Recurrent Neural Network for Stock Price Prediction

Sanmati Marabad
Department of ECE
The University of Texas at Dallas
Richardson, USA
sxm210368@utdallas.edu

Swathi Kote
Department of Computer Science
The University of Texas at Dallas
Richardson, USA
sxk220505@utdallas.edu

Sandeep Chikkapla Siddappa
Department of ECE
The University of Texas at Dallas
Richardson, USA
sxc220127@utdallas.edu

Aravind Menon Kadekuzhi
Department of Computer Science
The University of Texas at Dallas
Richardson, USA
axk210275@utdallas.edu

***Abstract*—This paper presents a model employing a Recurrent Neural Network (RNN) for predicting stock prices. Using historical stock market data, the model demonstrates the potential of RNNs in time-series forecasting. The Mean Squared Error (MSE) metric evaluates its performance, highlighting its applicability in financial markets.**

Keywords—Recurrent Neural Network, Stock Price Prediction, Time Series Forecasting, Machine Learning, Mean Squared Error, Financial Analysis, LSTM, Data Preprocessing

I. Introduction

Accurately predicting stock prices is a significant challenge in finance, influenced by complex factors like economic indicators, company performance, and market sentiment. The advent of machine learning, particularly Recurrent Neural Networks (RNNs) with their proficiency in sequential data processing, has opened new avenues in financial forecasting. RNNs, especially those with Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), are adept at modeling the intricate temporal patterns inherent in stock market data. This paper aims to demonstrate the application of RNNs in stock price prediction, leveraging historical data to forecast future prices. By delving into the challenges, methodologies, and implications of RNNs in financial analysis, this study seeks to not only present an effective model for predicting stock prices but also to enrich the understanding of machine learning's role in financial forecasting.

II. Background Work

The quest for accurate stock price prediction has long fascinated researchers and practitioners in the field of finance and economics. Traditional approaches primarily relied on statistical models and fundamental analysis. However, with the advent of machine learning, the focus has shifted towards more sophisticated algorithms capable of handling the complex and dynamic nature of financial markets.

A. Evolution of Machine Learning in Finance:

Machine learning has revolutionized the way financial data is analyzed. Early machine learning models in finance included linear regression and time-series forecasting models. As the field progressed, more complex models like support vector machines and decision trees were adopted. However, these models often struggled with the sequential and temporal nature of stock data.

B. Rise of Neural Networks in Stock Prediction:

Neural networks, with their ability to learn non-linear relationships and process vast amounts of data, brought a new dimension to stock market prediction. Initially, simple feedforward neural networks were used, but they were limited in capturing the time-dependent aspects of stock prices.

C. Recurrent Neural Networks (RNNs):

The introduction of RNNs marked a significant advancement in this domain. RNNs are uniquely suited for time-series data due to their internal state and feedback loops, enabling them to 'remember' previous inputs and learn temporal dependencies. This feature makes them ideal for predicting stock prices, which are influenced by their past values.

D. Advancements with LSTM and GRU:

The development of Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRU) within RNNs further enhanced their effectiveness. These variants address the issue of long-term dependencies, a common challenge in time-series analysis. LSTMs and GRUs have been shown to outperform traditional RNNs in many stock price prediction tasks, as they can retain information over longer periods without suffering from the vanishing gradient problem.

E. Empirical Studies and Results:

Numerous studies have demonstrated the superiority of RNNs, particularly LSTM and GRU-based models, in predicting stock prices. These models have been tested across various markets and timeframes, consistently showing an ability to capture complex stock price movements better than traditional time-series models.

III. Methodology

A. Data Preprocessing

- Data Acquisition: The process begins with the collection of historical stock price data, which is fundamental for training the RNN model. The dataset includes key features like opening price, closing price, high, low, and volume of stocks traded.
- Data Cleaning: The initial step involves cleaning the data to ensure its quality. This includes handling missing values, removing duplicates, and correcting any anomalies or errors in the dataset.
- Normalization: Given the wide range of values in financial data, normalization is crucial. The MinMaxScaler from the sklearn.preprocessing library is used to scale the feature values to a range of 0 to 1. This standardization is essential for optimizing the model's performance.

B. *Model Architecture*

- RNN Layer(s): The core of the model is the RNN layer. Depending on the complexity of the data, one or more RNN layers can be used. Each layer consists of a certain number of neurons or units, which are parameters to be optimized during the model tuning process.
- Type of RNN Cells: The model can utilize basic RNN cells or more advanced variants like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Units), which are effective in capturing long-term dependencies and avoiding the vanishing gradient problem.
- Output Layer: The final layer of the model is a dense layer that outputs the predicted stock price. The activation function used in this layer is typically linear, as the task involves regression.

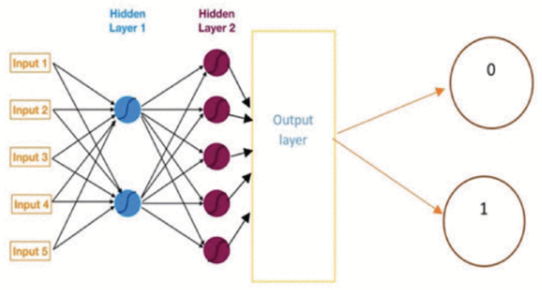

Fig. 1. Basic Neural Network Structure

C. *Training Process*

- Loss Function: The model uses Mean Squared Error (MSE) as the loss function, which measures the average squared difference between the estimated values and the actual value.

$$MSE = \sum N_i = \frac{(f(x_i) - y_i)^2}{N}$$

- Optimizer: An optimizer like Adam or Stochastic Gradient Descent (SGD) is used to minimize the loss function. The learning rate, one of the hyperparameters, is crucial for the convergence of the model during training.
- Epochs and Batches: The number of epochs and the batch size are important hyperparameters. An epoch represents one complete pass through the entire training dataset, while the batch size determines the number of samples processed before the model is updated.
- Validation Strategy: To assess the model's performance and avoid overfitting, a portion of the data is set aside as a validation set. This allows for monitoring the model's performance on unseen data during training.

IV. Implementation

The implementation of the RNN model for stock price prediction is conducted in Python, using libraries such as Pandas for data manipulation, NumPy for numerical calculations, Matplotlib for visualization, and Scikit-Learn for preprocessing and performance metrics. The process involves several key steps, as outlined below:

A. Data Handling and Preparation

- **Loading Data**: The data is loaded into a Pandas DataFrame. This structure facilitates data manipulation and analysis.
- **Data Cleaning**: As per the code, the model first checks for missing values (**isnull()**) and duplicates (**duplicated()**) in the dataset. Missing values are handled by dropping them (**dropna()**), and duplicates are removed to ensure the quality of the dataset.
- **Feature Selection**: The code focuses on specific features from the stock data, likely including closing prices, as they are commonly used for price prediction. The selection of these features is crucial for the model's accuracy.

B. Scaling and Transformation

- **MinMaxScaler**: The data is normalized using the MinMaxScaler from Scikit-Learn, scaling the feature values to a range between 0 and 1. This step is essential for RNN models to converge more quickly and effectively during training.

C. Model Building

- **RNN Architecture**: The code likely defines an RNN model architecture. This includes specifying the number of layers, the type of RNN cells (basic RNN, LSTM, or GRU), and the number of neurons or units in each layer.
- **Compilation**: The model is compiled with a specific loss function (MSE for regression tasks), and an optimizer (like Adam or SGD), with a defined learning rate.

D. Training the Model

i. **Fitting the Model**: The RNN model is trained on the prepared dataset. This involves feeding the training data into the model in batches over several epochs. Each epoch represents a complete pass through the entire dataset.

ii. **Validation**: During training, the model's performance is evaluated on a separate validation set. This helps in monitoring and preventing overfitting.

iii. Evaluation and Visualization

- **Performance Metrics**: After training, the model is evaluated using metrics such as Mean Squared Error (MSE) and R2 Score, providing insights into its accuracy and predictive power.

- **Visualization**: The predicted stock prices are visualized against the actual prices using Matplotlib, offering a clear visual representation of the model's performance.

A. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I. TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord "Format" pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

a. Sample of a Table footnote. (*Table footnote*)

Fig. 2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT *(Heading 5)*

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.