공학석사학위논문

Financial Time Series Clustering: Obtaining Long-Term Trends with Deep Embedding Network

장기적 추세를 반영한 심층 임베딩 기반 금융 시계열 군집화에 관한 연구

2019 년 8 월

서울대학교 대학원

산업공학과

고 형 진

Financial Time Series Clustering: Obtaining Long-Term Trends with Deep Embedding Network

장기적 추세를 반영한 심층 임베딩 기반 금융 시계열 군집화에 관한 연구

지도교수 이 재 욱

이 논문을 공학석사 학위논문으로 제출함

2019 년 06 월

서울대학교 대학원

산업공학과

고 형 진

고형진의 공학석사 학위논문을 인준함

2019 년 06 월

위 원 장	박 종 헌	(인)
부위원장	이 재 욱	(인)
위 원	장 우 진	(인)

Abstract

Financial Time Series Clustering: Obtaining Long-Term Trends with Deep Embedding Network

Hyungjin Ko
Department of Industrial Engineering
The Graduate School
Seoul National University

In the field of asset selection and portfolio, there are active researches on clustering for various reasons. In recent years, there have been increasing cases of applying machine learning and deep learning methodology to asset clustering studies. This is because it is difficult to reflect insights such as long-term trends and patterns reflected in high-dimensional image data by traditional correlation-based analysis. Therefore, this thesis investigated how to clustering financial time series through deep embedding network that is specialized for processing high-dimensional data efficiently. It is shown that the existing algorithm is not suitable for the financial time series data, and proposed algorithm can perform the clustering better than the existing algorithm. In addition, we have clustered KOSPI data with the proposed algorithm and determined the optimal number of clusters through various performance measures. We also examined whether the insights trends inherent in the actual high-dimensional images can be reflected in the clustering results. In addition, based on the results of this thesis, it can be shown that the actual effect of incorporating the results of this study to the portfolio management by comparing the performance measures of various portfolios with the benchmark results, in the future works.

Keywords: Financial time series clustering, Asset clustering, Asset selection, Deep embedding network, Deep learning, Long-term trends

Student Number: 2017-21537

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Since Markowitz's Modern Portfolio Theory in [23], asset selection and asset allocation have become the most important consideration in the investment management. This is because asset selection and allocation can have diversification effect over risk. In other words, it is possible to construct a portfolio with a lower risk under a given level of return. In his research, he proposed a mathematical analysis framework called mean-variance analysis, which uses the expected returns and the variance obtained through historical returns. Since then, there have been many studies on asset selection and allocation based on this theory. When choosing the assets that make up the portfolio and determining how much to allocate to those assets, the effect of risk diversification can be gained by measuring and using the correlation between assets. For that reason, the correlation between assets, that is, the similarity between assets, is a very important measure in asset allocation.

In this respect, many studies have been carried out to utilize the similarity of assets for clustering of assets and to use them for portfolio management. Tola et al. [32] attempted to mitigate the statistical uncertainty inherent in the variance-covariance matrix used as an estimate of risk in the portfolio optimization by using clustering algorithms. Nanda et al. [24] conducted clustering based on various corporate valuation variables such as historical returns and price-to-book ratio using k-means, self-organizing map, and fuzzy c-means. In addition, he used the results to construct efficient portfolio. Leon et al. [18] have shown that using a hierarchical clustering technique in a risk-adjusted portfolio improves performance over the existing mean-variance portfolio approach.

The reasons for attempting clustering in research related to asset selection and asset allocation are as follows. First, for diversification effects for less risk under a given return, assets with different characteristics should be selected and a portfolio constructed. If the portfolio is composed of multiple clusters with small similarities and then the assets are selected from the clusters, diversification effect can be expected [18]. Second, there may be assets with similar characteristics or movements of similar stock prices where these redundancy can be eliminated through clustering. This is important in terms of time efficiency. If the investor has a very large set of base assets to choose from, the number of cases where the asset can be selected is very large, which is inefficient in terms of the time efficiency of constructing the portfolio. Therefore, choosing assets after removing redundancy through asset clustering would be much more efficient in terms of time efficiency [15].

Recently, machine learning and deep learning have been actively studied. In some areas, including image processing, performance already exceeds human performance according to Guo et al. [16]. In addition, as the superior performance of the technology of machine learning and deep learning has been demonstrated, attempts and researches have been actively made to apply to actual financial field [6]. For example, studies on outliers detection and filtering have been actively conducted to solve the problem of outliers [14] that cause performance degradation of various financial models such as data-driven prediction models [9, 21]. A financial time series is a collection of data about time and is nonlinear and dynamical data. Li et al. [19] attempted to analyze the entire time-series data by dividing the financial time series by time and identifying the patterns of the parts. Financial time series prediction is also very important for researchers in the field of machine learning [31]. A variety of machine learning and deep learning methodologies such as Artificial Neural Network [20], Multilayer Neural Network [11], Support Vector Machine [22], and Tree and Ensemble [4] have been used for financial time series prediction models. In particular, clustering is one of the most important areas of unsupervised learning, one of the representative areas of machine learning, and is a methodology directly applied to asset selection studies. D'Urso et al. [12] conducted a study on financial time series clustering using novel distance measure based on GARCH model. Aghabozorgi et al. [1] also researched the clustering of time series, which is called co-movement, showing similar motion patterns regardless of time difference.

1.2 Research Motivation and Contribution

Generally, a person who sees an object in a visual system recognizes time series data as high-dimensional image data rather than single-variable time series numerical data. According to Aigner et al. [2], the purpose of visualizing time series data is to understand the data by sensing the long-term trends and patterns inherent in the data and by recognizing its intuition. In other words, it is possible to reflect the intuition such as long-term trends and patterns by converting the time series into an image which is high-dimensional data instead of low-dimensional numerical data. However, most studies on clustering-based asset selection have focused on clustering based on low-level financial time series data [15, 24]. The use of only simple numerical data of a time series with a low dimension has a disadvantage that the intuition intrinsic to the high dimensional data imaged by the time series is difficult to be reflected in the clustering analysis. In other words, the conventional clustering method uses only low-dimensional simple time-series numerical data, and it is difficult to reflect intuitive relationships between assets represented by high-dimensional image data.

This problem mainly occurs when the correlation coefficient is obtained based on historical returns of each assets which are simple low-dimensional time series, and clustering is performed based on this correlation. If we simply look at correlations based on historical returns, we may not have a compensating effect on the correlation coefficient that depends on the time resolution. In other words, in measuring the similarity of two assets, only micro-movements at a specific time point are reflected but macroscopic movements may not be accurately reflected. Numerical examples for understanding are as follows.

Assume that two assets exist as in Table 1.1. Starting with a price of 100 for the first asset, the price goes up by 2 for even days and the price goes down by -1 for odd days. This price change is repeated, resulting in a final price of 146. For the second asset, the asset price starts at 100, the price drops by -1 for even days, and the price goes up by 2 for odd days. By repeating this price change, the final price becomes 146 as well.

The price of two assets is shown in Figure 1.1. When price of time series data is represented by high-dimensional image data, the correlation between two assets

Date	**Assets**	
	price of asset1	**price of asset2**
start date	100	100
even days	Increased by +2	Decreased by -1
odd days	Decreased by -1	Increased by +2
end date	146	146

Table 1.1: Numerical example of the problem of clustering analysis based on historical return of the low-dimensional time series

seems to be very high in the long term. However, if correlation coefficient is calculated based on historical returns, the result is -0.997, which is the opposite result to our intuition. In other words, high-dimensional image data appears to have a positive correlation in the long term, but negative correlation in the historical return data. There is a possibility that the actual numerical result and the intuition recognized in the high-dimensional image may not match. When the clustering is performed using the existing correlation-based clustering method, the two assets belong to different clusters. It may vary depending on the purpose of the clustering, but it do not seems to be a result consistent with the purpose of clustering in general.

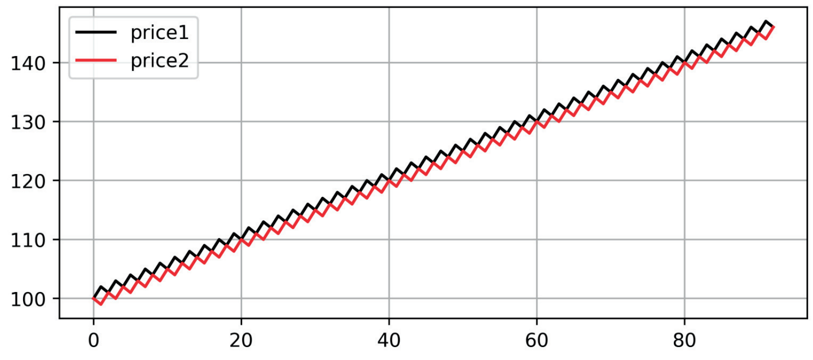

Figure 1.1: Image for two asset prices in the numerical example

Conventional clustering methods have such problems, but attempts to directly solve these problems have not been studied yet. Therefore, in order to solve these problems, it is necessary to study the clustering technique that can process high-dimensional image data. Fortunately, in recent research, a deep embedding clustering technique [34] has been studied that efficiently clusters high-dimensional image data through deep embedding techniques. This shows that it is possible to efficiently cluster the high dimensional image data than other existing clustering models.

Therefore, in this thesis, we will apply the deep embedding technique that can efficiently process the image data, which is high-dimensional data, to the clustering analysis for the stock price of financial time series. However, existing deep embedding techniques are not suitable for financial time series data because they have good performance only for certain data such as MNIST, which is frequently used as experimental data in the field of deep learning. Therefore, in this thesis, we propose a new algorithm that solves this problem by modifying the existing algorithm appropriately.

Clustering analysis was performed as follows. First, clustering is performed on artificially divided data through a specific criterion to see whether the existing deep embedding technique can appropriately cluster the financial time series data. We then proposed a new algorithm by modifying existing algorithm and experimented to see if the proposed algorithm works properly. Finally, through the proposed algorithm, clustering analysis is performed on KOSPI stock price data that is not artificially divided. We will also investigate the number of optimal clusters in various performance measures. We will interpret how the intuitive relationship between assets inherent in high-dimensional images is reflected in the clustering results.

1.3 Organization of the Thesis

The organization of this thesis is as follows. In the Chapter 2, we will discuss the theoretical background of portfolio allocation, clustering, deep learning, deep embedding clustering, Geometric Brownian motion. In Chapter 3, we will describe simulated data and real KOSPI data and explain description of three data sets preprocessed for three experiments. We will also diagnose the problems of existing algorithm and propose new algorithm that solve them. In Chapter 4, we show that the financial time series data can be appropriately clustered through the newly proposed algorithm. We will also analyze the clustering results of the financial time series data and intuitive relationships between assets inherent in high-dimensional image data after determining the optimal number of clusters through clustering performance measures. Finally, we will conclude and present future work in Chapter 5.

Chapter 2

Related Work

2.1 Markowitz's mean-variance Portfolio Theory

Markowitz [23] proposed a mean-variance analysis model to find a portfolio with the lowest risk under a given expected return for appropriate asset allocation to pursue a portfolio with risk diversification effects. This mean-variance portfolio model postulates that investors are risk-averse. In other words, the investor prefers the least risky asset among the assets with the same expected return, and on the contrary, prefer the high expected return at the same risk level. In this case, the expected return is the expectation of the historical returns, and the variance-covariance matrix is calculated using the variance of historical returns and covariance between historical return of each asset. The investor tries to minimize the non-systematic risk by calculating the optimum weight w^* considering the trade-off between maximizing profit and minimizing risk through the expected return and the variance-covariance. This can be expressed by the following optimization problem.

$$\min_{w} \quad w^T\mu - \frac{1}{2}w^T\Sigma w$$

$$\text{s.t.} \quad \sum_i w_i = 1$$

$$w_i \geq 0$$

where μ is the expected return vector of each asset and Σ is the variance-covariance matrix. The optimum weight w^* is obtained by the above optimization problem. The optimal portfolio is expressed as $w^* \cdot \mu$, a linear combination of the optimal weight w^* and the expected return of the assets μ.

2.2 Clustering

Unsupervised learning is studied extensively in the field of machine learning. Supervised learning refers to a machine learning method in which both input and output data are paired as a data sample for learning when a function f that maps the predictive variables X to the target variables Y is found. On the other hand, unsupervised learning is a learning method in which the predictive variable X exists but the target variable Y does not exist. In other words, the probability distribution of the predictive variable X, $P(X)$, is directly deduced without the help of the target variable Y [13].

A representative example of such unsupervised learning is clustering analysis. The clustering analysis is to group the entities according to the common characteristics extracted from the input variable X. That is, the input data is grouped based on homogeneous features, relationships among the elements, and hidden patterns intrinsic in the data. This object can generally be achieved by the following process. First, minimize dispersion within the cluster. Second, maximize the dispersion among the clusters. In other words, clustering is performed so that the cohesion degree between instances becomes higher by making the distance between the data belonging to the same cluster close to each other, and at the same time, the distances between the clusters are made large so that the degree of separation between the clusters is high. In general, the clustering analysis are performed by defining the following distance or dissimilarity measures.

$$D(x_i, y_j), i \neq j$$

If the above distance between instances in the same cluster is small, it means that the similarity between instances is high. Also, the distances among individuals belonging to different clusters are high, which means that there is a low similarity among individuals in different clusters.

2.3 Deep Learning and Researches on Deep Embedding Clustering

2.3.1 Deep Learning

Deep learning, an area of machine learning, is a methodology based on the concept of biological mechanisms by which neural networks in the brain work [28]. In the early research, it was in the form of a simple artificial neural network or perceptron, which is not deep in layers, but it has developed into a deep neural network with a multilayer neural network. It works as a nonlinear function approximator

$\hat{y} = f(X; w)$ through multi-layer neural networks and performs learning for tasks in one of supervised learning, semi-supervised learning, and unsupervised learning. Generally, in the learning process of deep learning, first, each layer composed of neural networks is stacked deeply and hierarchically. The neural network of each layer receives the output value of the previous neural network as the input value, and outputs the linear combination of the input value and the weight, passing through the nonlinear activation function. It is the multilayer neural network that the neural networks of each layer that make these calculations form several layers. The goal of the deep learning is to define the appropriate loss function $L(y, \hat{y})$ and then update the parameter of the neural network w with the learning rate η according to the optimization method such as gradient descent method.

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

By updating the parameters w in the above manner, it is achieved that the optimal parameter w^* makes the loss function have minimum value as follows.

$$w^* = \arg\min_{w} L(y, \hat{f}(X; w))$$

2.3.2 Batch Normalization

In order for the optimization to proceed properly in the deep learning process, the distribution of the activation function output value of each layer should be appropriately distributed so that it does not deviate to one side. This is because the gradient vanishing problem may cause the optimization to no longer be updated. Ioffe et al. [17] solved this problem by forcing the distribution of output values of the activation functions of each layer through batch normalization. The key procedure of batch normalization is as follows. First, the mean μ_B and variance σ_B^2 for a particular batch $B = x_{1,\ldots,m}$ are obtained as follows.

$$
\begin{aligned}
\mu_B &\leftarrow \frac{1}{m}\Sigma_{i=1}^{m} x_i \\
\sigma_B^2 &\leftarrow \frac{1}{m}\Sigma_{i=1}^{m}(x_i - \mu_B)^2
\end{aligned}
$$

where, x_i is k-dimensional input vector. Then, normalization is performed so that the distribution of data becomes appropriate.

$$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

where, ϵ is and small constant for numerical stability. Finally, transformation is performed as follows. At this time, the initial value of γ and β is 1 and 0. As learning is performed, the value is adjusted to an appropriate value.

$$y_i \leftarrow \gamma\hat{x_i} + \beta$$

Batch normalization improves the learning speed and solves the dependency problem caused by the initial parameter value. In addition to this, the overfitting problem can be solved to a great extent.

2.3.3 Deep Auto Encoder

Deep auto encoders are bottleneck-shaped deep neural networks that reduce the dimension of input data and learn its embedding in an unsupervised manner. In the middle network, the representations inherent in the data are encoded in the latent space and for this, the output data similar to the input data are used simultaneously. Deep auto encoders can be applied to many applications such as dimensional reduction, representation learning, embedding extraction, generative model, clustering. The deep autocoder consists of an encoder, which is a function to map the input data on the feature space to the latent space, and a decoder, which is a function that reconstruct the embedding mapped in the latent space to the output data similar to original input data again. The encoder and the decoder are expressed as follows.

$$f_\phi : X \to Z$$
$$g_\psi : Z \to X$$

where f_ϕ is non-linear mapping for the encoder, g_ψ is non-linear mapping for the decoder, X is the input space, and Z is the latent space. The dimensionality of X is larger than that of Z due to dimension reduction. Learning of the auto encoder proceeds to obtain the optimal ϕ^* and ψ^* by updating the learnable parameters ϕ and ψ constituting the encoder and decoder in the direction of minimizing the reconstruction error $L(X, \tilde{X}) = ||X - g_\psi(f_\phi(X))||^2$. The optimal ϕ^* and ψ^* is as follows:

$$\phi^*, \psi^* = \arg\min_{\phi,\psi} L(X, \tilde{X})$$

The conventional clustering algorithms used in machine learning have the problem of not handling high-dimensional image data properly. This is because the existing machine learning algorithm works well for low-dimensional data, but its computational complexity for high-dimensional data is very high and thus, can not guarantee its performance. However, as researches on deep neural networks and deep auto encoders have been actively conducted recently, clustering techniques using such models have been studied. Due to the characteristics of deep neural networks capable of processing high-dimensional image data that have not been efficiently processed in the past, various clustering models are being developed to handle high-dimensional data effectively.

2.3.4 Deep Embedding Clustering

Xie et al. [34] effectively clustered high-dimensional data by using variation of deep auto encoders. The embedding is extracted through a deep autocoder, and the clustering is performed by changing the form of embedding gradually as the distance between the clusters is increased and the distance within the cluster is closer. The clustering process proceeds as follows. First, learn the auto encoder that minimizes the following reconstruction error.

$$L(X, \tilde{X}) = ||X - g_{\psi}(f_w(X))||^2$$

where f is non-linear mapping for the encoder, g_{ψ} is non-linear mapping for the decoder, X is the input space, and Z is the latent space. At this time, since dimension reduction occurs, the dimensionality of X is larger than that of Z. The dimension of the embedding of the auto encoder is ten dimensions. After learning of deep auto encoder is completed, the center vector μ_j^0 is extracted using the k-means algorithm. Next, a new clustering layer is added to the previously learned encoder to construct a new neural network. The output value at the last layer of new neural network is as follows.

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||^2)^{-1}}{\Sigma_{j'}(1 + ||z_i - \mu_{j'}||^2)^{-1}}$$

where $z_i \in Z$ is the vector of i^{th} data point on the embedding output by the encoder, $f_w(x_i) \in Z$ corresponding to $x_i \in X$, $i = 1, ...N$ where x_i is vector of i^{th} input data and N is the number of data points and $\mu_j \in Z$ is vector of the center point in the j^{th} cluster which is learnable parameter and $j = 1, ..., J$ where J is the number of clusters. The parameter to be updated is the parameter of the encoder w and center μ_j. The value of the center μ_j^0, already extracted by the k-means algorithm, is used as the initial center vector μ_j. q_{ij} is a distance measure indicating the degree of similarity between z_i and μ_j. That is, the probability that the point z_i belongs to the j^{th} cluster. Next, we define the loss function as follows.

$$L = \Sigma_i \Sigma_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

where p_{ij} is defined as the probability q_{ij} that the point z_i belongs to the j^{th} cluster is transformed to have an extreme distribution and is defined as follows.

$$p_{ij} = \frac{q_{ij}^2}{\Sigma_i q_{ij}} / \Sigma_{j'} \frac{q_{ij'}^2}{\Sigma_i q_{ij'}}$$

To minimize the loss function L, the parameters w and μ_j are updated as follows.

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$
$$\mu_j \leftarrow \mu_j - \eta \frac{\partial L}{\partial \mu_j}$$

where η is learning rate. The loss function L used at this time is updated in the direction that the distribution q is close to the distribution p by the Kullback-Leibler divergence. As a result, clustering is performed so as to increase the separation between clusters by increasing the distance between the clusters on the embedding, and to increase the compactness by reducing the distance within the clusters. Finally, the i^{th} data belongs to the j^{th} cluster satisfying the following equation.

$$j^* = \arg\max_j q_{ij}$$

Despite its moderate performance, existing deep embedding clustering methodologies are problematic to be applied to financial time series clustering for asset selection. This is because it works effectively only for data such as MNIST, which is commonly used in machine learning research. For financial time series data, clustering is not done properly with the methodology presented in the paper. Therefore, in this thesis, we propose a new clustering algorithm suitable for financial time series image data by improving the algorithm of existing deep embedding technique.

2.4 Geometric Brownian Motion and Monte Carlo Simulation

2.4.1 Geometric Brownian Motion

The Brownian motion is used to analyze changes in the price of financial assets. Assuming that the change in the price of financial assets follows a random walk, which is a discrete-time stochastic process, the Brownian motion expands it to a continuous-time stochastic process. Bachelier [3] introduced this Brownian motion to financial theory for the first time. It can be considered the beginning of the efficient market hypothesis that stock prices are random and unpredictable, since all predictable information is already reflected in the prices of financial assets. Since then, research on Brownian motion has continued to develop. Wiener [33] presents Brownian motion as a stochastic process and presents it as a complete mathematical concept. Osborne [25, 26] conducted a study to express the stock price return as Brownian motion. Since then, Samuelson [29, 30] introduced the Brownian motion, which modified the Brownian motion by introducing the Wiener process. Since then, he introduced the Geometric Brown motion(GBM), which modified the Brownian motion by introducing the Wiener process. The GBM is a continuous-time stochastic process that follows the lognormal distribution except for drift parameters. In the Brownian motion model, there was a problem that the stock price could be negative. However, this problem was solved by following the lognormal distribution in the GBM. If S_t which is a stochastic process satisfies the below stochastic differential equation, it is said to follow a GBM.

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

The result of solving the above differential equation is as follows.

$$S_t = S_0 exp[\sigma W_t + (\mu - \frac{\sigma^2}{2})t]$$
$$E[log(\frac{S_t}{S_0})] = (\mu - \frac{\sigma^2}{2})t$$

where, W_t is a Wiener process. μ is the drift parameter and σ is the volatility parameter where both are constants.

2.4.2 Monte Carlo Simulation

A Monte Carlo(MC) simulation, whose rationale is from The Law of Large Numbers, is a random sampling of the values that are calculated probabilistically for simulation purposes. It can be used to approximate the true distribution by generating a very large number of sample data of a certain probability distribution or stochastic process. In this thesis, we will generate stock price model data through MC simulation based on the GBM model to see if the deep embedding technique can properly cluster the generated financial time series data for simulation.

Chapter 3

Data Description and Proposed algorithm

3.1 Data Description

3.1.1 Toy Data: Simulated Financial Time Series Data from GBM

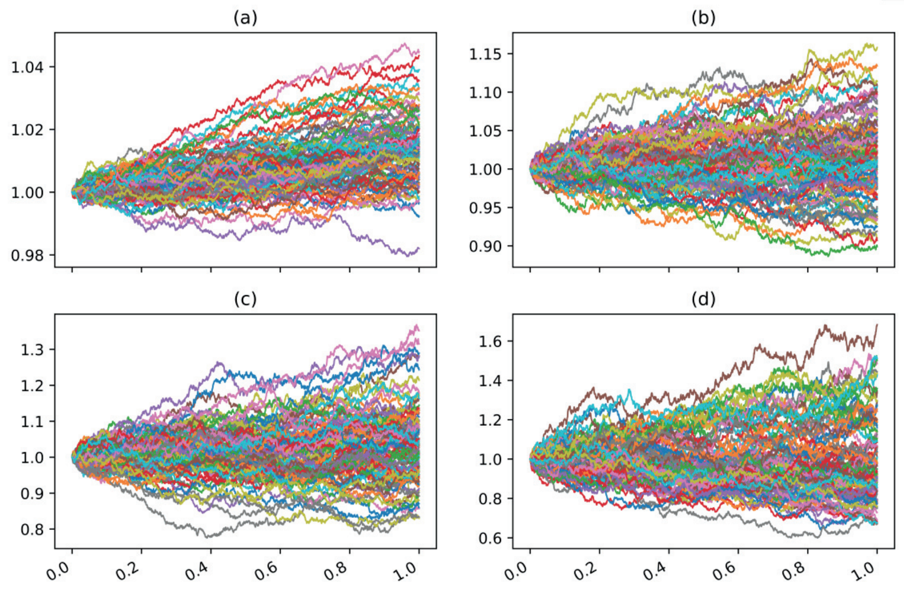

Figure 3.1: The stock price model data sampled by MC simulation based on the GBM. (a) $\sigma = 1\%$ (b) $\sigma = 5\%$ (c) $\sigma = 10\%$ (d) $\sigma = 30\%$

In this study, we investigated whether the algorithm of existing deep embedding technique clusters simulated data properly before clustering real-data. Stock price data was generated by MC simulation based on the GBM as toy data. The drift parameter μ means the annual average return of the stock price. We used 1.4%, the average annual return over the past eight years of the KOSPI index as a drift parameter. The simulations were performed by generating data with a wide range of volatility σ, which is equivalent to the annual volatility, using the range of 1% to 30 %, at 1% intervals. A total of 5 years data was sampled based on 252 trading days. Figure 3.1 shows the stock price model data sampled by MC simulation based on the GBM.

3.1.2 Real-Data: KOSPI data

Real-data used in this study are daily, weekly, and monthly closing price of stocks listed on KOSPI stock market, which is Korean representative stock market. The closing price data from May 2013 to April 2018 was collected from http://investing.com. Figure 3.2 shows the daily closing price of the four stocks listed on the KOSPI from May 2013 to April 2018.

For accurate analysis, all data are divided by the closing price of May 2, 2013, which is the first stock price, and the initial value is scaled to 1. This is because stock prices are very diverse in each stock. This is necessary because the degree of trend reflected in the image may be distorted when the stock price scaling is not performed. Figure 3.3 shows that the characteristics and intuition of the data reflected in the stock price vary according to the time resolution of monthly stocks, weekly and daily closing price of stock. Therefore, monthly, weekly, and daily experiments were conducted separately to see how clustering is performed at each time resolution.

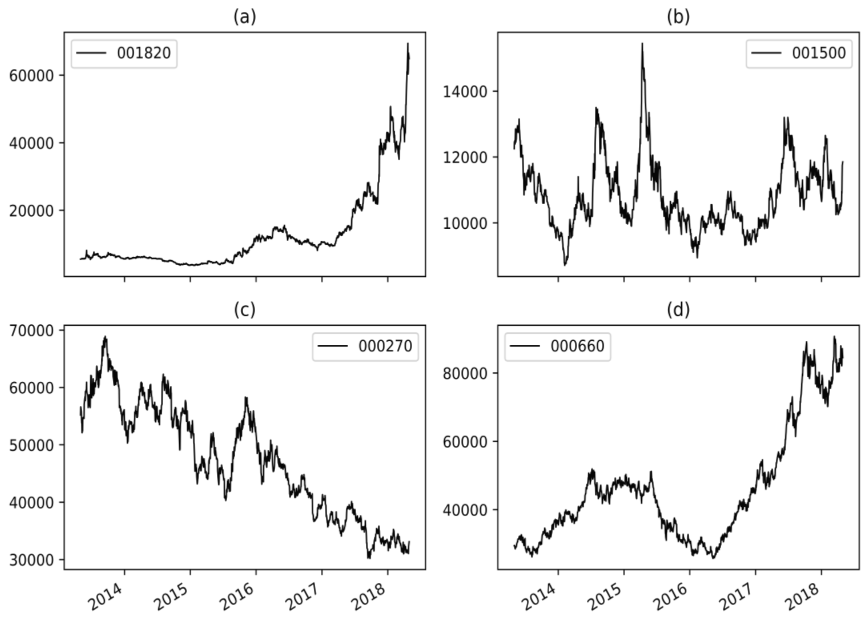

Figure 3.2: Daily closing price of four stocks in KOSPI for five years. (a) SAMWHA CAPACITOR (b) HYUNDAI MOTOR SECURITIES (c) KIA MOTORS (d) SK HYNICS

3.1.3 Data Preprocessing for Three Types of Data Set

In this thesis, we use three types of data set as shown in Table 3.1 for three experiments. First, simulated data artificially divided into two clusters. Second, stock price data of traded stocks listed in KOSPI that are artificially divided into two clusters. Third, stock price data of traded stocks listed in KOSPI data that are not artificially divided.

The first simulated data, as toy data, was based on the stock price model data generated by MC simulation based on the GBM. According to the condition of Table 3.2, toy data is artificially divided into two groups of uptrend and downtrend. Figure 3.4 shows this toy data. Similarly, the second data set is constructed by dividing the

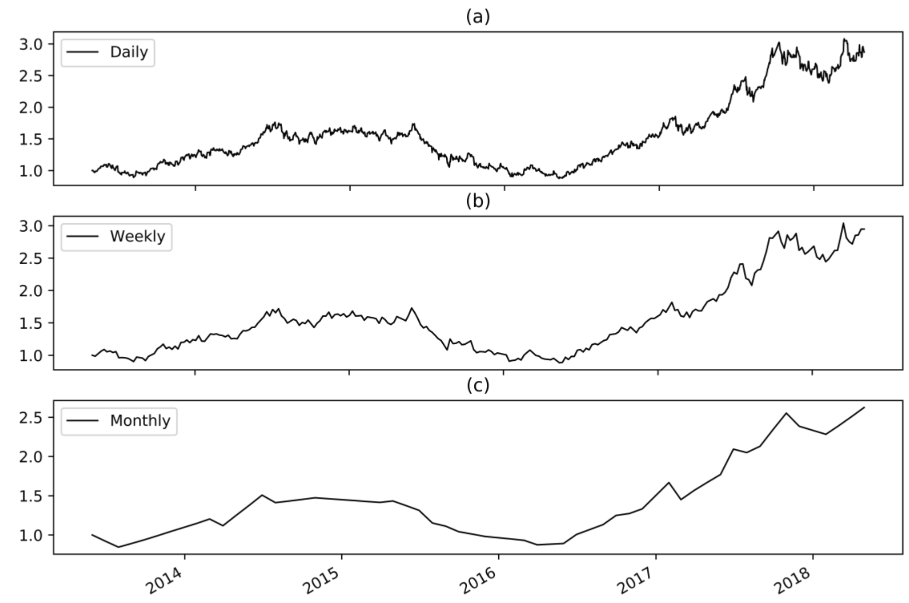

Figure 3.3: Financial time series price data based on time resolution. (a) daily (b) weekly (c) monthly

data of the stocks listed on the KOSPI into two groups artificially, uptrend and downtrend, according to the condition of Table 3.2. Figure 3.5 shows this real data. The third data set is from all stocks listed on the KOSPI. The number of optimal clusters is unknown because no artificial splitting conditions are applied.

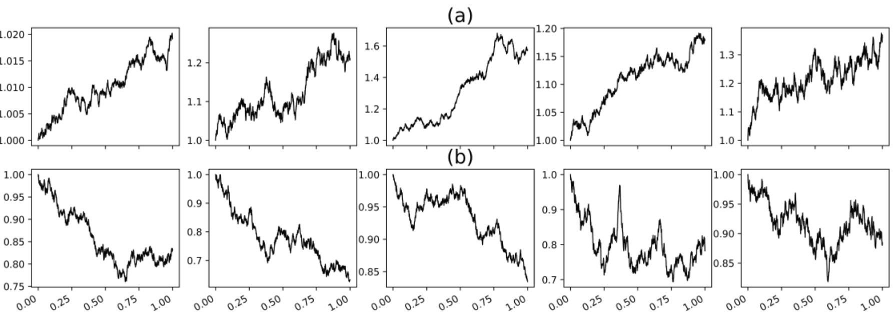

Figure 3.4: Toy data artificially divided into two groups (a) Uptrend (b) Downtrend

	Data	**Time Resolution**	**Artificially Divided Trend**	**Kinds**	**# of Stocks**
1	Simulated data based on GBM	Daily	Uptrend Downtrend	Toy	90
2	Stocks listed in KOSPI	Daily	Uptrend Downtrend	Real	92
3	Stocks listed in KOSPI	Daily, Weekly, Monthly	No Artificially Divided	Real	770

Table 3.1: Explanation of three types of data for three experiments.

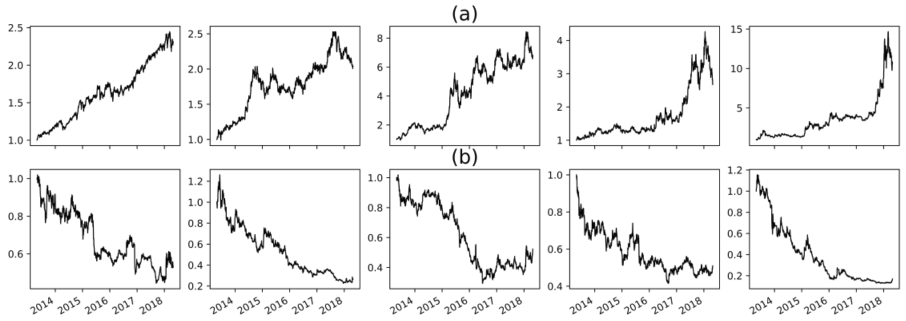

Figure 3.5: Real data(stocks in KOSPI) artificially divided into two groups (a) Uptrend (b) Downtrend

3.2 Proposed Algorithm

The existing deep embedding clustering study may not be suitable for clustering of financial time-series data because only MNIST, STL-HOG, and REUTERS data, which are typical experimental data in the field of machine learning, are used for empirical experiments. Therefore, in this thesis, we will propose a new algorithm by modifying and improving existing deep embedding algorithm to be suitable for financial time series data clustering.

Condition	**Trend**
[minimum price >= initial price] and [last price >(initial price + maximum price)/2]	Uptrend
[maximum price <= initial price] and [last price <(initial price + minimum price)/2]	Downtrend

Table 3.2: Conditions for artificial segmentation of data

3.2.1 Problems of existing algorithm

Figure 3.6 (a) shows embedding of the first data, which is artificially divided into two financial time series data as toy data, after the clustering with the existing algorithm. The dimension of embedding is ten and it is drawn three dimensional graph through principal component analysis. The distance between the clusters on the embedding is close and the boundary between the two clusters is ambiguous due to the overlap between the two clusters. Figure 3.6 (b) and (c) show the results of the clustering and the true labels with different colors according to the clusters. There is a large gap between clustering results and true labels. In other words, it can be seen that clustering of toy data, divided into two groups, artificially uptrend and downtrend, can not be performed properly. The reason why existing deep embedding techniques are not suitable for financial time series data clustering is as follows.

First, as shown in Table 4.1, the difference in scale between the data of each dimension of the embedded data is very large. Scale differences in each dimension have an undesirable effect on performance in proper clustering. According to Celebi et al. [7], the scale differences in each of these dimensions have adverse effects on the distance calculation in the clustering process and thus cause numerical instability. Figure 3.7 shows the extracted embedding by changing the angle along the axis. The scale of the range of data distribution along the axis is very different.

Second, in the process of learning, the update of the parameter of the encoder w is much higher than the update of the central parameter of each cluster μ_j, thus learning is not performed properly. In other words, $\partial L/\partial w$, the gradient value of w, the parameter of the encoder, is much larger than $\partial L/\partial \mu_j$, the gradient value of μ_j, the central parameter of each cluster, and the new cluster center is not properly performed as the cluster center because it is pushed out from cluster on the embedding of data as shown in Figure 3.8.

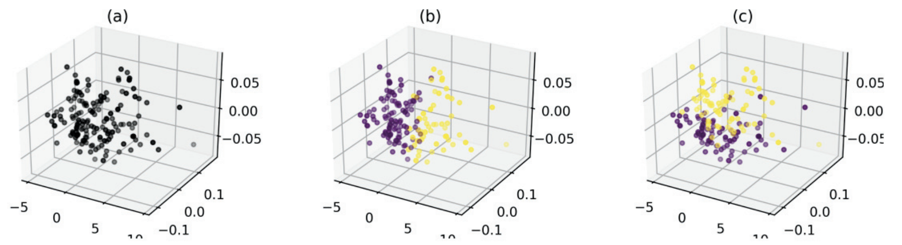

Figure 3.6: Embedding extracted as a result of existing algorithm on toy data

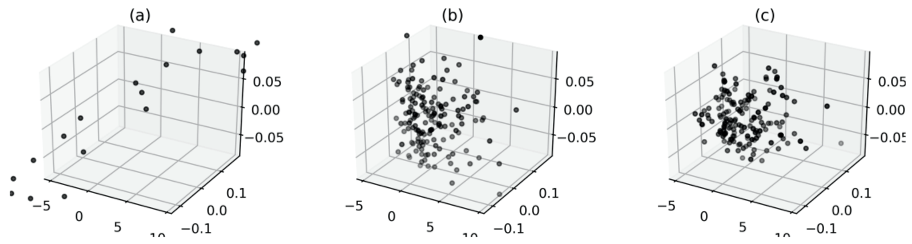

Figure 3.7: Embedding by turning the angle according to the axis direction. (a) x-axis (b) y-axis (c) z-axis

3.2.2 Proposed Algorithm

In this thesis, we solve the above problems by modifying existing algorithm and improve them to fit financial time series data. The first problem can be solved by introducing a normalization operation that can force the scale of each dimension to

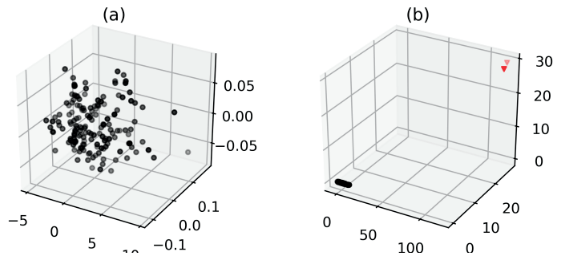

Figure 3.8: (a) Extracted results of existing algorithm (b) Embedding with center

be no large different. Among the various normalization methods, normalization by inserting a batch normalization layer showed the best clustering performance.

The second problem can be solved by forcibly allocating the center within the cluster through the k-means algorithm again, at every specific update cycle, so that the center does not deviate far out of the cluster. It is the core of the existing technique that the learning progresses so that the other data points belonging to the corresponding cluster are gradually getting closer to each other to the direction of the center of their cluster in the embedding. However, there is a problem that the cluster center is located far away from the data, and it is not possible to grasp to which cluster each data point belongs. Therefore, this problem can be solved by forcibly allocating the center within a cluster for each specific period. Algorithm 1 and 2 below shows pseudo code for the existing algorithm and the proposed algorithm, respectively.

Algorithm 1 Existing Algorithm

```
while before objective function $L_{recon}$ converges do
    Update the encoder $f_w$ and decoder $g_\psi$ in the direction that the following objective function $L_{recon}$ decreases:
    $L_{recon} = \Sigma_i ||x_i - g_\psi(f_w(x_i))||^2$
end while
Initiate encoder $f_w$, clustering layers $c_\mu$, and a new deep neural network $h_{w,\mu} = c_\mu \circ f_w$
Initialize parameters $w$ of new deep neural networks with parameters of pre-learned encoder:
$w \leftarrow w_0$
Initialize parameters $\mu$ of new deep neural networks with the center of k-mean:
$\mu \leftarrow \mu_0$
while before objective function $L_{KL-divergence}$ converges do
    Compute $q$ and $p$ every $n$ steps, for all $i, j$:
    $q_{ij} \leftarrow \frac{(1+||z_i-\mu_j||^2)^{-1}}{\Sigma_{j'}(1+||z_i-\mu_{j'}||^2)^{-1}}$
    $p_{ij} \leftarrow \frac{q_{ij}^2}{\Sigma_i q_{ij}} / \Sigma_{j'} \frac{q_{ij'}^2}{\Sigma_i q_{ij'}}$
    Update new deep neural networks $h_{w,\mu}$ in the direction that the following objective function $L_{KL-divergence}$ decreases:
    $L_{KL-divergence} = \Sigma_i \Sigma_j p_{ij} log \frac{p_{ij}}{q_{ij}}$
end while
```

Algorithm 2 Proposed Algorithm

```
while before objective function $L_{recon}$ converges do
    Initiate a batch $B = x_1, ..., x_m$ and $\mu_B$, $\sigma_B^2$ for batch normalization:
    $\mu_B \leftarrow \frac{1}{m}\Sigma_{i=1}^{m} x_i$

    $\sigma_B^2 \leftarrow \frac{1}{m}\Sigma_{i=1}^{m}(x_i - \mu_B)^2$

    $\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$

    Update the encoder $f_w$ and decoder $g_\psi$ in the direction that the following objective function $L_{recon}$ decreases:

    $L_{recon} = \Sigma_i ||x_i - g_\psi(f_w(x_i))||^2$

end while

Initiate encoder $f_w$, clustering layers $c_\mu$, and a new deep neural network $h_{w,\mu} = c_\mu \circ f_w$
Initialize parameters $w$ of new deep neural networks with parameters of pre-learned encoder:

$w \leftarrow w_0$

Initialize parameters $\mu$ of new deep neural networks with the center of k-mean:

$\mu \leftarrow \mu_0$

while before objective function $L_{KL-divergence}$ converges do

    Compute $q$ and $p$ every $n$ steps, for all $i, j$:

    $q_{ij} \leftarrow \frac{(1+||z_i-\mu_j||^2)^{-1}}{\Sigma_{j'}(1+||z_i-\mu_{j'}||^2)^{-1}}$
    $p_{ij} \leftarrow \frac{q_{ij}^2}{\Sigma_i q_{ij}} / \Sigma_{j'} \frac{q_{ij'}^2}{\Sigma_i q_{ij'}}$

    Replace center vector $\mu$ with k-mean center $\mu_0$: $\mu \leftarrow \mu_0$

    Update new deep neural networks $h_{w,\mu}$ in the direction that the following objective function $L_{KL-divergence}$ decreases:

    $L_{KL-divergence} = \Sigma_i \Sigma_j p_{ij} log \frac{p_{ij}}{q_{ij}}$

end while
```

Chapter 4

Experimental Results

We have confirmed that the existing algorithm can not properly cluster for financial time series data through toy data. We also proposed a new algorithm by identifying the cause and correcting and improving the existing algorithm. In this section, we will show how the newly proposed clustering algorithm can properly cluster financial time series data by experimenting with the three types of data we prepared earlier. We will look at the results of each embedding for all three data sets, and compare the performance of the proposed algorithm with those of the existing algorithm through various performance measures used for clustering performance evaluation.

4.1 Performance Measure

In this thesis, we will use the accuracy, ARI, NMI, Silhouette coefficient, Calinski-Harabasz score, and Davies-Bouldin index as the performance measure of clustering. Accuracy, ARI, and NMI are measures that can be measured on the assumption that the true label is known, so they can be measured only for the first toy data and the second actual KOSPI data that are artificially divided into two types and the true label is known for. The third data, which was not artificially divided, had only the silhouette coefficient, Calinski-Harabasz score, and Davies-Bouldin score.

The accuracy can be expressed as $\Sigma_i 1_{(y_i=\hat{y_i})}/N$, a ratio of the number of data in which the predicted value and the true value are equal, to the total number of data. The Adjusted Random Index (ARI) indicates the degree of random assignment of the cluster. Closer to 0 means that the data cluster allocation is random, and the closer to 1, the opposite. Normalized Mutual Information (NMI) is a normalized measure of interdependence between two clusters. The closer to 0, the more interdependence disappears. The closer to 1, the greater the interdependence. Accuracy, ARI, and NMI are all postulated to know the true label. In the case of clustering problems, there is generally no label value for the data class, and thus the accuracy of the classification based on the true label can not be measured unlike the classification problem. In the first and second experiments, however, accuracy, ARI, and NMI were calculated because we used toy data and KOSPI data artificially divided into two groups.

silhouette coefficients indicate whether the overall clustering is performed properly by comparing the distance between one data and the other data in the predicted cluster and the distance from the data in the other cluster [8]. The larger the coefficient, the better the clustering is done. The Calinski-Harabaz score is defined as the ratio of intra-cluster deviation to inter-cluster deviation [5]. The larger the value, the more appropriately the clustering was performed. The Davies-Bouldin index is a measure related to ratio calculated by intra-cluster distance and inter-cluster distance, and the lower the value, the better the clustering [10]. The silhouette coefficient, the Calinski-Harabaz score, and the Davies-Bouldin index do not need to know the true label value. Therefore, we measured for the first, second, and third data set in the experiments.

4.2 Experiments for First and Second Data Set(the number of custer = 2)

4.2.1 First Experiment for Toy Data

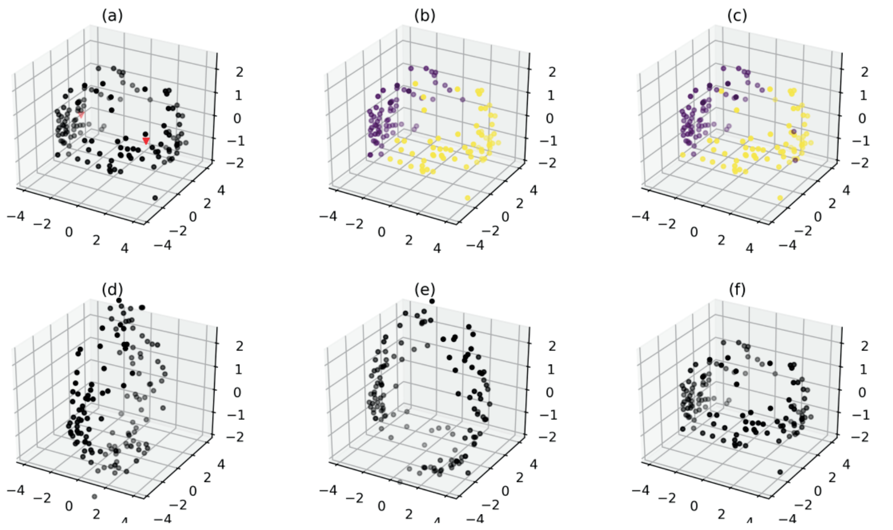

Figure 4.1: Results of embedding extracted by proposed algorithm for toy data(top) Embedding with varying angles along the axis(bottom). (a) embedding (b) result of clustering (c) true labels (d) x-axis (e) y-axis (f) z-axis

The first experiment is an experiment to verify whether the proposed algorithm performs clustering properly for the simulated data, which is the first data in Table 3.1. Figure 4.1 (a) shows the embedding extracted by the proposed algorithm. Compared to (a) in Figure 3.6, it can be seen that the clustering more clearly with two clusters is visually distinct. Comparing the clustering result in (b) of Figure 4.1 with the true label in (c) is quite different from the result of (b) and (c) in Figure 3.6. Existing algorithm do not perform clustering properly, and there is a large gap between clustering results and true labels. On the other hand, comparing the cluster labeling results with the true label values through the proposed algorithm, we can see that the two results are quite similar.

The first problem of the existing algorithm is that the scale difference in the range of data distribution for each dimension(axis) is large. The problem is solved by introducing a batch normalization layer which can adjust the scale of each dimension(axis) without any difference of deviation. (d), (e), and (f) in Figure 4.1 show the extracted embedding by angles along the axis. Compared to Figure 3.7, it can be seen that the scale of the range in which the data is distributed along the axis is very uniform. As shown in Table 4.1, it can be seen that the standard deviation of data distribution for each axis is fairly uniform around 1.

	Axis #	1	2	3	4	5	6	7	8	9	10
dec	Mean	20.1	4.2	32.1	-82	33.5	-37	12	-7	25.6	19.3
	Std	0.54	0.11	0.85	2.2	0.9	0.99	0.31	0.18	0.68	0.52
ours	Mean	1.6	0.4	-4	-0.7	1	4.3	-3.7	-4.5	1.5	-0.2
	Std	1.09	1.06	1.16	1.24	1.25	1.1	1.11	1.07	1.08	1.10

Table 4.1: Basic statistics for each axis of embedding extracted from toy data by existing and proposed algorithm

The second problem, in which the update rate of the parameters of the encoder, w, is much higher than the update rate of the central parameter of each cluster, μ_j, is solved by reallocating the center obtained from the k-means at every n steps to the inside the center of the cluster. It can be seen that new cluster center has played a role properly as a clustering center because it has not been pushed out of the data cluster of the embedding.

4.2.2 Second Experiment for Real-Data

We show that the proposed algorithm can perform clustering properly in the toy data. Experiments were carried out to find out whether the proposed algorithm performs clustering properly on the real KOSPI data, which is the second data of Table 3.1. Figure 4.2 shows the extracted embedding result of clustering of the KOSPI data by the proposed algorithm. Observing the embedding, we can see clearly two distinct clusters. This is a reflection of the fact that two artificially divided data were used.

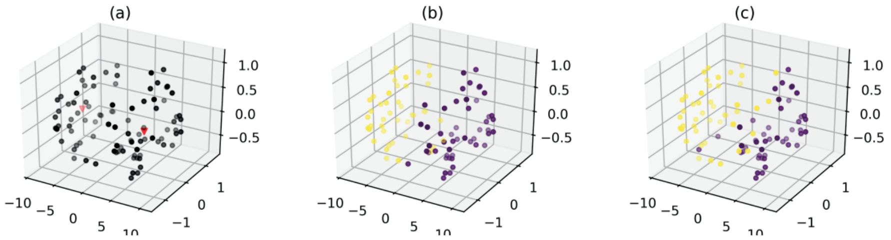

Figure 4.2: Results of embedding extracted proposed algorithm for real KOSPI data(the number of cluster = 2). (a) embedding (b) result of clustering (c) true labels

As in the case of simulated data experiment results, the first problem in which the scale of the range in which the data is distributed in each dimension(axis) in the real KOSPI data is different is solved by introducing a batch normalization layer. In addition, we can see that the second problem, in which the updating of the parameter of the encoder is much higher than the update of the central parameter of each cluster, and the learning did not proceed properly can be solved by proposed method as seen in Figure 4.2 (a).

4.2.3 Performance Evaluation

Table 4.2 shows the results of clustering performance measurement for the toy data and the real KOSPI data for the existing algorithm and the proposed algorithm. In both cases, the performance of the proposed algorithm is much better than that of the existing algorithm.

Performance	**Toy Data**		**Real KOSPI Data**	
Measure	**Existing**	**Proposed**	**Existing**	**Proposed**
Accuracy	0.554	**0.947**	0.533	**0.837**
ARI	0.007	**0.798**	-0.007	**0.448**
NMI	0.010	**0.702**	0.003	**0.362**
Silhouette	0.640	**0.728**	0.542	**0.673**
Calinski-Harabasz	451	**734**	172	**388**
Davies-Bouldin	0.507	**0.374**	0.632	**0.418**

Table 4.2: Performance evaluation of existing and proposed algorithm

We compare the performance of the proposed algorithm with hierarchical clustering, which is mainly used in the field of asset clustering. Hierarchical clustering uses the correlation matrix of the expected returns of the financial time series as a distance matrix. On the otherhand, the proposed algorithm performs clustering on the embedding extracted using image of financial time-series data as input. It is a completely different approach and we can not directly compare the two methodologies in terms of typical performance measures for clustering. We can not use performance measures that use only X data to compare the two. Only the performance measures using the label value y will be completely comparable. Therefore, we compared performance using Accuracy, ARI, and NMI that can be used when knowing true labels. As shown in the Table 4.3, the proposed methodology is superior to the hierarchical clustering methodology in entire performance measures.

Performance Measure	**Toy Data**		**Real KOSPI Data**	
	Hierarchical	Proposed	Hierarchical	Proposed
Accuracy	0.609	**0.947**	0.652	**0.837**
ARI	0.039	**0.798**	0.086	**0.448**
NMI	0.046	**0.702**	0.121	**0.362**

Table 4.3: Performance evaluation of hierarchical clustering and proposed algorithm

4.3 Experiment for Third Data Set(the number of custer $\geq$ 2)

4.3.1 Experiment for Third Data and Performance Evaluation

The third data set (KOSPI data - monthly, weekly, and daily) without artificially split conditions were clustered from 2 to 11 clusters. Table 4.4 shows the results of Shilouette coefficient, Calinski-Harabasz score, and Davies-Bouldin index for each monthly, weekly, and daily cases. The two highest values for each measure are shown in bold. According to the experimental results, two or three clusters can be regarded as appropriate clusters based on three performance measures in monthly and daily data. In weekly data, the number of clusters two is the most appropriate, and other proper the number of clusters is four or five.

4.3.2 Interpretation to Intuition in the Embedding

Figure 4.3 and Figure 4.4 show the clustering of monthly and daily data into 2 and 3 clusters and the clustering of weekly data into 2,4 and 5 clusters according to the results of Table 4.4. Figure 4.5 shows KOSPI data belonging to each cluster for the monthly data of the three clusters among the clustering results. The shape of the overall embedding can be seen to represent the shape of the ellipsoid. The more data are distributed in the clusters close to both ends, the larger the uptrend or the greater the downtrend. The data in the central cluster also represents the sideways trend. In real data, there may be various images showing a trend between the uptrend and the downtrend, which is a result of fitting with intuition. In other words, if clustering is performed through the proposed algorithm, it can be seen that each cluster actually reflects the intuitive relationship between assets inherent in high-dimensional image data.

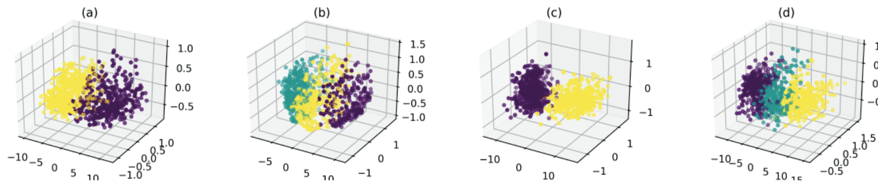

Figure 4.3: (a) Two clusters of monthly data (b) Three clusters of monthly data (c) Two clusters of daily data (d) Three clusters of daily data

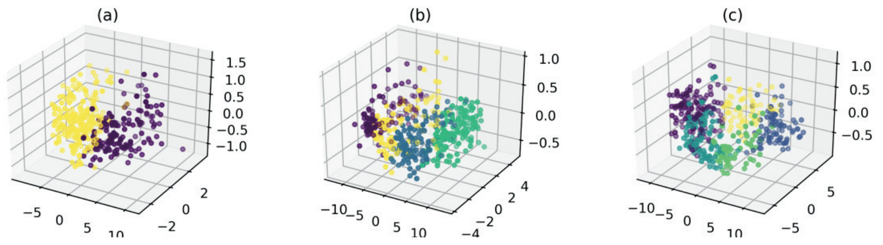

Figure 4.4: Weekly Data (a) two clusters (b) four clusters (c) five clusters

Monthly KOPSPI Data										
Performance Measures	# of Cluster									
	2	3	4	5	6	7	8	9	10	11
Silhouette	**0.673**	**0.524**	0.416	0.361	0.368	0.344	0.4	0.377	0.347	0.337
Calinski-Harabasz	**3534**	**2694**	1059	1692	929	962	1319	1123	1010	943
Davies-Bouldin	**0.442**	**0.611**	0.827	0.932	0.853	0.951	0.766	0.818	0.874	0.888
Weekly KOSPI Data										
Performance Measures	# of Cluster									
	2	3	4	5	6	7	8	9	10	11
Silhouette	**0.534**	0.418	0.383	**0.426**	0.379	0.353	0.392	0.369	0.379	0.387
Calinski-Harabasz	**1782**	1478	**1719**	1232	1036	782	1252	962	1057	1048
Davies-Bouldin	**0.653**	0.879	0.959	**0.772**	0.825	0.873	0.825	0.833	0.840	0.795
Daily KOSPI Data										
Performance Measures	# of Cluster									
	2	3	4	5	6	7	8	9	10	11
Silhouette	**0.720**	**0.599**	0.441	0.402	0.395	0.368	0.353	0.275	0.368	0.409
Calinski-Harabasz	**4843**	**4057**	1079	1176	1001	1095	837	764	989	1129
Davies-Bouldin	**0.371**	**0.509**	0.8	0.813	0.789	0.859	0.919	1.156	0.862	0.767

Table 4.4: Performance evaluation of proposed algorithm for third experiment. Prepare three sets of data for monthly, weekly, and daily time resolution of KOSPI data. And clustered them into a proposed model and then obtained the performance measures for each cluster number.

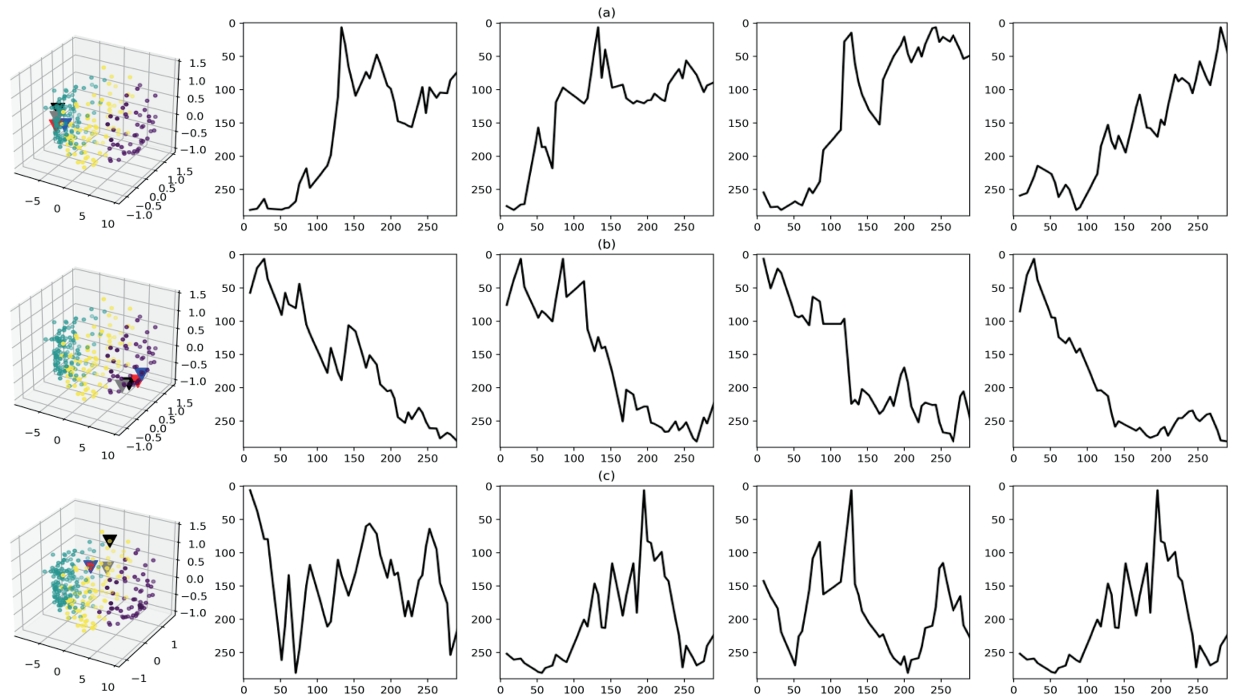

Figure 4.5: The result of clustering three clusters using monthly data and actual shape of embedding. (a) uptrend cluster (b) downtrend cluster (c) sideways cluster 3

Chapter 5

Conclusion

5.1 Conclusion

Clustering is one of the important issues in portfolio research on existing asset selection. Clustering is important in portfolio research because it can have a diversification effect by selecting assets from dissimilar clusters, and it can be time-efficient in asset selection through elimination of redundancy.

In recent years, researches have been actively conducted to apply machine learning and deep learning to asset clustering in the financial sector. However, most of the existing studies use simple low-dimensional time-series data based on correlations of expected returns for clustering. This has the disadvantage that it is difficult to reflect the intuitive relation between the assets reflected in the high-dimensional image data and the time scale correction effect according to the time resolution. Therefore, in this thesis, we tried to solve the above problems through deep embedding technique which can process high dimensional image.

There are three main types of data sets. First, the toy data simulated by the GBM are artificially divided into two groups: uptrend and downtrend. The second is real KOSPI daily closing price data, which is artificially divided into two groups. The third is KOSPI daily, weekly, and monthly closing price data, which are not artificially divided.

Existing algorithm have failed to cluster the financial time series properly. The causes are as follows. The first is that the scale difference on the data distribution of each dimension(axis) on the embedding is very large. The second is that the cluster center does not play its own role because the cluster center deviates out of the embedding. In this thesis, we propose a new algorithm to solve this problem by improving the algorithm in the following way. The first is to insert a batch normalization layer, and the second is to reallocating center every n steps by using k-means.

In the empirical experiment, we verified that the proposed algorithm properly performed clustering on the three data sets prepared above. For the first and second data sets, the cluster performance of the proposed algorithm is much better than that of the existing algorithm in terms of six performance measures. For the third data, we calculated the optimal number of clusters by daily, weekly, and monthly through three performance measures, and examined the distribution of data in the embedding.

The results of this study are as follows. First, existing algorithm are not suitable for financial time series clustering. Therefore, we propose new clustering algorithm for financial time series clustering. The proposed algorithm has superior quantitative performance not only in the simulated data but also in the real data.

Second, we clustered the monthly, weekly, and daily data of the actual KOSPI data through the proposed algorithm and calculated the optimal number of clusters according to the performance measures. In monthly and daily data, two or three are optimal clusters, and in weekly data, 2, 4, and 5 are optimal clusters for KOSPI.

Lastly, it is confirmed that the clustering result obtained through the proposed algorithm can actually reflect the intuitive relationship of assets inherent in high dimensional image data. For example, we can see the patterns or long-term trends in high-dimensional image data by clustering monthly data into three groups of optimal clusters, clustering into three groups: uptrend, downtrend, sideways trend.

5.2 Future Direction

We can extend our approaches to apply to other problems which need to do asset clustering. For example, instead of using hierarchical clustering as described above for portfolio management, we can use our proposed algorithm. We can examine whether our proposed algorithm can actually work more effectively in portfolio selection. Therefore, in our future research, we can try to verify whether the proposed model is more effective in reflecting the intuitive relationship between assets intrinsic in high-dimensional images in the portfolio. To verify this, we can consider not only various benchmark models including hierarchical clustering but also performance measures that can measure appropriately the diversification effect and return of the portfolio.

Bibliography

[1] S. Aghabozorgi and Y. W. Teh, *Stock market co-movement assessment using a three-phase clustering method*, Expert Systems with Applications, 41 (2014), pp. 1301–1314.

[2] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski, *Visualizing time-oriented data—a systematic view*, Computers & Graphics, 31 (2007), pp. 401–409.

[3] L. Bachelier, *Théorie de la spéculation*, in Annales scientifiques de l'École normale supérieure, vol. 17, 1900, pp. 21–86.

[4] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, *Evaluating multiple classifiers for stock price direction prediction*, Expert Systems with Applications, 42 (2015), pp. 7046–7056.

[5] T. Caliński and J. Harabasz, *A dendrite method for cluster analysis*, Communications in Statistics-theory and Methods, 3 (1974), pp. 1–27.

[6] R. C. Cavalcante, R. C. Brasileiro, V. L. Souza, J. P. Nobrega, and A. L. Oliveira, *Computational intelligence and financial markets: A survey and future directions*, Expert Systems with Applications, 55 (2016), pp. 194–211.

[7] M. E. Celebi, H. A. Kingravi, and P. A. Vela, *A comparative study of efficient initialization methods for the k-means clustering algorithm*, Expert systems with applications, 40 (2013), pp. 200–210.

[8] G. Chen, S. A. Jaradat, N. Banerjee, T. S. Tanaka, M. S. Ko, and M. Q. Zhang, *Evaluation and comparison of clustering algorithms in analyzing es cell gene expression data*, Statistica Sinica, (2002), pp. 241–262.

[9] W. Dai, J.-Y. Wu, and C.-J. Lu, *Combining nonlinear independent component analysis and neural network for the prediction of asian stock market indexes*, Expert systems with applications, 39 (2012), pp. 4444–4452.

[10] D. L. Davies and D. W. Bouldin, *A cluster separation measure*, IEEE transactions on pattern analysis and machine intelligence, (1979), pp. 224–227.

[11] S. Dhar, T. Mukherjee, and A. K. Ghoshal, *Performance evaluation of neural network approach in financial prediction: Evidence from indian market*, in 2010 International Conference on Communication and Computational Intelligence (INCOCCI), IEEE, 2010, pp. 597–602.

[12] P. D'Urso, C. Cappelli, D. Di Lallo, and R. Massari, *Clustering of financial time series*, Physica A: Statistical Mechanics and its Applications, 392 (2013), pp. 2114–2129.

[13] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1, Springer series in statistics New York, 2001.

[14] A. Grané and H. Veiga, *Wavelet-based detection of outliers in financial time series*, Computational Statistics & Data Analysis, 54 (2010), pp. 2580–2593.

[15] H.-S. Guam and Q.-S. Jiang, *Cluster financial time series for portfolio*, in 2007 international conference on wavelet analysis and pattern recognition, vol. 2, IEEE, 2007, pp. 851–856.

[16] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, *Deep learning for visual understanding: A review*, Neurocomputing, 187 (2016), pp. 27–48.

[17] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167, (2015).

[18] D. León, A. Aragón, J. Sandoval, G. Hernández, A. Arévalo, and J. Niño, *Clustering algorithms for risk-adjusted portfolio construction*, Procedia Computer Science, 108 (2017), pp. 1334–1343.

[19] X. Li, Z. Deng, and J. Luo, *Trading strategy design in financial investment through a turning points prediction scheme*, Expert Systems with Applications, 36 (2009), pp. 7818–7826.

[20] F. Liu and J. Wang, *Fluctuation prediction of stock market index by legendre neural network with random time strength function*, Neurocomputing, 83 (2012), pp. 12–21.

[21] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, *Financial time series forecasting using independent component analysis and support vector regression*, Decision Support Systems, 47 (2009), pp. 115–125.

[22] F. Luo, J. Wu, and K. Yan, *A novel nonlinear combination model based on support vector machine for stock market prediction*, in 2010 8th World Congress on Intelligent Control and Automation, IEEE, 2010, pp. 5048–5053.

[23] H. Markowitz, *Portfolio selection*, The journal of finance, 7 (1952), pp. 77–91.

[24] S. Nanda, B. Mahanty, and M. Tiwari, *Clustering indian stock market data for portfolio management*, Expert Systems with Applications, 37 (2010), pp. 8793–8798.

[25] M. F. Osborne, *Brownian motion in the stock market*, Operations research, 7 (1959), pp. 145–173.

[26] ——, *Periodic structure in the brownian motion of stock prices*, Operations Research, 10 (1962), pp. 345–379.

[27] N. Rochester, J. Holland, L. Haibt, and W. Duda, *Tests on a cell assembly theory of the action of the brain, using a large digital computer*, IRE Transactions on information Theory, 2 (1956), pp. 80–93.

[28] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), p. 386.

[29] P. A. Samuelson, *Proof that properly anticipated prices fluctuate randomly*, Industrial Management Review, 6 (1965), p. 41.

[30] ——, *Mathematics of speculative price*, Siam Review, 15 (1973), pp. 1–42.

[31] F. E. Tay and L. Cao, *Application of support vector machines in financial time series forecasting*, omega, 29 (2001), pp. 309–317.

[32] V. Tola, F. Lillo, M. Gallegati, and R. N. Mantegna, *Cluster analysis for portfolio optimization*, Journal of Economic Dynamics and Control, 32 (2008), pp. 235–258.

[33] N. Wiener, *Differential-space*, Journal of Mathematics and Physics, 2 (1923), pp. 131–174.

[34] J. Xie, R. Girshick, and A. Farhadi, *Unsupervised deep embedding for clustering analysis*, in International conference on machine learning, 2016, pp. 478–487.

국문초록

자산 선택 및 포트폴리오 분야에서 군집화에 대해 활발히 연구가 진행되고 있다. 특히 최근, 이러한 자산 군집화 연구에 기계학습 및 심층 학습 방법론을 적용하고자 하는 사례가 증가하고 있다. 기존의 상관관계 분석만으로는 고차원 이미지 데이터에 반영된 장기적 추세, 패턴 등의 직관적인 자산간의 관계를 반영하기 어렵기 때문이다. 따라서 본 연구는 고차원 이미지를 처리할 수 있는 심층 임베딩 기법을 통해 금융 시계열을 군집화할 수 있는 방법을 연구하였다. 기존 알고리즘이 금융 시계열 데이터에 적절하지 않음을 보이고, 본 연구진이 제안한 새로운 알고리즘이 기존 알고리즘보다 군집화를 더 적절히 수행할 수 있음을 보였다. 또한, 제안된 알고리즘을 통해 실제 KOSPI 데이터를 군집화하여 각종 성능 척도를 통해 최적 군집 수를 산출해 보았으며 실제 고차원 이미지에 반영된 직관적인 자산간의 관계가 군집화 결과에도 반영될 수 있는지를 살펴보았다. 또한, 본 논문의 연구 결과를 바탕으로, 후속 연구를 통해 실제로 포트폴리오 구성에 이러한 연구 결과를 반영을 하게 됐을 때의 실제 효과에 대해 다양한 포트폴리오의 성능 측정 측도와 벤치마크 결과와의 비교를 통해 보여줄 수 있을 것이다.

주요어: 금융 시계열 군집화, 자산 군집화, 자산 선택, 심층 임베딩 네트워크, 심층 학습, 장기적 추세

학번: 2017-21537