Predicting Stock Prices with Relational Learning

John Theurer
University of California, Los Angeles
theurerjohn3@gmail.com

Yugantar Prakash
University of California, Los Angeles
yugantarp@g.ucla.edu

Utsav Munendra
University of California, Los Angeles
utsavm9@g.ucla.edu

Fabrice Harel-Canada
University of California, Los Angeles
fabricehc@cs.ucla.edu

Abstract

The economy is often discussed with the two concepts central to this class - Probability and Networks. Probability plays a foundational role in the modern economy of debt, credit, and risk, and features prominently in our course. Buyers and sellers, as presented in the economy, could be viewed as nodes in a directed graph. We propose using a probabilistic programming language to model the price of a single stock, and using a graph neural network (GNN) to model an industry. **Part 1** entails modeling a stock and/or index using ARIMA. **Part 2** entails using GNNs to predict stock prices with both historical stock data and relational data. Lastly, we learn a new correlational embedding to describe company relations solely from stock price movements. Our results indicate that relational data is important for achieving gains that surpass the passive performance of holding an index of stocks and that this relational data can be learned to a large extent only from stock price movements. We present all project code and the data we worked with at: `https://github.com/fabriceyhc/ppl_gnn_stocks`.

1 Introduction & Motivation

This paper explores using GNN's and ARIMA models to predict price changes in the stock market. Modeling financial data is often split into technical analysis, which focuses purely on the stock prices, and fundamental analysis, which focuses on the actions and activities of the company. Graph neural networks offer an opportunity to combine these two approaches. We chose this project because it would help us understand each of these approaches.

In particular, in this paper we look at the time-series nature of stock market, and we take two approaches to modeling and predicting stock prices for various companies: Markov Processes in the form of Autoregressive Integrated Moving Average (ARIMA) models and using relational information about the elements in an industry (such as supply chain relationships) through Graphical Neural Networks (GNNs). By comparing a GNN that is trained only on stock data and correlations to one that includes market research, we are able to examine the advantage fundamental analysis provides.

2 Background

We explore the use of Probabilistic Programming libraries to predict stocks using an AR model, which is typically programmed purely mathematically, using known formulas. The equation of an AR(1) model, also known as a random walk, is:

$$X_{t+1} = \psi \cdot X_t + \epsilon_t$$

where X_i is the stock's closing price at the end of day i. [1]

34rd Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

We then employ GNNs to understand relationships between elements of industries. We believe that stock prices of individual companies in an industry are dependent on each other to the point that the aggregate movement of their stock prices mimic the trends that the industry as a whole follows due to external factors. At the same time, *ceteris paribus*, the stock prices should have a negative correlation with each other as, internally, the elements of the industries try to compete for the same, finite market. In particular, by using the stock prices of various companies of an industry and learning the graphical relationship between them, we hope to better understand the connections between these elements and the nature of stock prices in reference to these two opposing forces.

For this analysis, we use data from the New York Stock Exchange, or NYSE and the Nasdaq Stock Market, or NASDAQ. These are the two largest stock markets by total market capitalization, and are the the standard bearers of the American financial world. Additionally, we gather market fundamentals from Wikidata, a Wikimedia run open source knowledge base that gives information about buyers and sellers. Market fundamentals are also gathered from similar sources about the sector and industry of the company.

3 Related Work

3.1 Random Walks

Autoregressive (AR) models, also known as random walks, cover a large variety of datasets, modeling random walks is an important tool in any time series toolbox [1]. To achieve this, we decided to model the data of a specific stock or index over a short period of time. It is widely believed that the daily behavior of stock prices can be modeled as a random walk [2], and a number of statistical analyses have supported that conclusion[3]. Analyzing this dataset is both interesting to us, as we want to be more financially aware, and serves the purpose of analyzing random walks.

3.2 Graphical Neural Networks

Graph Neural Networks (GNNs), initially proposed by Sperduti *et al.* [4] in 1997, have shown remarkable results in modeling relationships between elements by learning their graphical representation. Spurred on by the success of convolutional neural networks (CNNs) [5] in the computer vision domain [6], a large number of methods that redefine the notion of convolution for graph data have yielded new convolutional GNNs, sometimes known as ConvGNNs or Graph Convolutional Networks (GCNs) [7, 8, 9, 10, 11]. In this work, we use GCNs, but refer to them interchangably as GNNs and we refer interested readers to an excellent survey of various GNN types in [12].

Stock predictions using GNN's often use known facts about the real world fundamentals of the company to produce a graph, which the neural network than uses to make predictions from [13, 14, 15, 16]. These graphs are produced from information such as companies industry, their buyers and sellers, and their investor relations. This enables the methodology to combine technical and fundamental analysis.

4 Technical Contributions

4.1 Random Walks

In this section, we seek to answer one primary research question:

RQ1. Can Autoregressive models be written in a probabilistic programming language?

We will use NumPyro, a probabilistic programming package for Python, to estimate the distribution of ϵ_t using a Markov Chain Monte Carlo (MCMC) algorithm.

The dataset used were the stocks listed in the NASDAQ. To evaluate the success of the model, we will apply the following trading strategy:

1. Each day, we will invest $100 in each stock estimated to rise in value, and sell that stock at the end of each day.
2. This will be compared to investing the same sum total quantity of money each day in the Vangaurd Total Stock Market index fund, VTI, and selling it at the end of the day.

4.2 Graphical Neural Networks

For this portion of the project we expect to address two primary research questions:

RQ2. Can we use GNNs to predict stock prices using relational data?

RQ3. Can we approximate company relations purely from stock price movements using GNNs?

4.2.1 Replication Effort Experience with GNNs

We plan to address RQ2. similarly to the approach taken by Matsunaga et al. in [13] using the stock and relational data collected by [14]. Namely, we plan to use both the stock rolling window analysis upon stocks in the NASDAQ index to ensure that the relational model we are learning performs well across all a wide variety of time slices. This is essentially a replication work where we implement a similar GNN in Pytorch [17] using the DGL framework [18]. We describe our approach as follows:

1. Replicate the GNN from [14] in Pytorch using the temporal rolling window analysis in [13].
2. Train two separate instances of the GNN, both on NASDAQ stock data (including the 5-day, 10-day, 20-day, and 30-day moving average features) but differing on the relational data: one with Wikidata and the other with sector-industry relations. This is intended to establish some baselines for which set of relational data performs better while fixing the stock data.
3. Predict stock prices to identify the top k performing stocks. At the beginning of each timestep, invest a fixed amount of money (e.g. \$100) according to some strategy (i.e. put all our money on the top performer, or diversify by selecting a distribution across the top k in some manner to be determined). Sell at the end of the time step to realize the return.
4. Evaluate how well these two GNNs are able predict stock prices based on their Mean Square Error (MSE) and the cumulative investment return ratio (IRR).
   (a) IRR can be compared against two metrics:
      i. The total return had we just invested in the full NASDAQ index.
      ii. The optimal return for the index as computed by greedily picking the best performing constituent stock from the index and investing entirely in that stock.
   (b) NOTE: we normalize the IRR by dividing the earned return by the optimal returns possible (i.e. investing in the best performing stock at each time step) to (1) make the results between NASDAQ and NYSE comparable and (2) remove the affect of selecting a particular daily investment amount.

4.2.2 Relational Learning Contribution

We observe that one of the major limitations of previous work using relational information for stock prediction is that the matrix tracking the relationships between companies is based on data from a particular point in time. Therefore, a hypothetical matrix of N companies has dimension $N \times N$ company-to-company relationships when we ideally want an $N \times N \times T$ tensor for predictions at each time step T. This need motivates RQ3 where we plan to approximate the target $N \times N \times T$ relational matrix R from stock information alone. We describe our approach as follows:

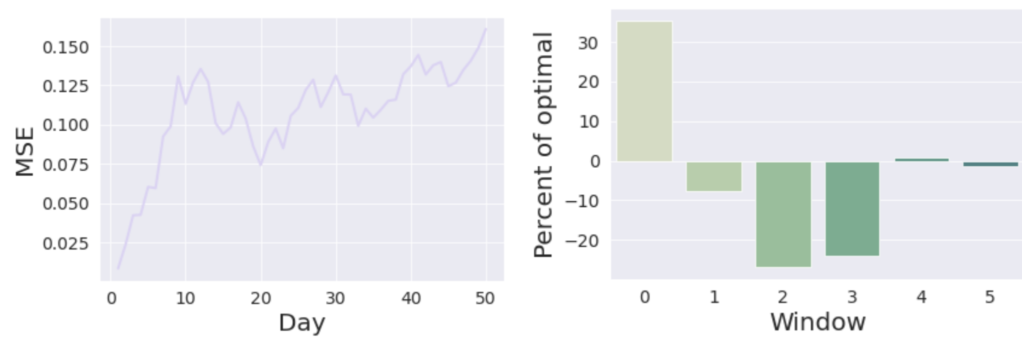
1. Initialize target relational tensor R:
   (a) In lieu of using one of the popular initialization techniques [19, 20], we intend to incorporate prior knowledge of the likely relationships based on 30-step rolling correlations of stock price movements. Specifically, for each time step T, generate a Pearson correlation matrix from the previous five time steps. Note: the first five time step correlations will necessarily be truncated due to a lack of preceding data.
   (b) Stack all correlations to yield initialized tensor $R_0 \in \mathbb{R}^{N \times N \times T}$.
2. Update R:
   (a) Using the gradients obtained from the training loss, update R to minimize further loss.
   (b) $R_{t+1} \leftarrow R_t + \beta \cdot \nabla_w J(w)$, where w are the learnable weights of the GNN, $J(w)$ is the training loss, and β is the learning rate for R updates. Note: let α be the standard learning rate for graph training, then we propose to set $\beta < \alpha$ to reflect the observation that company relationships change more slowly than stock prices.
3. Evaluation:
   (a) Constructing the ground truth G_T matrix for comparison:
   [14] uses an $N \times N \times K$ tensor for K different kinds of relationships. For our purposes, relationship type is not as important as the net effect of the relationship's influence. Therefore, we will average over the K dimension to acquire a $N \times N$ target matrix, which represents the influence of all companies on one another at time T.
   (b) We propose to slice the final time step T from our R tensor to get our final relational matrix R_T to compare against G_T via the Frobenius norm: $||G_T - R_T||_F$.

5 Findings

5.1 ARIMA Evaluation

We invest 100 dollars each day in the company predicted by AR(1) model to have the highest percent increase in stock. We compare the returns accumulated from the above strategy to the returns we would have gained by investing in the company that actually had the highest percent increase in closing price of its stock.

We used a window of 200 training days, followed by predicting the next 1 day, which we call the 200/1 split. After this, the training period is moved to the end of the last training period, and the process repeated until the end of the dataset. Using this rolling window, we averaged a loss of 60 cents per training window.



(a) MSE

(b) % of Optimal Return

Figure 1: (a) Mean squared error of AR(1) model as a function of days since last data. (b) Return of AR(1) model in each window.

The AR model is not particularly good at selecting stocks with positive returns, as we did worse than break even half the time.

We switched our methodology to 500 training days and 50 testing days to evaluate how the model did as time passed. Our loss went from a total of \$ .57 in the first window to a total of \$ 16.01 in the second window.

The mean squared error rose from 0.85% of the stocks value to 16% of the stocks value over the 50 day period in the AR(1) model applied to the NASDAQ stock prices in the 500/50 window structure.

RQ1. Probabilistic programing langauges can perform the same analysis as traditional approaches. However, our implementation in NumPyro was slower as compared to a conventional implementation.

5.2 GNN Evaluation

This section discusses the results of the GNN evaluation with respect to our two evaluation metrics — MSE and IRR expressed as a percent of the optimal return – for each of our relational data types. Each evaluation was conducted with various parameters, for which the most important are shown in Table 1 of the Appendix. Note that, the # of windows is dynamically sized depending on the selection of training size (i.e. $\lfloor \text{No. of steps/Training Size} \rfloor = \lfloor 1215/200 \rfloor = 6$. Windows slide by a length equivalent to "training size" so that no temporal correlations are adjusted for gradients in more than one window.

Lastly, as a kind of sanity check for our novel correlational approach, we decomposed the evaluation to consider pre-training and post-training performance. This approach trained a model as usual — which includes updating the temporal correlational tensors, which we view as a kind of external model parameter — and then when it came time for evaluating the performance, we used the trained correlational tensors to report "Trained Correlations" and then reverted back to the original tensors to compute the performance for "Untrained Correlations." Since the training was confirmed to work for NASDAQ, ultimately demonstrating the viability of our approach, we did not conduct the same experiment for the NYSE market and only report the "Trained Correlations."

5.2.1 Can we use GNNs to predict stock prices using relational data?

Table 2 shows the total MSE loss across all windows of testing and shows "Sector Industry" as having the lowest error for NASDAQ and "Wikidata" having the lowest error for the NYSE. This result aligns with the performance seen in Figure 3 and Tables 2 and 4 in the Appendix which also shows that overall, using a GNN with relational information sourced from "Sector Industry" and "Wikidata", respectively, yields the highest returns of all relations considered.

Relation	MSE
Untrained Correlations	0.00168
Trained Correlations	0.00133
Wikidata	0.00050
Sector Industry	**0.00046**

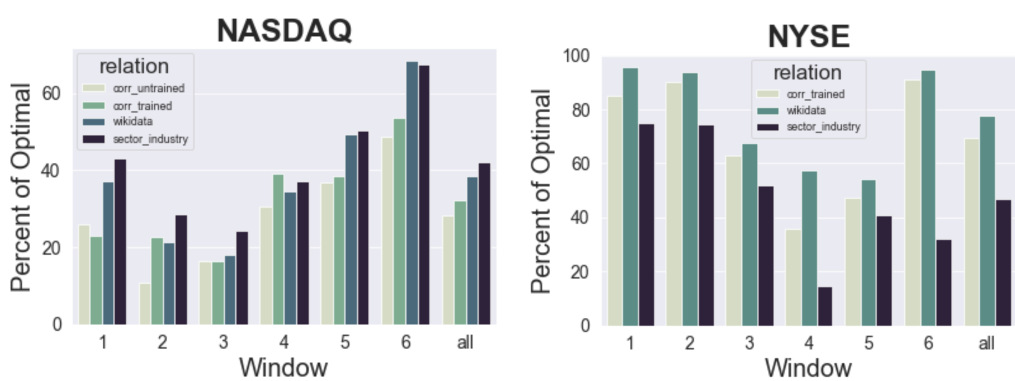
(a) NASDAQ

Relation	MSE
Trained Correlations	0.00012
Wikidata	**0.00009**
Sector Industry	0.00019

(b) NYSE

Figure 2: Total MSE Loss for relational data on (a) NASDAQ and (b) NYSE stock predictions.

The earned returns for each relation type shown in the aforementioned tables is computed by investing $\$100.00$ into the top $k = 1$ stock at each of 114 selected time steps for testing. More specifically, the model predicts stock prices for each day and we pick the company's stock that does best on that day. Then we lookup the true performance of that company and multiply the return by the daily investment amount of $\$100.00$. Tables 3 and 5 in the Appendix shows the raw results and includes the baseline "Mean Return", which is what an investor would have earned had they invested in the full index evenly. A return at or below the mean return would indicate a complete failure for any stock prediction project because a passive holding of the full index does better. Our testing interval happened to have a negative mean return and all approaches managed to make positive gains. This result is encouraging for our particular approach as well the entire enterprise of stock prediction using relational data.



(a) NASDAQ (b) NYSE

Figure 3: Comparison of relational data on NASDAQ stock predictions. [1]

RQ2. Yes, GNNs are an effective way of predicting stock prices because they always do significantly better than the baseline buy-hold strategy (i.e. "mean return").

5.2.2 Can we approximate company relations from stock price movements using GNNs?

In this section, we address RQ3 by comparing the Frobenius norms of the difference between each pairing of relational datasets in a given market. By averaging over the temporal dimension T of the

[1] Unfortunately, the Sector Industry model only completed training on the first two rolling windows.

correlational tensor, we obtain a matrix in $\mathbb{R}^{N \times N}$. By averaging over the relational type dimension K of the Wikidata and Sector Industry tensors, we obtain matrices of the same dimension as above. Then we can compare the Frobenius norm between Wikidata and Sector Industry to find a baseline of closeness. If the $\|\cdot\|_F$ between our correlational tensor and the other relational tensors is comparable (i.e. within a small ϵ distance), then we can answer the RQ in the affirmative.

NASDAQ	**Correlational**	**Wikidata**	**Sector Industry**
Correlational	—	239.84	239.16
Wikidata	239.84	—	2.62
Sector Industry	239.16	2.62	—

(a) NASDAQ

NYSE	**Correlational**	**Wikidata**	**Sector Industry**
Correlational	—	309.16	308.13
Wikidata	309.16	—	5.70
Sector Industry	308.13	5.70	—

(b) NYSE

Figure 4: Frobenius Norms between all relational datasets for (a) NASDAQ and (b) NYSE.

Unfortunately, Figure 4 shows the difference between our relational tensors is significant. The correlational tensors are much different (i.e. range [-1, +1]) from the relational tensors created froms craping web data on known company relationships (i.e. either $\{0, 1\}$). However, our results from RQ2 suggest that we have actually learned some aspect of the relationships because we could leverage our tensors with comparably close performance. This suggests that we may have learned a different representation of the company-to-company relationships after all.

RQ3. It depends. The Frobenius norms between our new temporal tensors and the standard relational tensors are large, but the relationships we did learn were still useful for stock prediction. We demonstrated this by using gradient adjusted stock correlations over a sliding 30-day window. Ultimately, our approach may represent another valid viewpoint from which company relations can be evaluated.

6 Conclusion

We found that probabilistic programming languages were able to model AR(1) processes as well as traditional mathematical approaches. However the GNNs were much better predictors of stock price movements and they always did better than the baseline strategy of buy-and-hold of the entire index. The implementations of all of our models is available at `https://github.com/fabriceyhc/ppl_gnn_stocks`.

The GNNs provided a far more fascinating set of results. While the Frobenius norms indicate that our correctional tensor does not replicate the same kind of relational embedding as either the sector industry or the Wikidata matrix, we do demonstrate that it is a successful usage for accurately predicting stock prices within a reasonable degree of optimality. While the MSE for the correlation matrix was rather poor for NASDAQ, approximately 3 times that of the sector industry and Wikidata MSE's, the correlation GNN performed very well on the NYSE. Importantly, the correlation matrix consistently outperformed the mean return, indicating that the selected stock was consistently better than the market. Ultimately, the relationships learned were useful for the stock prediction as demonstrated from gradient adjusted stock correlations and we believe our results represented another viewpoint from which company relations can be seen.

7 Appendix

7.1 GNN Evauation Parameters

Parameters	Values
Market	NASDAQ, NYSE
Relation	sector_industry, wikidata, correlational
Learning rate (model) α	0.001
Learning rate (correlations)* β	0.0001
# of epochs	100
Training size	200
Validation size	20
Testing size	20
# of windows	6

Table 1: Highlighting important experimental parameters. For parameters with more than one value, every combination is evaluated (e.g. NASDAQ is paired with each relation using the remaining default parameter values). *"Learning rate (correlations)" does not apply to wikidata or sector_industry.

7.2 NASDAQ Results

Window	**Untrained Corr.**		**Trained Corr.**		**Sector Industry**		**Wikidata**	
	Earned	% Optimal	Earned	% Optimal	Earned	% Optimal	Earned	% Optimal
1	$72.23	25.95%	$63.60	22.84%	**$120.01**	**43.11%**	$103.71	37.25%
2	$27.95	10.85%	$58.51	22.71%	**$73.83**	**28.66%**	$55.07	21.38%
3	$35.40	16.26%	$35.58	16.34%	**$53.05**	**24.37%**	$39.07	17.95%
4	$79.31	30.36%	**$102.43**	**39.21%**	$96.57	36.96%	$89.86	34.39%
5	$110.77	36.67%	$115.70	38.30%	**$151.92**	**50.29%**	$148.70	49.22%
6	$114.49	48.66%	$125.86	53.49%	$158.56	67.39%	**$160.72**	**68.31%**
all	$440.16	28.35%	$501.68	32.32%	**$653.94**	**42.12%**	$597.14	38.47%

Table 2: Comparison of relational data on NASDAQ stock predictions. Using the relations contained within "Sector Industry" usually results in the highest performance.

Relation	Window	MSE	Earned Return	Optimal Return	Mean Return	% of Optimal
corr_untrained	1	2.2E-3	72.23	278.41	-0.09	25.95%
corr_untrained	2	2.8E-3	27.95	257.59	-0.49	10.85%
corr_untrained	3	1.4E-3	35.40	217.72	-0.45	16.26%
corr_untrained	4	2.2E-3	79.31	261.26	0.22	30.36%
corr_untrained	5	588.0E-6	110.77	302.09	0.55	36.67%
corr_untrained	6	595.0E-6	114.49	235.30	-0.18	48.66%
corr_untrained	all	1.7E-3	440.16	1552.38	-0.09	28.35%
corr_trained	1	683.0E-6	63.60	278.41	-0.09	22.84%
corr_trained	2	429.0E-6	58.51	257.59	-0.49	22.71%
corr_trained	3	1.0E-3	35.58	217.72	-0.45	16.34%
corr_trained	4	740.0E-6	102.43	261.26	0.22	39.21%
corr_trained	5	1.9E-3	115.70	302.09	0.55	38.30%
corr_trained	6	4.0E-3	125.86	235.30	-0.18	53.49%
corr_trained	all	1.3E-3	501.68	1552.38	-0.09	32.32%
wikidata	1	530.0E-6	103.71	278.41	-0.09	37.25%
wikidata	2	355.0E-6	55.07	257.59	-0.49	21.38%
wikidata	3	614.0E-6	39.07	217.72	-0.45	17.95%
wikidata	4	564.0E-6	89.86	261.26	0.22	34.39%
wikidata	5	464.0E-6	148.70	302.09	0.55	49.22%
wikidata	6	422.0E-6	160.72	235.30	-0.18	68.31%
wikidata	all	495.0E-6	597.14	1552.38	-0.09	38.47%
sector_industry	1	515.0E-6	120.01	278.41	-0.09	43.11%
sector_industry	2	313.0E-6	73.83	257.59	-0.49	28.66%
sector_industry	3	550.0E-6	53.05	217.72	-0.45	24.37%
sector_industry	4	522.0E-6	96.57	261.26	0.22	36.96%
sector_industry	5	436.0E-6	151.92	302.09	0.55	50.29%
sector_industry	6	396.0E-6	158.56	235.30	-0.18	67.39%
sector_industry	all	458.0E-6	653.94	1552.38	-0.09	42.12%

Table 3: Comparison of relational data on NASDAQ stock predictions.

7.3 NYSE Results

Window	Trained Corr.		Sector Industry		Wikidata	
	Earned	% Optimal	Earned	% Optimal	Earned	% Optimal
1	\$238.49	85.10%	\$210.05	74.95%	**\$267.69**	**95.52%**
2	\$172.59	90.22%	\$142.35	74.41%	**\$179.81**	**93.99%**
3	\$204.37	62.74%	\$169.45	52.02%	**\$219.50**	**67.38%**
4	\$84.08	35.70%	\$33.74	14.32%	**\$135.80**	**57.66%**
5	\$129.93	47.17%	\$112.03	40.66%	**\$149.83**	**54.39%**
6	\$326.75	91.00%	\$114.74	31.96%	**\$341.16**	**95.02%**
all	\$1,156.21	69.34%	\$782.36	46.92%	**\$1,293.80**	**77.59%**

Table 4: Comparison of relational data on NYSE stock predictions. Using the relations contained within "Wikidata" always resulted in the highest performance.

Relation	**Window**	**MSE**	**Earned Return**	**Optimal Return**	**Mean Return**	**% of Optimal**
corr_trained	1	7.00E-05	238.49	280.25	-0.36 az	85.10%
corr_trained	3	1.07E-04	204.37	325.76	-0.42	62.74%
corr_trained	4	2.27E-04	84.08	235.54	0.01	35.70%
corr_trained	5	1.27E-04	129.93	275.49	0.76	47.17%
corr_trained	6	1.07E-04	326.75	359.06	-0.03	91.00%
corr_trained	all	1.16E-04	1156.21	1667.40	-0.36	69.34%
wikidata	1	5.70E-05	267.69	280.25	-0.36	95.52%
wikidata	2	4.40E-05	179.81	191.31	-0.03	93.99%
wikidata	3	9.00E-05	219.50	325.76	-0.42	67.38%
wikidata	4	1.32E-04	135.80	235.54	0.01	57.66%
wikidata	5	1.11E-04	149.83	275.49	0.76	54.39%
wikidata	6	8.80E-05	341.16	359.06	-0.03	95.02%
wikidata	all	8.70E-05	1293.80	1667.40	-0.36	77.59%
sector_industry	1	1.21E-04	210.05	280.25	-0.36	74.95%
sector_industry	2	1.42E-04	142.35	191.31	-0.03	74.41%
sector_industry	3	1.83E-04	169.45	325.76	-0.42	52.02%
sector_industry	4	3.02E-04	33.74	235.54	0.01	14.32%
sector_industry	5	1.94E-04	112.03	275.49	0.76	40.66%
sector_industry	6	1.96E-04	114.74	359.06	-0.03	31.96%
sector_industry	all	1.89E-04	782.36	1667.40	-0.36	46.92%

Table 5: Comparison of relational data on NASDAQ stock predictions.

7.4 Feedback

With regards to the GNN experiments, we knew ahead of time that our correlational tensor would have dimensions $\mathbb{R}^{N \times N \times T}$, but did not anticipate how unwieldy it would be to generate, store, and use. For the NASDAQ market, it took $\sim$ 8hrs to initialize the tensor and $\sim$ 5GB to store 1.28 billion correlations while NYSE took $\sim$ 24hrs to initialize and was $\sim$ 16GB to store 3.67 billion correlations. These figures required us to take on the additional task of splitting the tensor by time step, loading up each one on demand, and ultimately retrofitting the code to exploit GPU computation so that training over 6 windows for 100 epochs each could complete in approximately 5hrs per combination of market $\in$ {NASDAQ, NYSE} and relational data $\in$ {correlational, wikidata, sector_industry}. The final GPU run time comes to a total of approximately 30hrs but many, many more were required for troubleshooting and miscellaneous restarts caused by issues like running out of memory mid-run. Next time, we'll know how difficult large tensor manipulation is and focus on research directions with more tractable components.

NOTE: the GitHub repository is several GB in size so we're not going to be able to submit the data to CCLE. Please use the GitHub.

References

[1] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[2] Paul H. Cootner. The random character of stock market prices. *MIT Press*, 1964.

[3] Louis Bachelier. Théorie de la spéculation. *Annales Scientifiques de L'Ecole Normale Supérieure*, 17:21–88, 1900. Reprinted in P. H. Cootner (ed), 1964, The Random Character of Stock Market Prices, Cambridge, Mass. MIT Press.

[4] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *Trans. Neur. Netw.*, 8(3):714–735, May 1997.

[5] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[9] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[10] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *CoRR*, abs/1705.07664, 2017.

[11] A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[12] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[13] Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. Exploring graph neural networks for stock market predictions with rolling window analysis. *ArXiv*, abs/1909.10660, 2019.

[14] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2):27, 2019.

[15] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1655–1658, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Yiying Yang, Zhongyu Wei, Qin Chen, and Libo Wu. Using external knowledge for financial event prediction based on graph neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2161–2164, New York, NY, USA, 2019. Association for Computing Machinery.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.