Title: Stock Price Prediction Extension with Graph Neural Networks

1. Introduction

The first stage in my research was dedicated to exploring various modules and techniques through which I can enhance the process of stock price prediction. Among the mentioned modules, I am interested in the usage of RNNs, LSTM networks, ARIMA for time series forecasting, and Random Forests for regression. After considering the strengths and limitations of the various approaches, I opted for GNNs since they can model the relationships between different stocks, an important virtue to capture interdependencies present within the financial market. GNNs provided a unique advantage in incorporating the relational structure of the stock data, which was not achievable using other approaches.

In financial markets, stock prices are not taken in isolation but interrelate in a complicated interrelatedness with other stocks, sectors, and global economic factors. Traditional prediction models, including LSTM and ARIMA, usually operate at the individual stock level and probably fail to recognize connectedness within the same market. For this, I extended my stock prediction approach using Graph Neural Networks to model relationships among multiple stocks; hence, my predictions would be more accurate and robust.

2. Summary of Research Work

GNNs are designed for neural networks operating on graph-structured data. In financial markets, these may be viewed as nodes of a graph, while edges may correspond to the relationships among them, like their correlations, industrial links, or value chains. GNNs learn from relationships and would thus be an appropriate choice of model for interdependencies influencing stock prices.

In this respect, GNNs have been applied to social network analysis, recommendation systems, and more recently in financial markets. The idea is simply to integrate the relational context of stocks through a GNN into the model, which would provide greater insight into market dynamics than the traditional approach, which handled stocks as

independent entities. I decided on a GCN because a GCN can only capture the local structures of the neighborhood, which is good for modeling relationships among stocks.

3. Implementation

My implementation involved the following major steps:

3.1 Market Graph Construction

I constructed a market graph with a few related stocks, namely, Apple-AAPL, Microsoft-MSFT, and Google-GOOGL. Each stock would be a node, while edges would get added between nodes based on significant correlations within the historical prices. I pulled down the necessary stock data from Yahoo Finance and built the graph with NetworkX. These edges were weighted by the strong level of correlation between stocks, so the model differentiates between strong and weak relationships.

The steps followed to implement this in my code are: import all the necessary libraries: networkx for constructing the graph, and yahoo_fin for the stock data. A graph is constructed using NetworkX, adding a node for every stock. Then, it calculates the correlation in historical prices between every pair of stocks, adding weighted edges to the graph if that correlation is above some threshold.

3.2 Graph Neural Network Model

I then implemented the GNN model using PyTorch Geometric. The model contains two graph convolution layers: the first performs the convolution to capture the features of neighbouring nodes in higher dimensions, while the second involves ReLU activation, hence introducing non-linearity, which is extremely important to be able to learn complicated relationships by the model.

This is followed by a second layer that reduces the output to a single value representing the stock price for the next time step. It supports the model in making an educated prediction by aggregating information from neighboring nodes about the wider context of the market.

Based on the above explanation, I will proceed to implement the following in my code: I defined a custom class for GNN using PyTorch Geometric.

The two `GCNConv` layers are instantiated; the first is to lift the dimensionality of input features and the second one back to a single output. Use ReLU for activation after the first layer, introducing non-linearity. Train the model using historical stock data, running 100 epochs using the Adam optimizer with the MSE loss function.

Moreover, I have further enhanced my predictions by incorporating the GNN model with my previously used models, namely the LSTM and ARIMA models. Output from GNN served as an additional feature for the LSTM model. That is, the output of the GNN, which captures the influence of related stocks, is appended to the features used by the LSTM. That's how it enables the LSTM to capture temporal patterns together with relational insights.

I also apply ARIMA to predict stock prices from historical data. The ARIMA model is good at capturing the linear trend and seasonality from time series data. I also combine ARIMA with LSTM and GNN to use the strengths of each model: ARIMA on linearity, LSTM on nonlinear temporality, and GNN on interstock relations.

To do this, I changed my code to:

• I then used the GNN output, after it had been trained, as an additional feature.

• I edited the loading data function to add the GNN output as a new column in the dataset.

• Then, it was the turn for the LSTM model to be trained with both the original features and the output from the GNN.

The final prediction on test data is an ensemble of the three models using a weighted average. The weights were assigned based on the performance during validation, higher the negative error lower the weight and vice-versa.

3.4 Evaluation

I evaluated my GNN-LSTM-ARIMA ensemble model performance using mean squared error as a metric. Indeed, the result proved that the performance of the ensembled model was higher than each individual model's performance, showing its effectiveness that might arise due to the inclusion of inter-stock relationships using GNNs. I also did a sensitivity test to analyze the impact of different weights in the resulting ensemble model. From sensitivity analysis, it follows that the GNN contributed most to decrease the total error of forecast by capturing the important features of the inter-stock dependency.

4. Results

The MSE of the ensemble model was way lesser compared to those of the lone models using LSTM and ARIMA. Since the GNN was a part of it, I could capture the influence from related stocks onto the target stock. The GNN was successful in modeling the relationship among the stocks, thus helping improve the accuracy of the ensembled model on the whole.

The following plot depicts the actual vs. predicted stock prices for Apple (AAPL):

- The blue line shows the actual stock prices.
- The green line represents the predictions by LSTM.
- The red line, which is the ensemble prediction, follows actual prices closely. Thus, this is the benefit that is derived because of the integration of GNN.

The results showed that the ensemble approach worked a lot better under volatile market conditions where interdependencies among stocks played an essential role in price movement. My model could adapt to such market dynamics relatively better by integrating GNNs.

5. Conclusion

By bringing Graph Neural Networks into my approach to the stock price predictions, I managed to capture these complex interdependencies among stocks and hence have more accurate and robust predictions. In this regard, GNNs allowed me to consider relationships among different stocks beyond the individual analysis of the stock and, therefore, enrich the predictive power of my model. This approach underlined the potential of GNN in financial forecasting and opened new avenues toward the investigation of interconnected data across other domains.

The integration of GNN with LSTM and ARIMA models provided a complete view of the prediction of stock prices by modeling temporal, linear, and relational perspectives of data. The multi-dimensional approach resulted in improved performance, especially during volatile market conditions, thereby proving the essence of including various modeling techniques.

6. Future Work

While the GNN-enhanced model showed promising results, there are several avenues for future work. It would be particularly interesting to extend the market graph further to involve a number of stocks and take into consideration other types of relationships, such as links about industry sectors, supply chain dependencies, or even sentiment analysis of news and social media. By including sentiment data, the insight regarding investor behavior could help further improve the predictive capability of the model.

Another possible enhancement could be exploring other GNN models, such as Graph Attention Networks, which will help capture the different importance of distinct relationships. Each edge will be assigned variable weights with the exploitation of attention mechanisms, therefore allowing a more focused model on the most influential relations. That again could improve my predictions about stock prices by underlining the most important factor for driving their movement.

Another direction I could go is temporal graph neural networks, in which the graph structure changes temporally. In this way, I can model how the relationships between stocks change over time, yielding a much more dynamic view of the market.

7. References

- Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.

- Velickovic, P., et al. (2018). Graph Attention Networks. In ICLR.

- Yahoo Finance API: https://finance.yahoo.com/

Hamilton, W., Ying, Z., & Leskovec, J. (2017). In-ductive Representation Learning on Large Graphs. In NeurIPS.