Graph Representation Learning in the Brazilian Stock Market

Luiza Santos

Instituto Tecnológico da Aeronáutica

November 21, 2024

Outline

Motivation

- Graphs capture **relationships** in data:
  - Useful for modeling systems like social networks, transport, biology, and markets.
- Models that ignore relationships overlooks critical insights.
- **Objective**: Apply embedding algorithms to learn features in a graph representation of the stock market that can be used in any downstream task.

Graphs and the Stock Market

Some applications of graphs in the stock market are:

- Portfolio optimization. [2] [5]
- Stock classification. [12]
- Risk management. [9] [13]
- Price Prediction. [11] [1]

Graph Construction Approaches

- **Knowledge-based** [4]
  - Most popular right now.
  - Uses alternative data such as Tweets, industry reports, news mentions and others.
- **Relationship-based** [13]
  - Uses market indicators (e.g., prices, volumes).
  - Edges are determined through historical data analysis using correlation coefficients or time series models.

Graph Construction

Node Representation and Residuals

To construct the graph, we followed a multi-step approach:

- **Step 1**: Remove systematic market movements:
  - Regress each stock's returns on the market index' returns.
  - Calculate the residuals.
  - Ensured that broader market trends do not dominate the relationships between stocks.

- **Step 2**: Construct a residuals' correlation matrix:
  - Use Pearson's correlation coefficient [10]:

$$\rho_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} \left(\frac{r_{ik} - \bar{r}_i}{s_i} \right) \left(\frac{r_{jk} - \bar{r}_j}{s_j} \right)$$

  - Terms:
    - ρ_{ij}: Correlation coefficient between stocks i and j.
    - n: Total number of timestamps.
    - r_{ik}, r_{jk}: Residuals of stocks i and j at timestamp k.
    - $\bar{r}_i, \bar{r}_j$: Mean residual for stocks i and j.
    - s_i, s_j: Standard deviations of residual for stocks i and j.
  - Captures the extent to which information in one stock explains another.
  - Correlation values range from -1 to 1:
    - $+1$: Strong positive correlation.
    - -1: Strong negative correlation.

- **Step 3**: Transform correlation coefficients to represent stronger positive correlation as closer nodes:

$$a_{ij} = 2 \times (1 - \rho_{ij})$$

  - Highly correlated stocks ($\rho_{ij} \to 1$) map to smaller distances ($a_{ij} \to 0$).
- **Step 4**: Construct a weighted, fully connected graph:
  - Nodes: Stocks.
  - Edge Weights: a_{ij}.

Sparsification of the Graph

Motivation

- Fully connected networks are computationally expensive:
  - High time and memory requirements.
- **Objective**: Create a sparse network that retains critical information.
- **Challenge**: Simply removing weak correlations (below a threshold) can result in isolated nodes.
  - Isolated nodes lack context, hindering embedding algorithms.
  - Leads to poor embeddings and affects downstream tasks (e.g., clustering, anomaly detection).

Sparsification of the Graph

Minimum Spanning Tree (MST)

- To address the issue of isolated nodes, while making the graph sparse, we used the **Minimum Spanning Tree (MST)** approach.
- MST properties:
  - Subset of edges that connects all nodes.
  - Ensures no cycles.
  - Minimizes total edge weight while preserving connectivity.
- **Advantages**:
  - Guarantees a connected network.
  - Retains the most important relationships.

Kruskal's Algorithm for MST

- Chosen algorithm: **Kruskal's Algorithm**.
- Complexity: $\mathcal{O}(E \log E)$, where E is the number of edges.
- Steps:
  1. Sort all edges by weight (ascending).
  2. Use a disjoint set to track connected components.
  3. Add edges to the MST, avoiding cycles, until all nodes are connected.
- Efficient and widely used for MST problems.

Kruskal's Algorithm for MST

Pseudocode

Require: Graph $G = (V, E)$ with weighted edges
Ensure: Minimum Spanning Tree T

1: $T \leftarrow \emptyset$ {Initialize empty MST}
2: **for** each vertex $v \in V$ **do**
3: MAKE-SET(v)
4: **end for**
5: $E \leftarrow SORTED(E)$ {Sort edges by increasing weight}
6: **for** each edge $(u, v) \in E$ **do**
7: **if** FIND-SET(u) $\neq$ FIND-SET(v) **then**
8: $T \leftarrow T \cup \{e\}$
9: UNION(u, v)
10: **end if**
11: **end for**
12: **return** T

Kruskal's Algorithm for MST

Visualization Complete Problem

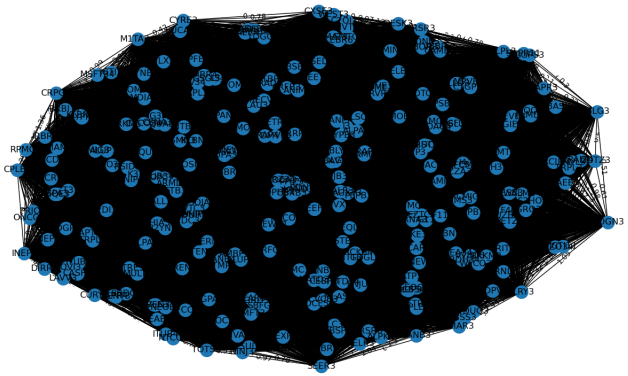

Figure 1: Fully connected graph.

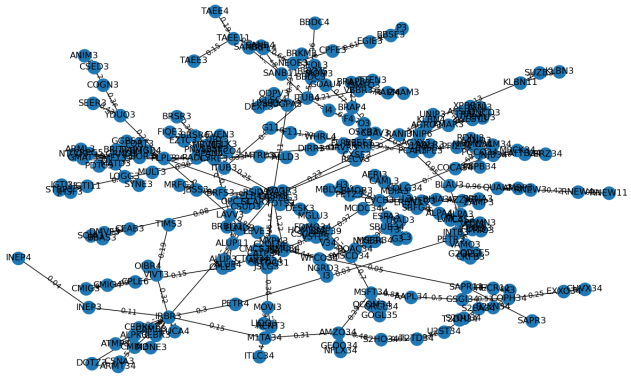

Figure 2: Minimum Spanning Tree.

Kruskal's Algorithm for MST

Visualization Simplified Problem

It's easier to visualize the transformation in a hypothetical world with only 13 stocks:

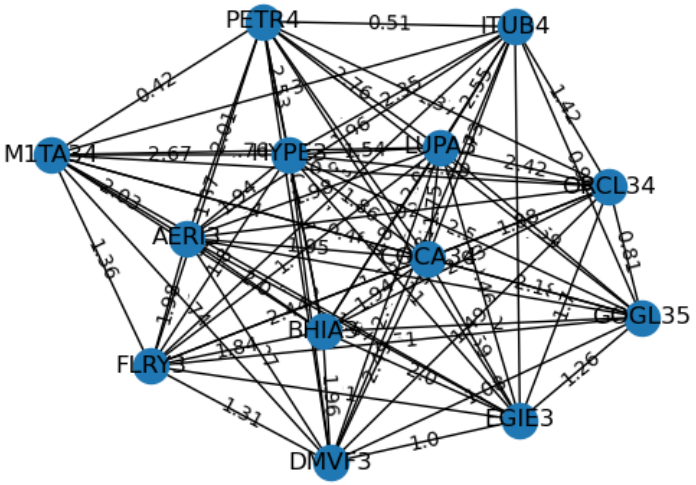

Figure 3: Simplified Problem: Fully connected graph.

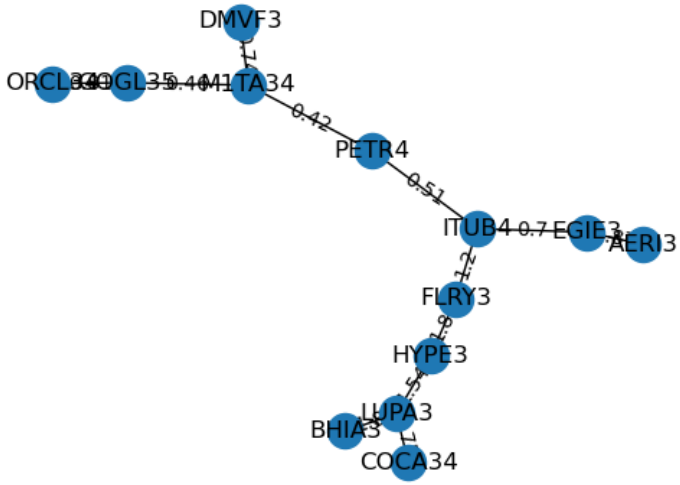

Figure 4: Simplified Problem: Minimum Spanning Tree.

Why we need embeddings?

- Graphs live in non-Euclidean spaces:
  - This makes it difficult to apply common ML techniques, such as k-means clustering.
  - These algorithms rely on Euclidean distances to measure distances between points.
- In graphs, we lack such straightforward Euclidean distances.

Mapping Networks into Euclidean space

- **Network embedding** projects graph elements (nodes, edges, subgraphs) into a lower-dimensional space. [6]
- **Objective**: Preserve graph properties in the embedding.
- **Challenges:**
  - There are many ways to embed nodes.
  - Graphs live on non-Euclidean spaces, so any mapping into Euclidean space loses some information.

- Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges:
- **Node embedding** learns a mapping function:

$$f : v_i \rightarrow \mathbb{R}^d$$

- The function encodes each node v_i into a low-dimensional vector of dimension d, where $d \ll |V|$.
- **Objective**: Preserve similarities between nodes in the embedding space.

Random Walk Embeddings

- DeepWalk
  - Introduced by Perozzi et al. (2014) [8]
  - Embeds network nodes by generating sequences (similar to sentences in natural language) through fixed-length random walks.
- Node2Vec
  - Proposed by Grover and Leskovec (2016) [3]
  - Extends DeepWalk with biased random walks:
    - Allows tuning between Breadth-First Search (BFS) for capturing local neighborhood structure and Depth-First Search (DFS) for global network exploration.
    - Offers greater flexibility in embedding generation.

Types of Embedding Evaluation

- Embedding functions can be evaluated using two main approaches [7]:
  - **Theoretical analysis**: Focuses on mathematical properties and guarantees.
  - **Empirical analysis**: Examines performance in practical applications.

Empirical Evaluation

- Empirical evaluation varies based on its application context [7], such as:
  - Pre-training
  - Regularization
  - Dimensionality reduction
  - Auxiliary tasks

Pre-training Evaluation

- In this analysis, embeddings are evaluated in a pre-training context, treating embedding functions as hyperparameters for downstream tasks.
- The evaluation process involves:
  1. Training the downstream model using different embedding functions individually.
  2. Comparing performance metrics for the downstream task, such as validation accuracy.

Clustering Task

- **Objective**: Evaluate whether embeddings preserve meaningful relationships between stocks.
  - Main requirement: Highly correlated stocks should cluster together, preserving distances in the embedding space.

$$\text{Loss} = \frac{1}{2N} \sum_{i}^{N} \sum_{j}^{N} \mathbb{1}(\rho_{ij} > \rho_{\text{min}}) \mathbb{1}(C_i \cap C_j = \emptyset)$$

  - Terms:
    - C_i, C_j: the cluster set to which each stock is assigned
    - ρ_{ij}: correlation value for the pair (i, j)

Results

Number of Clusters	DeepWalk (% Loss)	Node2vec (% Loss)
3	9.85	9.82
4	10.89	10.86
5	11.54	11.69
6	12.23	11.78

Table 1: Percentage of nodes with correlation over 0.5 that were clustered in separate clusters

Results Part 2

Number of Clusters	DeepWalk (% Loss)	Node2vec (% Loss)
3	0.88	0.85
4	0.92	0.89
5	1.00	0.97
6	1.06	1.00

Table 2: Percentage of nodes with correlation over 0.8 that were clustered in separate clusters

Results Part 3

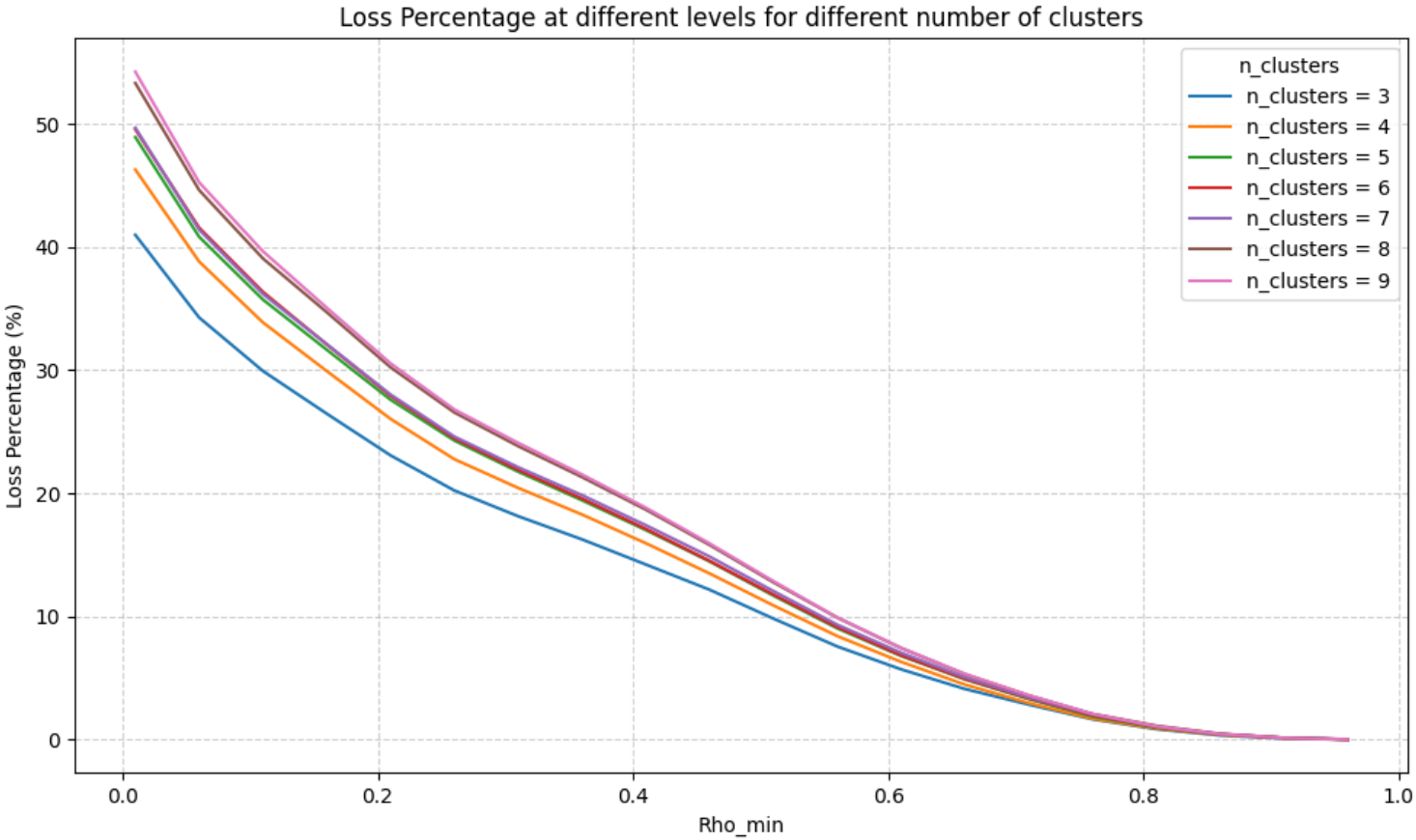

Figure 5: Loss percentage for different number of clusters using Node2Vec

Discussion

- DeepWalk and Node2vec performance is comparable for the different number of clusters.
- For strongly correlated stock, both embeddings are able to able to capture the correlation.

Conclusion

- Shallow embedding methods like DeepWalk and Node2vec effectively capture strong correlations in low-dimensional representations.
- Next steps include training a deep embedding model leveraging state-of-the-art Graph Neural Networks algorithms to evaluate their ability to capture mid-level features.
- Additionally, exploring other downstream tasks beyond clustering—such as classification or link prediction—will provide a more comprehensive evaluation of the embeddings' performance.

References I

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, pages 1655–1658, New York, NY, USA, 2018. Association for Computing Machinery.

MohammadAmin Fazli, Parsa Alian, Ali Owfi, and Erfan Loghmani.
Rps: Portfolio asset selection using graph based representation learning.
Intelligent Systems with Applications, 22:200348, 2024.

References II

Aditya Grover and Jure Leskovec.
node2vec: Scalable feature learning for networks.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

Brindha Priyadarshini Jeyaraman, Bing Tian Dai, and Yuan Fang.
Temporal relational graph convolutional network approach to financial performance prediction.
Machine Learning and Knowledge Extraction, 6(4):2303–2320, 2024.

References III


Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I. Webb, Irwin King, and Shirui Pan.
A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12):10466–10485, 2024.


Shima Khoshraftar and Aijun An.
A survey on graph representation learning methods.
ACM Trans. Intell. Syst. Technol., 15(1), January 2024.


Kento Nozawa and Issei Sato.
Evaluation methods for representation learning: A survey.
In *IJCAI*, pages 5556–5563, 2022.

References IV


Bryan Perozzi, Rami Al-Rfou, and Steven Skiena.
Deepwalk: online learning of social representations.
In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. Association for Computing Machinery.


Bernardete Ribeiro, Ning Chen, and Alexander Kovacec.
Shaping graph pattern mining for financial risk.
Neurocomputing, 326-327:123–131, 2019.


Martin Winistörfer and Ivan Zhdankin.
Measure of dependence for financial time-series.
SSRN Electron. J., 2023.

References V


Junran Wu, Ke Xu, Xueyuan Chen, Shangzhe Li, and Jichang Zhao.
Price graphs: Utilizing the structural information of financial time series for stock prediction.
Information Sciences, 588:405–424, 2022.


Huize Xu, Yuhang Zhang, and Yaoqun Xu.
Promoting financial market development-financial stock classification using graph convolutional neural networks.
IEEE Access, 11:49289–49299, 2023.


Ke Xu, You Wu, Haohao Xia, Ningjing Sang, and Bingxing Wang.
Graph neural networks in financial markets: Modeling volatility and assessing value-at-risk.
Journal of Computer Technology and Software, 1(2), Apr. 2022.