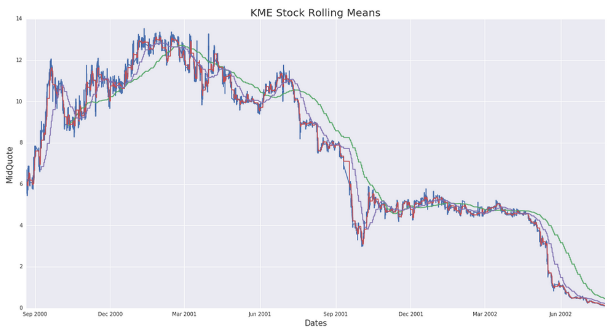


Figure 6: Time-Series Plot of Stock KME with Rolling Means

6.2 Data Preprocessing

The data was initially received in the format of 13671 sas7bdat files and was converted into csv files using the sas7bdat.py module. After all the files were converted, a brief run through of them was completed and empty csv file within the dataset were discarded. With the csv files, a sqlite database was constructed where each table within the database corresponds to an individual stock.

6.3 Keras Multilayer Perceptron Neural Network

First, the dataset was split into train and testing dataset. In this instance the dataset is split as follows: 67% train and 33% test.

References

Online references will be linked to their original source, only if possible. To enable this linking extra care should be taken when preparing reference lists.
References should be cited in the text by placing sequential numbers in brackets (for example, [1], [2, 5, 7], [8–10]). They should be numbered in the order in which they

are cited. A complete reference should provide enough information to locate the article. References to printed journal articles should typically contain:

- The authors, in the form: initials of the first names followed by last name (only the first letter capitalized with full stops after the initials),
- The journal title (abbreviated),
- The volume number (bold type),
- The article number or the page numbers,
- The year of publication (in brackets).

Here are some examples [**? ? ? ?**]: