Investigation of the Latent Space of Stock Market Patterns with Genetic Programming

Sungjoo Ha
Seoul National University
Seoul, Republic of Korea
shurain@soar.snu.ac.kr

Sangyeop Lee
Seoul National University
Seoul, Republic of Korea
leesy714@soar.snu.ac.kr

Byung-Ro Moon
Seoul National University
Seoul, Republic of Korea
moon@snu.ac.kr

ABSTRACT

We suggest a use of genetic programming for transformation from a vector space to an understandable graph representation, which is part of a project to inspect the latent space in matrix factorization. Given a relation matrix, we can apply standard techniques such as non-negative matrix factorization to extract low dimensional latent space in vector representation. While the vector representation of the latent space is useful, it is not intuitive and hard to interpret. The transformation with the help of genetic programming allows us to better understand the underlying latent structure. Applying the method in the context of a stock market, we show that it is possible to recover the tree representation of technical patterns from a relation matrix. Leveraging the properties of the vector representations, we are able to find patterns that correspond to cluster centers of technical patterns. We further investigate the geometry of the latent space.

CCS CONCEPTS

• **Computing methodologies** → **Non-negative matrix factorization**; **Genetic programming**;

KEYWORDS

matrix factorization; latent space models; technical patterns; genetic programming;

ACM Reference Format:
Sungjoo Ha, Sangyeop Lee, and Byung-Ro Moon. 2018. Investigation of the Latent Space of Stock Market Patterns with Genetic Programming. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205493

1 INTRODUCTION

Dealing with the relationship between different types of objects is an interesting area of research. In the context of blog subscription, people subscribe to different sites, creating a relationship between them. When dealing with consumers in online markets, understanding the preferences of users is a crucial aspect of the enterprise. Usually, we can infer such preferences from the relationship between users and items. In stock markets, we are interested in finding attractive patterns and further understanding the relationship between patterns and specific stocks. Such relationship often has underlying structures that govern the collective behavior of the objects which, unfortunately, is seldom observable.

One well-known approach to finding the latent structure in a given relationship is to express the relationship using a matrix and applying matrix factorization techniques. In collaborative filtering problems, low-rank matrix factorization approaches have been applied successfully. Given the ratings for movies by users, a relation matrix between users and movies can be constructed. By decomposing the matrix into two smaller matrices, we can extract the latent representation of users and movies. Given the latent representation, we can predict the unobserved relationship between a user and an item. The rationale is that if users' preferences and movies' characteristics share some common qualities among them, we can assume that they will be close to one another in the latent space and this adjacency helps predict the unobserved behaviors. One problem with the matrix factorization approach is that it is hard to understand the latent representation. We can speculate that the latent space captures differing aspects of the relationship. Nevertheless, without further investigation of the latent space, it remains a speculation.

One way to address the problem of interpretability is to introduce side information. Frequently, we have additional information apart from the relationship given as a matrix. For example, in movie recommendation, we usually know the characteristics of a movie such as the genre of the movie, the list of starring actors, the basic plot of the movie, etc. A viable approach is to make use of the side information to build a classifier that takes them as inputs and predicts the relationship. We can apply this approach to study the latent structure found by matrix factorization models.

In this paper, we use side information and genetic programming (GP) to transform the latent representation into an interpretable representation. Specifically, we transform the vector representation obtained from low-rank matrix factorization to corresponding tree representation. We study the relationship between technical patterns and ⟨stock, date⟩ pairs in the context of stock markets. Given black box technical patterns, we can create a relation matrix between patterns and ⟨stock, date⟩ pairs as a binary matrix. Once posed as a relational data it is straightforward to extract and analyze the latent structure by applying the proposed method. Using various technical indicators as side information for the given relationship, we transform the latent vector representations of black box patterns into pattern trees.

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205493

Key aspects of the paper are as follows:

- We pose the problem of analyzing black box patterns of stock markets using the matrix factorization model. By applying our proposed method, latent space representations induced by the factorization of a relation matrix is transformed into tree representations (Section 3).
- We show that the tree representation corresponding to latent space vectors can shed some light on what the matrix factorization method does. This approach has an additional benefit of yielding tree representations that allow us to make decisions on previously unseen data (Section 4).

2 BACKGROUND

Sometimes it is helpful to assume that observations are correlated because of some underlying structures that cause them. We model this by introducing hidden, or latent, variables that are unobserved. A problem that is particularly interesting is finding the latent structure when dealing with relational data. A famous example of latent structure approach to relational data is the winning entry for Netflix challenge [2]. In Netflix challenge, one is asked to predict the unobserved relationship between users and movies using the observed relation matrix. The winning entry for the challenge used an ensemble of matrix factorization models, neighborhood models, and restricted Boltzmann machines [9][20]. Inspecting the latent space gives us insights into how matrix factorization models work. Koren *et al.* [10] examine the first few most important dimensions of the latent space. By exploring the space, they were able to identify that the first axis has, on one side, "lowbrow comedies and horror movies" and, on another side, "drama and comedy with serious undertones and strong female leads."

Instead of relying on the latent structure of the relation matrix, one can take an advantage of attributes or characteristics of entities. In recommender systems community, this approach is known as content-based recommendations. Content-based approach to recommendation has several benefits such as avoiding the cold-start problem, and ability to recommend to users with unique tastes. One important aspect of the content-based approach is the capability to give an explanation of recommended items using the attributes of the item. While there are many benefits of the content-based approach, there are also negative sides to it as well. One problem is the hardness of exploiting other users' preferences. There are researches that combine both approaches. Xu *et al.* [22] explored social networks using infinite hidden relational models where they predicted entity attributes and relationships between entities. By combining user/movie attributes and ratings in MovieLens data, they outperformed the traditional low-rank matrix factorization models.

Evolutionary computation has been applied to collaborative filtering problems [3]. Fong *et al.* [6] directly encoded different features such as demographic attributes and movie attributes into a chromosome and used genetic algorithms to search for suitable recommendations. Guimarães *et al.* [7] proposed a framework called GUARD, that uses GP to create ranking functions taking multiple measures for recommendations into account. Anand [1] used GP to convert user-item relation matrix to a user-feature matrix by learning functions which map ratings for subsets of items to individual features.

The use of NMF together with evolutionary computation has been attracting attention lately. Liskowski and Krawiec [15] has used NMF to derive multiple objectives for solving problems with GP. Similar approach was also applied to coevolutionary settings by Liskowski and Jaśkowski [14] where they compute only a fraction of interactions and utilized NMF to impute the missing values for the problem of learning position evaluation in Othello.

Mining interesting patterns from stock markets has been a focus of research for some time [18][13][8]. For example, Lee and Moon [12] applied GP for mining attractive technical patterns using a modular GP. While explicitly mining patterns using side information has been a usual approach, there are also researches that exploit the latent structure induced by matrix factorization [16][23]. Drakakis *et al.* [5] looked into the relationship between closing prices of Dow Jones stocks on a particular day and applied the non-negative matrix factorization (NMF) model to extract the latent structure. Upon examining the latent representation of stocks, clusters roughly corresponding to market sectors were found. Wong *et al.* [21] predicted the directional movement of stock prices by aligning the latent space induced by multiple stock prices and news articles. Another latent variable approach is illustrated in Doyle and Elkan [4] where they applied topic models to financial data.

3 PROBLEM STATEMENT

Given a relation matrix, we perform matrix factorization to uncover the latent space representation of patterns. We evolve GP trees to mimic the behavior of certain patterns using side information. Once we have the tree representation corresponding to the latent space representation, we can better interpret the latent space. The resulting tree representation essentially acts as a classifier which we can employ to classify previously unseen data. Figure 1 depicts the procedure as a whole.

3.1 Matrix Factorization

Given a bipartite relational graph, we create a matrix whose entries indicate a relationship between two groups. We consider the relation matrix of technical patterns and ⟨stock, date⟩ pairs. Here we refer to a technical pattern as a black box classifier that yields a Boolean value for a certain stock on a certain day. This Boolean value is to be understood as whether the specific ⟨stock, date⟩ pair matches the criteria of the technical pattern or not. If we are given such technical patterns, we can match them against the data at hand which creates a binary matrix where the rows represent the patterns and the columns represent the ⟨stock, date⟩ tuples. Thus, a decomposition of the match occurrence matrix yields matrices $\mathbf{W}^T$ whose column vectors correspond to the latent representation of patterns, and $\mathbf{H}$ whose column vectors correspond to the latent representation of particular ⟨stock, date⟩ tuples.

There is an abundance of algorithms to factorize a matrix each having different characteristics. Here we use non-negative matrix factorization [11] to decompose the given relation matrix. NMF considers the following problem: Given a non-negative matrix $\mathbf{V} \in$

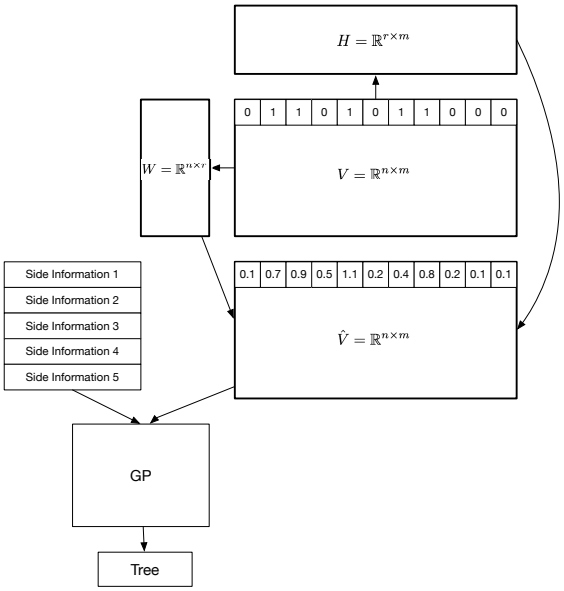

Figure 1: Bird's-eye view of the process. Given a relation matrix between patterns and ⟨stock, date⟩, matrix factorization is performed to extract a latent representation of patterns. An arbitrary point in the latent pattern space can be converted to a row of $\hat{V}$ which corresponds to the behavior of a hypothetical pattern. We can recover a tree representation of the pattern using side information and GP.

$\mathbb{R}^{n \times m}$, decompose $\mathbf{V}$ into a product of two non-negative matrices,

$$\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times m}$

Usually, r is chosen to be smaller than n or m so the resulting matrices $\mathbf{W}$ and $\mathbf{H}$ are smaller than the original matrix $\mathbf{V}$.

For the problem we are considering, the matrices $\mathbf{W}$ and $\mathbf{H}$ correspond to the latent representation of patterns and ⟨stock, date⟩ tuples respectively. If we further decompose the matrix $\mathbf{W}$ as a collection of row vectors, then the i-th row $\mathbf{W}^{(i)}$ corresponds to the latent representation of the i-th pattern. Similarly, the decomposition of $\mathbf{H}$ into column vectors gives us the latent representation of each ⟨stock, date⟩ tuple where the j-th column $\mathbf{H}^{(j)}$ represents the latent representation of the j-th tuple.

The choice of a specific matrix factorization model is arbitrary. One can choose whatever factorization method that seems appropriate for the given relationship. Here, we chose NMF since the occurrence of a pattern match only produces non-negative values. We used Nimfa [24], a Python library of NMF to perform the factorization.

Name	Notation
Opening Price	$p_o(t)$
Closing Price	$p_c(t)$
Highest Price	$p_h(t)$
Lowest Price	$p_l(t)$
n-day Highest Price	$HIGH_n(t)$
n-day Lowest Price	$LOW_n(t)$
n-day Moving Average	$MA_n(t)$
Upper Bollinger Band	$BOL_u(t)$
Lower Bollinger Band	$BOL_l(t)$

Table 1: The set of functions used for the experiment. These functions form the terminal nodes of the GP tree expression, together with various constants.

3.2 Side Information

We often have additional information, apart from the relation matrix, that can be utilized. For patterns in stock markets, there are various elementary functions such as opening and closing prices, and moving average that can be used as side information. The list of side information used in this work is provided in Table 1.

Note that the side information, in this context, describes the property of the ⟨stock, date⟩ tuple. For example, $MA_n(t)$ is the n-day moving average of a particular stock on a particular day. It is a property of the tuple, not of the pattern that matches against it. We are assuming that any black box pattern will take advantage of the latent structure that is shared among ⟨stock, date⟩ tuples. By applying matrix factorization, such a latent structure is revealed as a collection of r-dimensional real vectors. It is our observation that we can explain the latent space using the side information which is much more interpretable.

3.3 Genetic Programming

Given the side information previously stated, we can create an expression tree using the side information. Basic functions can be combined with constants or other functions using arithmetic operators and logical operators. In the case of stock market side information, we can specify the time t for the relative shift from the current day in question. $MA_{10}(-5) > LOW_5(-4) \wedge p_o(-3) < p_c(-1)$ is an example of an expression tree created by such a process. The expression means that the 10-day moving average of five days earlier was greater than the 5-day lowest price of four days earlier and the opening price of three days earlier is less than the closing price of yesterday. We can grow arbitrarily complex expression trees using GP.

For this experiment, we use a total of 100 individuals to form a population. A steady state GP is used to evolve trees each corresponding to the latent representation of a technical pattern. The initial population is created to have random trees of depth two or three. Crossover swaps random subtrees from two parents with 50% probability or performs a geometric crossover on Boolean space [17] with 50% probability. Mutation replaces a random subtree with a random tree or performs a geometric mutation on boolean space with equal probability. Local search is performed after the mutation. We search the neighboring trees where a neighbor is defined to be a

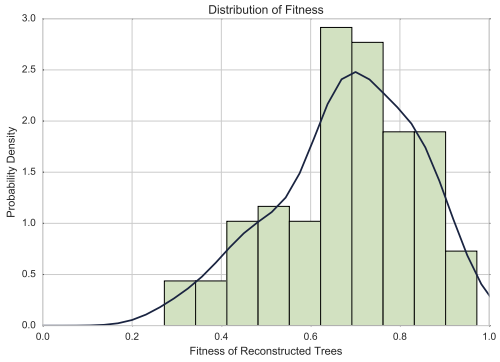

Figure 2: Fitness distribution of reconstructed patterns.

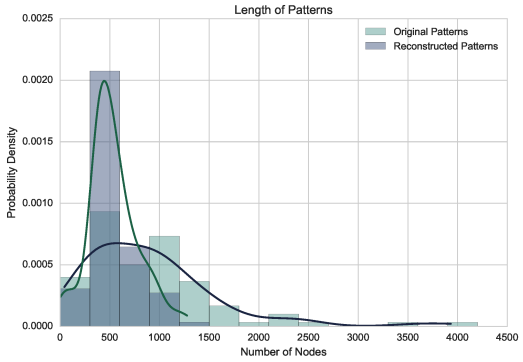

Figure 3: Comparison of the lengths of original patterns and that of reconstructed patterns.

tree of the same structure with differing constants. While traversing each node, constants related to the node is changed in pursuit of finding a local optimum. For the fitness of a tree, we use the Jaccard distance between the match occurrences of the target pattern and that of the generated tree. The Jaccard distance is defined to be $1 - J(A, B)$ where $J(A, B)$ is the Jaccard similarity between two sets, defined as follows:

$$J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{2}$$

We execute GP for 2,500 generations before termination.

4 EXPERIMENTAL RESULTS

Experiments were conducted on the common stocks from the Korean stock market during the period of 2013 to 2014. A relation matrix was created using 100 black box patterns which are matched against a total of 247,968 ⟨stock, date⟩ tuples. The patterns were generated by evolving trees using GP that aims to find patterns that match frequently and yield high expected earning rate during the same period. We specifically tested our method on these patterns to compare the characteristics of black box classifiers against newly found trees.

Given a relation matrix, we apply NMF to extract the latent space of the relationship. Representations were embedded into the space of 50 dimensions. Once we extracted the matrices $\mathbf{W}$ and $\mathbf{H}$, we checked to see how much of the original matrix the latent representation could recover. After binarizing the matrix $\hat{\mathbf{V}} = \mathbf{WH}$, the Jaccard distance between $\mathbf{V}$ and $\hat{\mathbf{V}}$ was 0.03745. There were a total of 21,302 different entries out of 24,796,800 elements.

4.1 Recovering Original Patterns

We evolve trees to match the behavior of black box patterns. The objective we are pursuing is a corrupted version of the behavior of original patterns. It is corrupted, in a sense, because we are using $\hat{\mathbf{V}}$, instead of $\mathbf{V}$, to compute the fitness of a tree. Figure 2 illustrates the resulting fitness distribution. The mean Jaccard similarity is 0.683.

A close inspection reveals that there are some qualitative differences between original pattern trees and newly found ones. The distribution of the tree lengths is shown in Figure 3. The tree length is defined as the number of nodes used to create the expression

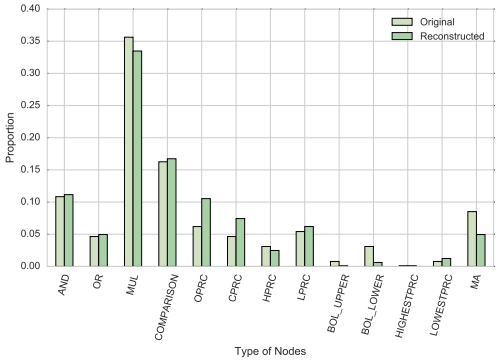

Figure 4: Comparison of an original pattern and a reconstructed pattern.

tree including constants, operators, and functions. We see that the original trees are relatively larger than the newly constructed ones. Not only that, the original trees are more uniformly distributed in terms of their length compared to the newly found trees.

Figure 4 compares a typical case of two pattern trees, one original and the other reconstructed, with regard to the proportion of the constituent nodes. The specific trees are of similar length where the original pattern has 448 nodes and the reconstructed one has 582 nodes. The fitness of the reconstructed tree is 0.68. Observe that their composition is considerably different. While the original pattern focuses on moving averages and Bollinger bands, the reconstruction seems to prefer opening prices and closing prices to achieve similar results.

We tested both the original patterns and the reconstructed patterns on previously unseen data. If the reconstruction was able to capture the latent structure, it should lead to similar results produced by the original patterns. Figure 5 shows the Jaccard similarities between the match results of the original pattern trees and the reconstructed pattern trees. We see that many of the reconstructions were able to mimic the behavior of the original patterns on unseen data. Notice that the shape of the distribution of the Jaccard

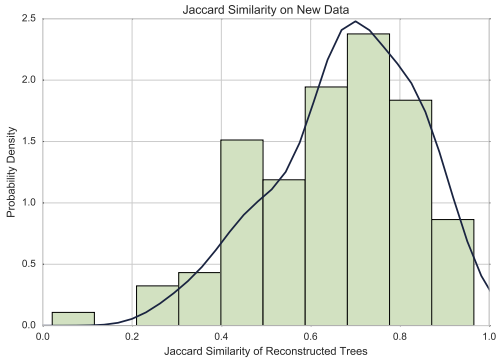

Figure 5: Jaccard similarity on new data.

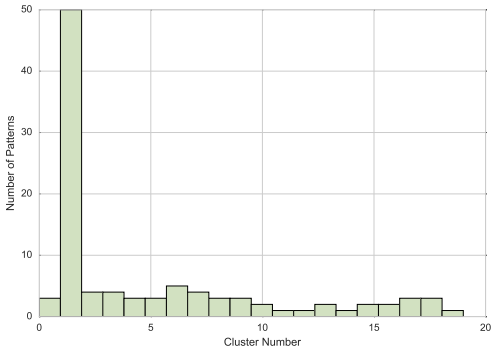

Figure 7: Result of k-means clustering.

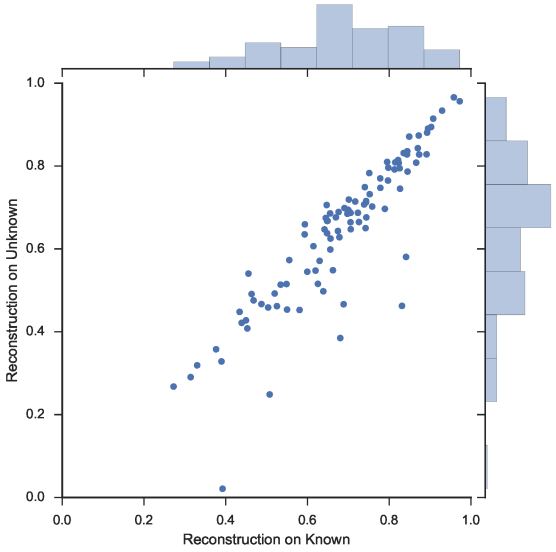

Figure 6: Comparison of reproducibility and generalizability. Reproducing the behavior of a pattern on known data correlates with the behavior of the same pattern on unknown data.

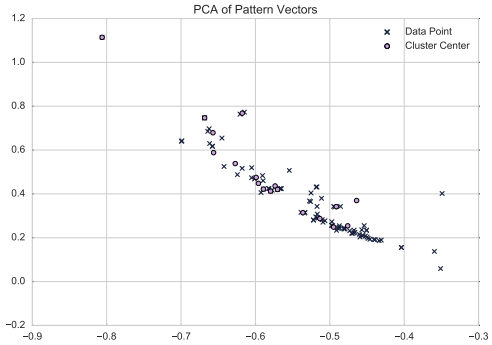

Figure 8: Visualization of pattern vectors after PCA. Crosses correspond to each pattern vector and dots correspond to cluster centers. Cluster center at the lower right corner corresponds to cluster one. Seven outliers were omitted to create a discernible plot.

similarities on new data resembles that of the fitness distribution in Figure 2.

We further looked at the relationship between reproducibility and generalizability. Figure 6 depicts the relationship using a joint plot. There is a highly linear relationship between the reproducibility and the generalizability. If we can reconstruct a tree that reproduces the behavior of the target black box pattern with high accuracy, the behavior generalizes to unseen data as well.

4.2 Analysis of Latent Space

4.2.1 Clustering. It is worthwhile to investigate whether the reconstructed patterns behave similarly to one another. We performed k-means clustering to the latent space representation of patterns expressed as vectors $\mathbf{W}^{(i)}$ using 20 clusters. The result of clustering in Figure 7. One large cluster indicates that there is a large group of patterns that has similar qualities to one another. To understand the result of the clustering, we performed principal component analysis (PCA) to visualize the pattern vectors. Figure 8 gives us a hint as to why there is one large cluster. Patterns are tightly packed around the lower right region of the plot and many of them are assigned to the cluster center located at the lower right corner. Seven outliers were removed from the visualization to focus on the data that are closely located.

One advantage of dealing with latent space is that we can easily find the center of a cluster. If the relation matrix was produced by true black box classifiers, it is impossible to find the classifier that corresponds to the center of a cluster. Even in the case of white box tree-based patterns, it is still not straightforward how one can find the center of a cluster. In latent space representation, it is easy to find a vector representation that corresponds to the center of a cluster. Once this vector representation is found, we can create the $\hat{V^{(i)}}$ result that corresponds to the behavior of a

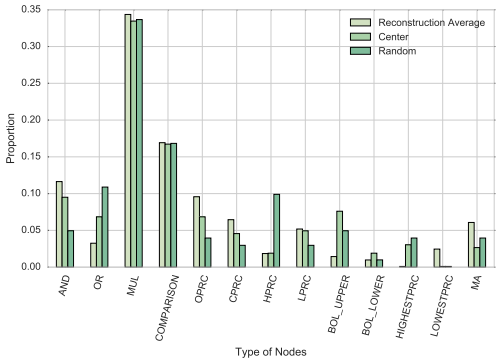

Figure 9: Comparing the average of reconstruction trees belonging to the same cluster with the cluster center and a randomly chosen pattern tree. For this particular case, the KL divergence between the reconstruction trees and the cluster center is 0.1595 and the KL divergence between the reconstruction trees and the random tree is 0.2809.

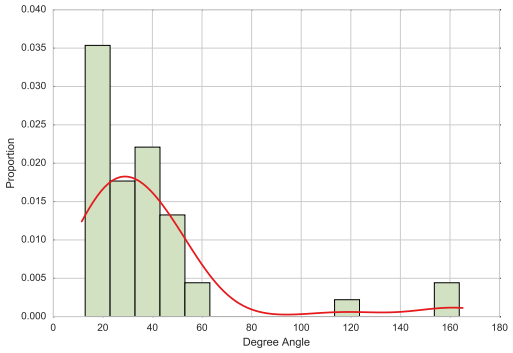

Figure 10: Angle between a vector formed by two parents and a vector formed by a parent and the offspring.

hypothetical cluster-center pattern. From this, we can find the tree representation using our approach.

Figure 9 illustrates a typical case of a cluster center. The nodes comprising a cluster center are fairly similar to the average composition of the trees belonging to the same cluster. The composition of a randomly chosen pattern tree is dissimilar from that of the other two. We can quantify the difference of compositions by computing the KL divergence between the proportion of nodes. For two probability distributions P and Q, the KL divergence is defined as follows:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{3}$$

The average KL divergence between reconstruction trees and the cluster center is 0.1575 and the average KL divergence between reconstruction trees and a random tree is 0.2178.

4.2.2 Orthogonal Basis. We further investigate the latent space by decomposing each axis of the latent space. Since a pattern is represented by a linear combination of axes, examining vectors corresponding to each axis offers insights of the topology of the latent space. A vector corresponding to a standard basis is used to create a $\hat{V^{(i)}}$ result that is assumed to capture the behavior of a standard basis. Instead of using a unit vector, we stretch the vector to make sure the length of the vector is similar to that of other pattern vectors. After creating tree representations corresponding to such vectors, we examine their contents.

Table 2 exhibits some of the trees corresponding to standard bases. We see that each axis captures information of differing qualities. For example, the axis 12 heavily exploits moving averages while the axis 18 does not use moving average at all.

A broader look at the composition of trees hints at the relative importance of different node types. Table 3 reveals the number of times a node type appears in a tree corresponding to an axis. Since there are 50 axes, a node type that occurs 50 times is present in all of the axis trees. One can tell that raw price information is used more often than abstract information such as Bollinger bands and n-day highest prices. While this does not mean that such information is less useful, it does seem to indicate that they are more specialized in some sense.

4.2.3 Geometric Operation. The match occurrences of a black box pattern $\mathbf{V}^{(i)}$ can be thought of as the semantic of the pattern. Matrix factorization transforms the semantic space into two separate latent spaces, each corresponding to pattern and ⟨stock, date⟩ tuple. Since we have separated the contribution of patterns and ⟨stock, date⟩ tuples, the geometry of latent pattern space is different from that of the semantic space. We examined the effect of geometric operations on the semantic space in the latent vector space.

We picked two original pattern trees and performed geometric semantic crossover defined as follows:

$$T_o = (T_1 \wedge T_r) \vee (T_2 \wedge \neg T_r) \tag{4}$$

where T_1 and T_2 are trees corresponding to parent patterns and T_r is a random tree of depth three to six. The resulting offspring T_o was matched against the given ⟨stock, date⟩ tuples yielding $\mathbf{V}^{(k)}$. We found the corresponding latent vector representation $w_o = \text{argmin}_{\mathbf{w}}(w \cdot \mathbf{H}) - \mathbf{V}^{(k)}$ by gradient descent.

Given latent vector representation of two parents w_{p1} and w_{p2}, we can create a hypersphere whose center is $w_m = \frac{w_{p1}+w_{p2}}{2}$ and the diameter is defined by $|w_{p1} - w_{p2}|_2$. 80% of the time, the offspring w_o was inside the hypersphere. The mean distance between w_o and w_p was 12.51 and it was 9.13 for w_o and w_m. The angle between $w_{p2} - w_{p1}$ and $w_o - w_{p1}$ is compared in Figure 10. The mean angle is 45°. While the latent pattern space does not completely preserve the geometry of the semantic space, they are not drastically different.

5 CONCLUSIONS

We proposed a method of inspecting the latent space induced by a low-rank matrix factorization for a relation matrix. We have experimented the method on the Korean stock market to see if we can make sense of the latent space of technical patterns. Once described as a matrix factorization problem, it was straightforward

Axis	$\wedge$	$\vee$	$\times$	$>$	$<$	p_o	p_c	p_h	p_l	BOL_u	BOL_l	$HIGH$	LOW	MA
0	11	8	40	9	11	5	2	4	6	7	5	5	2	4
1	15	15	62	15	16	6	10	8	13	0	6	9	0	10
2	5	10	32	9	7	0	3	10	3	1	1	4	1	9
3	21	9	66	13	18	12	8	2	7	9	4	4	7	9
4	1	1	6	3	0	0	1	0	1	1	1	0	0	2
5	4	16	44	2	19	6	3	6	4	0	2	2	0	19
6	10	6	34	9	8	1	2	8	6	1	9	2	4	1
7	16	0	34	0	17	0	8	8	3	0	4	3	4	4
8	15	0	32	11	5	2	3	14	0	4	0	1	1	7
9	25	8	74	10	24	5	1	7	12	10	0	0	17	16
10	24	1	46	13	13	4	3	11	9	6	6	2	10	1
11	19	0	41	11	9	2	2	9	3	5	5	2	7	5
12	73	15	178	25	64	19	14	34	32	10	14	0	23	32
13	16	6	46	9	14	9	0	3	11	2	6	7	6	2
14	26	2	58	19	10	8	12	2	6	2	8	0	3	17
15	13	5	38	9	10	5	2	5	7	3	8	2	4	2
16	15	9	54	13	12	5	1	4	17	4	14	4	0	1
17	16	8	46	12	13	7	0	11	6	5	4	2	9	6
18	19	1	42	12	9	3	14	9	2	2	4	4	4	0
19	8	6	30	4	11	5	2	4	4	4	4	1	4	2
20	22	4	54	16	11	2	4	9	10	8	7	3	0	11
21	10	15	52	8	18	3	3	7	7	9	10	7	4	2
22	39	8	103	24	24	7	20	20	19	7	0	3	0	20
23	32	0	66	3	30	1	2	13	8	0	3	18	12	9
24	51	11	127	49	14	11	6	28	9	17	0	20	24	11

Table 2: Composition of trees corresponding to standard bases.

Node Type	Count
$\wedge$	49
$\vee$	44
$\times$	50
$>$	49
$<$	49
p_o	43
p_c	47
p_h	49
p_l	48
BOL_u	41
BOL_l	40
$HIGH$	39
LOW	39
MA	44

Table 3: The number of times a node appears in a tree corresponding to standard basis.

to examine the nature of the black box classifiers that correspond to technical patterns. The relationship between reproducibility and generalizability tells us that if we can reproduce the result with high accuracy, we can rely on the reconstructed tree to interpret the black box algorithm that created the relationship.

It is noteworthy to point out that the method is not specific to stock markets. When we have a relation matrix and appropriate side information, we can apply the method to inspect the latent space and make use of the resulting trees. Such trees not only bring us insights about the latent structure, they can also be used to make decisions since these trees act as classifiers.

Future work includes applying the framework on other domains such as collaborative filtering and social network connections. We also wish to address the problem of ignoring the time dependencies between ⟨stock, date⟩ pairs. There are more sophisticated models that explicitly model such dependencies using context-aware matrix factorization methods including factorization machines [19] and tensor factorization.

REFERENCES

[1] D Anand. 2012. Feature Extraction for Collaborative Filtering: A Genetic Programming Approach. *International Journal of Computer Science Issues* 9, 5 (2012), 348–354.

[2] James Bennett and Stan Lanning. 2007. The Netflix Prize. In *KDD-Cup and Workshop at International Conference on Knowledge Discovery and Data Mining*.

[3] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Javier Alcalá. 2011. Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-Based Systems* 24, 8 (2011), 1310–1316.

[4] Gabriel Doyle and Charles Elkan. 2009. Financial Topic Models. In *NIPS Workshop on Applications for Topic Models: Text and Beyond*.

[5] Konstantinos Drakakis, Scott Rickard, Ruairí de Fréin, and Andrzej Cichocki. 2008. Analysis of financial data using non-negative matrix factorisation. *International Mathematical Forum* 3, 38 (2008), 1853–1870.

[6] Simon Fong, Yvonne Ho, and Yang Hang. 2008. Using Genetic Algorithm for Hybrid Modes of Collaborative Filtering in Online Recommenders. In *International Conference on Hybrid Intelligent Systems*. 174–179.

[7] Adolfo Guimarães, Thales F Costa, Anisio Lacerda, Gisele L Pappa, and Nivio Ziviani. 2013. GUARD: A Genetic Unified Approach for Recommendation. *Journal of Information and Data Management* 4, 3 (2013), 295–310.

[8] Sungjoo Ha and Byung Ro Moon. 2015. Fast Knowledge Discovery in Time Series with GPGPU on Genetic Programming.. In *Genetic and Evolutionary Computation Conference*. 1159–1166.

[9] Yehuda Koren. 2009. *The bellkor solution to the netflix grand prize*. Technical Report.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[11] Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*. 556–562.

[12] Seung-Kyu Lee and Byung Ro Moon. 2010. A new modular genetic programming for finding attractive technical patterns in stock markets. In *Genetic and Evolutionary Computation Conference*. 1219–1226.

[13] Piotr Lipinski. 2007. ECGA vs. BOA in discovering stock market trading experts.. In *Genetic and Evolutionary Computation Conference* (2007-08-21). 531–538.

[14] PawełLiskowski and Wojciech Jaśkowski. 2017. Accelerating Coevolution with Adaptive Matrix Factorization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*. ACM, New York, NY, USA, 457–464. https://doi.org/10.1145/3071178.3071320

[15] PawełLiskowski and Krzysztof Krawiec. 2016. Non-negative Matrix Factorization for Unsupervised Derivation of Search Objectives in Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, New York, NY, USA, 749–756. https://doi.org/10.1145/2908812.2908888

[16] Tang Liu. 2009. Non-Negative Matrix Factorization for Stock Market Pricing. In *International Conference on Biomedical Engineering and Informatics*. 1–5.

[17] Alberto Moraglio, Krzysztof Krawiec, and Colin G Johnson. 2012. Geometric Semantic Genetic Programming. In *Parallel Problem Solving from Nature*. 21–31.

[18] Jean-Yves Potvin, Patrick Soriano, and Maxime Vallée. 2004. Generating Trading Rules on the Stock Markets with Genetic Programming. *Computers and Operations Research* 31, 7 (June 2004), 1033–1047.

[19] Steffen Rendle. 2010. Factorization Machines. In *International Conference on Data Mining*. 995–1000.

[20] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *International Conference on Machine Learning*. 791–798.

[21] Felix Ming Fai Wong, Zhenming Liu, and Mung Chiang. 2014. Stock Market Prediction from WSJ: Text Mining via Sparse Matrix Factorization. In *International Conference on Data Mining*. 430–439.

[22] Zhao Xu, Volker Tresp, Achim Rettinger, and Kristian Kersting. 2010. Social Network Mining with Nonparametric Relational Models. In *Advances in Social Network Mining and Analysis*. Vol. 5498. 77–96.

[23] Zhong-Yuan Zhang. 2012. Nonnegative Matrix Factorization: Models, Algorithms and Applications. In *Data Mining: Foundations and Intelligent Paradigms*. Vol. 24. 99–134.

[24] Marinka Žitnik and Blaž Zupan. 2012. Nimfa: A Python Library for Nonnegative Matrix Factorization. *Journal of Machine Learning Research* 13 (2012), 849–853.