PRÉDICTION DE LA TENDANCE DES ACTIONS BASÉE SUR LES RÉSEAUX CONVOLUTIFS GRAPHIQUES ET LES LSTM

par

Mingxuan Sun

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 30 août 2022

Le 2022/08/30

le jury a accepté le mémoire de Monsieur Mingxuan Sun
dans sa version finale.

Membres du jury

Professeur Shengrui Wang
Directeur de recherche
Département Faculté des sciences

Professeur Bessam Abdulrazak
Président-rapporteur
Département Faculté des sciences

Belkacem Chikhaoui
Professeur
Membre externe
Département Informatique

Université Teluq

Sommaire

Comme les prix des actions évoluent au fil des décennies, la tendance et le prix d'une action sont souvent utilisés pour effectuer des prévisions en bourse. Bien anticiper la tendance future des actions peut non seulement aider les décideurs à mieux estimer les possibilités de profit, mais aussi les risques. Dans cette thèse, une approche quantitative est présentée pour prédire les fluctuations d'actions. L'approche se base sur une méthode de clustering pour identifier la tendance des actions à partir de leurs données historiques. C'est un type particulier de clustering appliqué sur des séries chronologiques. Il consiste à découvrir les tendances des actions sur des intervalles de temps, tel que des tendances haussières, des tendances baissières, et ensuite d'utiliser ces tendances pour prédire leurs états futurs. La méthode de prédiction proposée se base sur les réseaux de neurones convolutionnels graphiques et des réseaux récurrents mémoire pour la prédiction. Cette méthode fonctionne également sur des ensembles de données où la proportion des classes est déséquilibrée. Les données historiques des actions démontrent que la méthode proposée effectue des prévisions précises. La méthode proposée peut ouvrir une nouvelle perspective de recherche pour le clustering de séries chronologiques, notamment l'utilisation d'autres réseaux de neurones graphiques pour prédire les tendances des actions.

Mots-clés : Séries temporelles, Clustering, Prédiction de données financières

Summary

As stocks have been developing over decades, the trend and the price of a stock are more often used for predictions in stock market analysis. In the field of finance, an accurate stock future trending can not only help decision-makers estimate the possibility of profit, but also help them avoid risks. In this research, we present a quantitative approach to predicting the trend of stocks in which a clustering model is employed to mine the stock trends patterns from historical stock price data. Stock series clustering is a special kind of time series clustering. We aim to find out the trend types, e.g. rising, falling and others, of a stock at time intervals, and then make use of the past trends to predict its future trend. The proposed prediction method is based on Graph Convolutional Neural Network for clustering and Long Short-Term Memory model for prediction. This method is suitable for the data clustering of unbalanced classes too. The experiments on real-world stock data demonstrate that our method can yield accurate forecasts. In the long run, the proposed method can be used to explore new possibilities in the research field of time series clustering, such as using other graph neural networks to predict stock trends.

Key words : Time series, Clustering, Finance data prediction, Graph Convolutional Network, LSTM

Acknowledgments

I would like to express my sincere thanks to Professor Shengrui Wang, my supervisor, for all his suggestions and constant support during the preparation and composition of this thesis.

I would like to thanks to Professors Froduald Kabanza, Pierre-Marc Jodoin and Mohamed Mehdi Najjar for their deep knowledge and valuable input in their courses.

Thanks to my friends Rongbo Chen, Jianfei Zhang, Jean Paul Latyr Faye. They help me generally throughout my study that I understand more.

Thanks also owing to every member of jury and the staff of the Département d'informatique for their assistance during my studies in Sherbrooke. With their help, I can be graduated.

Finally, thanks to my parents, without their support I would never on this stage.

Table of Contents

List of Figures

List of Tables

Abbreviations

AHP Analytic hierarchy process

ANN Artificial Neural Network

ARIMA AutoRegressive Integrated Moving Average

ARMA Autoregressive–moving-average

BP Back propagation

CNN Convolutional Neural Networks

DTW Dynamic time warping

DWT Discrete Wavelet Transform

EPCNN Evolutionary Partial Connection Neural Network

GCN Graph Convolutional Network

LS-SVM Least squares support vector machines

LSTM Long short-term memory

SVM Support vector machines

Introduction

The main focus of this thesis is to develop a stock trend prediction model based on time series clustering. The framework employed here consists in using a graph convolutional neural network for time series clustering, and a long and short-term memory (LSTM) neural network for stock trend prediction.

The stock market has attracted the attention of many investors and investment institutions due to its high returns and high risk. In order to increase the profit and reduce the risk in investment, stock trend prediction has been investigated to forecast stock price tendency in the long run. On the other hand, the stock market is inherently complex, dynamic and chaotic due to the joint actions of many factors including political and economic environments, company conditions, supply and demand, investor psychology, etc, making the prediction of stock market price movement challenging. There have been a considerable number of methods reported for stock forecast, e.g., autoregressive model, support vector machine (SVM), convolutional neural network (CNN), and LSTM. Although these models have shown their usefulness, they may oversimplify the complexity of the financial market by assumptions such as 1) dichotomous stock trend, i.e. existing methods have focused exclusively on either rising or falling stock trends in an individual stock ; 2) uniform distribution of trends, i.e. existing methods are not effective in dealing with dynamic stock trends, e.g., a stock price may cease rising and start dropping at some turning points, such information is very important for investors ; 3) independence between various stocks : existing methods consider each stock as an isolated entity and ignore interactions between various companies' stocks during different periods.

To address the above issues, we propose a novel GCN-LSTM method that integrates graph neural network (GCN) and LSTM for stock trend prediction. GCN-

LSTM leverages a GCN to learn trend patterns in an unsupervised way and a LSTM model to forecast the stock trends based on the GCN clustering results. We transform time series into a set of segments first and then cluster these segments according to the Dynamic time warping (DTW) distance between them. By such clustering process, GCN can discover various underlying trend patterns at each time window of time series. Then, we label and serialize the corresponding patterns of the original sequences according to the time series partition. The experiments on real-world stock data demonstrate that our method can yield accurate forecasts. The main contributions of this thesis are as follows : 1) we proposed a new method for building adjacency matrix in GCN for dealing with unbalance data ; 2) we improved the GCN clustering method to render it suitable to discover trend patterns from the time series segments ; 3) we evaluated our proposed method based on financial gain on real-world data.

The structure of the thesis is as follows. Chapter 1 discusses main issues of stock trends prediction and time series clustering. Chapter 2 provides a review of the previous research on stock prediction and presents some background knowledge of the research. Chapter 3 describes the model of time series clustering based on GCN and stock trend prediction process based on LSTM, including data preprocessing, measuring the similarity of time series, building the adjacency matrix for GCN, clustering process based on GCN, and prediction process based on LSTM. Chapter 4 gives the experiments results analysis and presents the financial evaluation. Finally, we conclude the thesis and discuss the future directions and enhancements to this work.

Chapter 1

Stock Trends Prediction and Time Series

In this chapter, we start with presenting some background information and describe the challenges of the study in this thesis. Then we specify the objective of this thesis.

1.1 Stock Trend Prediction

Stock trends prediction studies trending relationships in historical data and makes use of these relationships to predict future trends. It helps decision makers by providing information about an asset's upcoming financial position. Among them, the stock market is regarded as one of the most complex financial systems, consisting of various components or stocks whose prices fluctuate greatly over time. It is well known that the stock market is highly sensitive and susceptible to rapid changes. Stock market forecasting involves revealing market trends over time. Moreover, they behave in a highly nonlinear and dynamic manner. The standard random walk assumption of stock future prices may conceal the fact of the variance of the price is a complex nonlinear process. To reveal the rule of variation and to make the forecasting of future prices more reliable, artificial intelligence will be useful. A valuable forecasting method can not only help decision-makers capture the possibility of profit, but also

help them avoid risks.

In essence, stock trend prediction is a time series forecasting problem. A time series is a series of data points arranged in the order of time occurrence. It is considered as dynamic because its data change over time, meaning that the value of each point in the time series is one or more observations made in chronological order. The price of a stock is changing over time. A general pipeline of steps includes a) transforming and preprocessing the historical stock data to time series; b) mining trend patterns from the series data based on a time series analysis model; c) predicting the stock trend in a future time based on the mined patterns. For mining trend patterns from the series data, an available method is clustering. Time series clustering can find interesting patterns in time series data sets. It is an important problem for researchers in different fields, such as: identifying dynamic changes in time series, prediction and recommendation, pattern discovery, etc. Stock series data clustering is a kind of special time series clustering, we aim to identify the trend pattern of the stock market at each time point through its cluster belonged, so as to predict the trend pattern of the stock in the future.

1.2 Review of the research on time series trend prediction

Research on time series analysis has been an active area since the middle of last century. As early as 1950s to 1980s [1, 2], most scholars made use of the ARMA model to solve time series prediction problems. In 1991, Trench W.F. [3] improved the ARMA model with explicit weighting coefficients to predict time series within a limited time range. At that time, the problem of the long span of time series appeared. The method was unable to achieve good results for time series with a long time range. In 2005, Abonyijanos [4] used a clustering method for fuzzy segmentation of multivariate time series, and since then many other machine learning methods have been applied to the research of time series. Until now, these traditional methods have occupied a very important position in the field of time series. In 2009, Lee [5] developed a trend prediction method by combining support vector machine (SVM)

model with mixed feature selection method. In 2017, Markovic et al. [6] proposed a hybrid method for stock trend prediction using AHP and weighted kernel LS-SVM. In 2019, Ozorhan et al. [7] used improved zigzag technical indicators to find the themes of stock trends and classified the themes by SVM. Although some traditional methods can achieve good results in time series prediction, the principal structure of these methods leads to a bottleneck. The potential information of time series data can not be learnt in completed by these previous methods The emergence of neural networks provides a solution to the problem of learning completion. In fact, neural networks have powerful online learning ability. It can learn its potential laws through multiple iterations according to the data itself

In the 1990s, Back-Propagation (BP) neural networks [8] began to be applied to practical problems on a large scale, and a large number of scholars used BP neural networks to predict time series. Although BP neural networks can be used for time series prediction, its internal structure does not effectively utilize the time characteristics of time series. At this time, cyclic neural networks, which had been silent for a long time, came into people's eyes again because of the appearance of LSTM. LSTM changes the internal structure of a cyclic neural network and makes it possible to solve its problem of long-term dependence. In this way, the poor prediction effect of traditional methods for the time series with a long time span can be improved. In 2016, Di Persio Luca and Honchar Oleksandr [9] applied an artificial neural network and a LSTM network on stock market data, and found out that the experimental results with a LSTM network were far better than those of conventional artificial neural networks. Sezer and Ozbayoglu [10] converted stock technical indicators into two-dimensional images and proposed a new algorithm trading model based on CNN to predict stock trends. Sezer and Ozbayoglu [11] proposed a two-dimensional CNN that only used two-dimensional stock bar graph images to predict stock trends. [12] introduced a novel method for sequential prediction based on Auto-Regressive Integrated Moving Average (ARIMA) and Artificial Neural Network (ANN) models. Hamzaebi et al. [13] proposed two multi-period prediction methods based on artificial neural network. The first one is an iterative method that uses past observations to predict subsequent periodic information. The second method is a forecasting method in which subsequent periods can be estimated at once. The author of [14] come out

a method of Evolutionary Partial Connection Neural Network (EPCNN) for stock prediction. This model makes random connections between neurons, has multiple hidden layers, and uses evolutionary algorithms to train the artificial neural network. It is often said that the stock market is unpredictable, and these studies provide the possibility for stock prediction.

In terms of the application of other mathematical models. In 2017, [15] Kocak and Cem proposed a time series prediction method of high-order ARMA model based on fuzzy logic relations, which proved that the improved ARMA model can still be used as the main research method for time series analysis. Guoliang proposed the concept of kernel feature , the feature of each variable from multivariable time series was taken as the kernel feature, the concept of measure was extended, Precision, Recall and Earliness were taken as the evaluation criteria of feature performance, the kernel feature was extracted and selected, and the classifier was constructed based on the kernel feature to predict the multivariable time series in advance. In this paper [16], the fuzzy time series is combined with granularity calculation method, and the numerical results obtained by this model are better than those obtained by Support Vector Regression (SVR), fuzzy GARCH time series and mixed fuzzy time series. The paper [17] proposed a new kernel function of support vector regression for stock market prediction. The authors of the paper used support vector machines as a predictive method to predict 12 different stock indices.Senapati et al. [18] proposed a hybrid method for stock forecasting. In this hybrid method, the weight update of Adaline neural network is controlled by [19]. Performance comparison of several schemes on the TICK and 15-min datasets of an Indian company. The comparison concluded that the TICK dataset had better accuracy than the 15-min dataset.

The methods mentioned in the above literatures have not been discussed in the unbalanced time series data. Unbalances data means some clusters(classes) containing a small number of data points and some others containing a large number of data points in a data set. Without considering data preprocessing to balance unbalanced data sets, it is difficult to develop effective models using the current traditional data mining and machine learning algorithms. For most data mining algorithms, rare objects are more difficult to identify than ordinary objects, because the rare objects easy to be submerging in ordinary objects. This is a problem encountered in many prac-

tical applications, such as medical diagnosis, financial crisis prediction, and E-mail filtering. In addition, the concerns in data mining tasks are usually a few categories. Moreover, the high dimension of time series data makes it more difficult to study unbalanced time series data.

For treading the unbalanced data, there are some general approaches. The threshold approach involves setting different thresholds for different clusters (classes) in the learning stage. The one-class learning approach needs to train the classifier from a training set containing only one specific class. Under-sampling and oversampling approaches are used in classification problem. Other types of approaches, such as evolutionary clustering in neuro-fuzzy systems, evolutionary clustering of dynamic data in peaking neural networks, personalized modeling clustering, and clustering via quantum-heuristic evolutionary algorithms, have also been developed to deal with unbalanced data. In view of the problems of stock time series unbalanced data, we develop a novel method based on threshold approach for avoiding large cluster erode small cluster.

1.3 Issues in this thesis and our solutions

This thesis will carry out approach in the following aspects: through clustering model to find out the trend pattern of historical stock price sequence data, converting the original time series to into simplified categorical trend pattern series, and then predicting the future trend of stock price based on categorical pattern series. We will discuss some issues as following.

The first is measure the similarity between time series. In order to cluster time series, the first problem we have to faced is how to compute similarity between time series. An effect similarity measure is the foundation of quantitative clustering. Some general measuring the similarity on temporal or spatial similarity. European distance, cosine similarity, Pearson correlation coefficient are some frequent used methods to measure the similarity in spatial. In the research, it is important to consider correlation between the two sequences as much as possible. Temporal measure similarity, dynamic time warping (DTW) (see section 2.2), can adept this demand more. The idea of DTW is coming from dynamic programming. It matches between two numeric

sequence segments by linearly stretching and compressing the sequence.

The second issue is mining the trend pattern form sequence segments. Because the pattern is hide in the original sequence and unlabeled, clustering model adepts the task. Graph convolutional network (GCN) (see section 2.3) clustering is a link-based convolutional networks, it can represent the relationship between sequence segments well. It appears often in literatures. For GCN clustering, we need to learn an adjacency matrix to represent the similarity between two data nodes. To this end, we map the complex relations of stock segments to data which has topological structure, then use the GCN method to cluster those stock segments based on their topological structure. Constructing this matrix, i.e. determining whether a connection should be established between two data nodes, is critical to clustering. We try to do clustering based on the adjacency matrix.

The third issue is dealing with unbalanced cluster problem. Traditional Gaussian kernel function graph leads to too many links between data points (sequence segments), so that the final clustering results tend to form large clusters, the smaller clusters are corroded. That is the unbalanced cluster problem. To avoid the effect, two thresholds is involved in. One of them is used as a cap on weight beyond. Another is to limit number of links to a vertex. The two parameters restrict too small and too large cluster appearing. The determination of the two threshold is employed S-Dbw (see section 2.5) index.

Based on above discuss, the proposed method for clustering and forecasting stock trends is developed. In this method, we use wavelet transform for noise reduction. To measure the similarity between time series, we use DTW. GCN algorithm is used clustering the time series fragments. Based on the clustering results, the time series fragments are assigning their cluster labels, one cluster label corresponding one trend pattern. So that, based on the clustering processing, we transform the original time service to a simple and short trend pattern service. Last, LSTM(see section 2.4) is used as the prediction algorithm. To deal with unbalanced data, we improve the graph building part of GCN algorithm, which makes the algorithm more suitable for unbalanced data processing.

Chapter 2

Background Knowledge

This chapter will briefly introduce research in the area of time series prediction, and explain the mathematical principles used in our study, including Discrete wavelet transform (DWT), Dynamic time warping (DTW), GCN and LSTM, etc.

2.1 DWT method used for noise reduction

Wavelet transform is a method of analyzing signals. Wavelet transform has been developed from the method of Fourier transform. Fourier transform is also a method of analyzing signals, it transforms a function of position space into frequency space. It can analyze the components of signals and use these components to synthesize signals. Many waves can be used as signal components, such as sine wave, square wave, etc., Fourier transform uses sine wave as signal components. The equation is:

$$F(w) = F(f(t)) = \int f(t)e^{-iwt}dt, \tag{2.1}$$

where $f(t)$ is the antiderivative. The essence of the Fourier transform is the inner product. The inner product of trigonometric functions is a complete set of orthogonal functions, where the inner product of trigonometric functions of different frequencies is zero, and only the inner product of trigonometric functions of the same frequency

is not zero [20].

Fourier transformation can be interpreted as the projection of $F(t)$ on e^{-iwt}, and the integral value is the integral from minus infinity to infinity of time, when the frequency positioning is good, the frequency component contained in the signal can be clearly obtained through the good frequency resolution of the signal. Although Fourier transformation itself can explain a lot of phenomena and solve a lot of problems, it still has some limitations. The Fourier transformation extracts all the spectrum information of the signal in the time domain, but it does not reflect the change of the frequency component of the signal with the change of time, and the processing effect of the Fourier transform on the mutation signal is not good. Wavelet transform has been developed to solve these problems.

Wavelet analysis has been developed on the basis of Fourier analysis, but there are many differences between wavelet analysis and Fourier analysis. Compared with Fourier transform, wavelet transform is the local transform of space (time) and frequency, so it can extract information from signals effectively. By means of operation functions such as stretching and shifting, functions or signals can be analyzed at multiple scales, and many difficult problems that cannot be solved by Fourier transform can be solved. In other words, wavelet transformation has a low time resolution and a high frequency resolution in the low frequency part, and a high time resolution and a low frequency resolution in the high frequency part. The wavelet transformation analyze the non-stationary signals and extract the local features of the signals, so the wavelet transform is known as the microscope of signal analysis and processing. In the signal processing and analysis, the wavelet transform has the self-adaptability to the signal, and is also a signal processing method superior to Fourier transform and window Fourier transform. The meaning of wavelet transform refers to the inner product of a function called the basic wavelet after displacement a distance τ and the signal $x(t)$ to be analyzed at different scales α,

$$WT_x(\alpha, \tau) = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{+\infty} x(t)\varphi(\frac{t-\tau}{\alpha})dt, \tag{2.2}$$

where $\alpha > 0$, and become scale factor. Its function is to stretch the basic wavelet φ(t)

function, τ represents the bias, its value could be positive or negative, $x(t)$ represents the pre-processing function [21].

There are two types of wavelet transform, one is continuous wavelet transform, and the other is discrete wavelet transform. Here we are going to use the discrete wavelet transform method to reduce the noise of data. The reversible problem of the discrete wavelet transform implies that the expression of the discrete wavelet transform can completely store all the information of the signal to be analyzed, which requires the support of the mathematical framework. If the framework condition is satisfied for all the signals to be analyzed, then DWT is reversible. Discrete wavelet transform actually means discrete wavelet sampling wavelet transform, as a result of the discrete wavelet transform with variable window size, so using discrete wavelet transform instead of Fourier transform is very important. It can be zoomed in and save the computing time compared with Fourier transform, discrete wavelet transform is much longer, therefore, its completion speed is very fast, so the workload of discrete wavelet transform is very small [22].

From the point of view of signal processing, wavelet denoising is just a problem of signal filtering. In figure 2.1, the signal with noise is extracted from the characteristic signal and passed through a low-pass filter, and the low-pass filter is finally reconstituted with the characteristic signal. Although wavelet denoising can be regarded as low-pass filtering to a large extent, it is superior to the traditional low-pass filter in this respect because the signal features can be retained successfully after denoising. Thus it can be seen that wavelet denoising is actually a combination of feature extraction and low-pass filtering, and its flow diagram is shown:

2.2 DTW method used for similarity measuring

Dynamic time warping (DTW) [23] is a time series distance measurement method employed in many fields and has proved to be effective in real applications. Even though the computation time of the dynamic time warping algorithm is much longer than the computation time of the Euclidean distance, the computation time of the dynamic time warping algorithm can reflect the actual situation better than that of the Euclidean distance in some specific calculation scenarios. In this research,

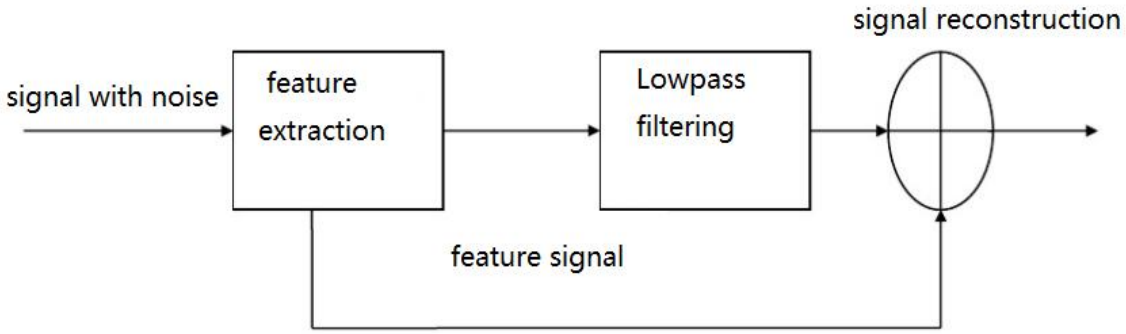

Figure 2.1 – Wavelet Transform noise reducing processing

we expect to use DTW method to measure the distance between two time series to represent their similarity, so as to provide reference for clustering steps [24].

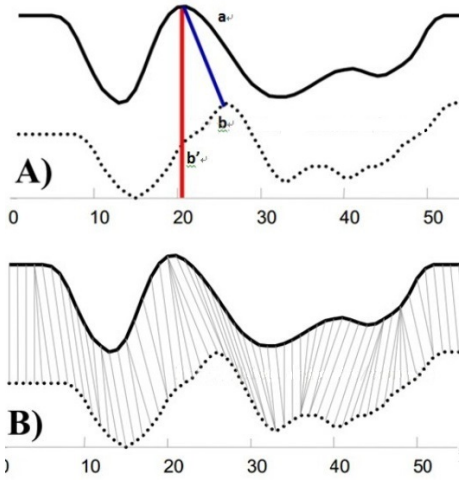

Figure 2.2 – Different comparison methods for two time series

Time series is a very common way of data existence, and in most data mining work, it is a common task to calculate the similarity between time series. However, in practice, time series used for similarity calculation are usually similar on the time axis, but the specific corresponding relationship is unknown. As shown in figure 2.2.A, point **a** should be more similar to point b in shape, rather than **b** at the same time. If the traditional Euclidean distance is calculated, the dynamic change in time will

not be taken into account, which will obviously cause a great error. In fact, most of times, the two sequences have very similar shapes overall, but those shapes do not align along the X-axis. So before we can compare their similarity, we need to warping one or both of the sequences under the timeline to get better alignment. DTW is an effective way to achieve this warping distortion. DTW calculates the similarity between two time series by extending and shortening the time series [25].

Computing dynamic time warping is a typical optimization problem. It uses a time warping function $W(n)$ satisfying certain conditions to describe the time correspondence between the test template and the reference template, and solves the warping function corresponding to the minimum cumulative distance between the two templates when matching. Since all time series have the same length in this research, suppose we have two time series Q and C, both of which have length n:

$$\begin{cases} Q = [q_1, q_2 ... q_n] \\ C = [c_1, c_2 ... c_n] \end{cases}, \tag{2.3}$$

We need to construct a matrix grid of $n * m$, with the matrix elements $(i,\ j)$ representing the distance $\mathrm{d}(q_i,\ c_j)$ between points q_i and c_j (that is, the similarity between each point of sequence Q and each point of C, the smaller the distance, the higher the similarity. Regardless of the order here), the Euclidean distance is generally used, $d(q_i, c_j) = (q_i - c_j)^2$ (can also be understood as distortion). Each matrix element $(i,\ j)$ represents the alignment of points q_i and c_j. Dynamic programmin (DP) algorithm can then be used to find a path through a number of grid points in the grid, the path through the grid points are the alignment of the two sequence calculation points [26].

This path is defined as a regular path and is represented by W. The kth element of W is defined as$w_k = (i, j)_k$, and the mapping of sequence Q and C is defined like:

$$W = [w_1, w_2, ... w_k] \quad max(m, n) \leq K \leq 2n - 1, \tag{2.4}$$

1) Boundary conditions: w_1 =(1, 1) and $w_k = (n, n)$. The order of the parts of

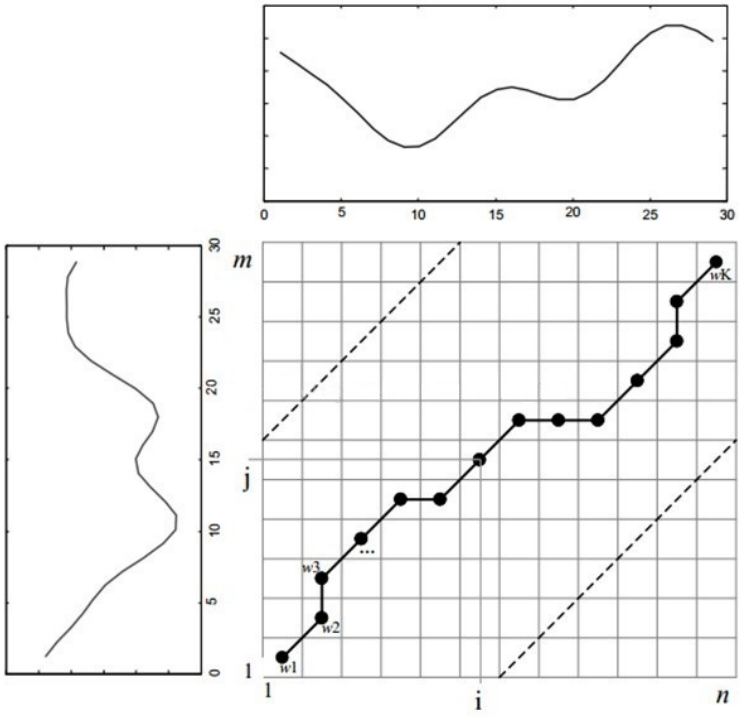

Figure 2.3 – Distance matrix of two time series

the sequence cannot be changed, so the chosen path must start at the bottom left corner and end at the top right.

2) Continuity: if $w_{k-1} = (a', b')$, then for the next point of the path $w_k = (a, b)$ a need to meet $(a - a') <= 1$ and $(b - b') <= 1$. This means each point can only match with its own adjacent points. This ensures that every coordinate in Q and C appears in W.

3) Monotonously: if $w_{k-1} = (a, b')$, then for the next point of the path $w_k = (a, b)$ need to meet $a(a - a') >= 0$ and $(b - b') >= 0$. This restricts the points above W to be monotonically progressive over time.

Combined with the continuity and monotony constraints, each lattice has only three paths. For example, if the path has already passed through the grid $(i,\ j)$, then the next grid point to pass through can only be one of three things :$(i + 1,\ j)$, $(i, j + 1)$, or $(i + 1, j + 1)$.

There can be many paths that satisfy these constraints, and then we are interested

in the path that minimizes the cost of normalization. In order to find the path with the minimal cost of structuring, define a cumulative distance. Match the two sequences Q and C starting at (0, 0), and at each point, the distances calculated at all previous points add up. After arriving at the end point (n, m), the cumulative distance is the total distance what we said above, the similarity between the sequence Q and C, represented by $\gamma(i, j)$ [27]

$$\gamma(i,j) = d(q_i, c_j) + min[\gamma(i-1, j-1), \gamma(i, j-1), \gamma(i-1, j)], \tag{2.5}$$

where $d(i, j)$ is the distance of the current grid point. The best path is the one that minimizes the accumulated distance along the path.

2.3 The Graph Convolutional Network(GCN) used for clustering

In this research, we make use of GCN method to cluster time series and find patterns in different time series. Convolution is an important operation in analytical mathematics, set $f(x)$ and $g(x)$ as two integrable functions on R, the continuous formal convolution of $f(x)$ and $g(x)$ is defined as:

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau, \tag{2.6}$$

The same for discrete convolution, for the signal X and the convolution kernel W, the result of the convolution is to multiply the signal and the convolution kernel at different locations, and then take the sum of their results [28]:

$$y_n = x * w = \sum_{k=1}^{K} w_k x_{n-k}. \tag{2.7}$$

The essence of discrete convolution is to calculate the sum of a group of weighted

numbers. CNN is a Neural Network that uses the convolution calculation. CNN adapts a parameter sharing kernel, and calculates the weighted sum of the center note and adjacent notes form a feature map to realize spatial feature extraction. But CNN could only deal with the data in Euclidean structure, which is easy to find the neighbors in each data point. If CNN face the data in non-Euclidean structure like social network or a graph of the structure of protein molecule, it will not be effective. This is the reason why we need to come out GCN.

In appropriate conditions, the Fourier transform of the convolutional of signals equals the product of their Fourier transform. For example, the convolutional of a domain (e.g time domain) equals to the product of an other domain (e.g frequency domain):

$$F\{f * g\} = F\{f\} * F\{g\}, \tag{2.8}$$

If we use F^{-1} to represent the inverse transformation of the Fourier transform, the convolutional can be expressed as:

$$f * g = F^{-1}\{F\{f\} * F\{g\}\}, \tag{2.9}$$

The Fourier transform transforms a function in the time domain into a function in the frequency domain, It is a different perspective of the same function(see equation (2.1)). Equation (2.1) is the integral the product of function $f(t)$ and e^{-iwt}, e^{-iwt} is the characteristic function of Laplace operator. That means the fourier transform is the integral of the product of characteristic function of Laplace operator and time domain. When discuss on discrete structure like graph, the fourier transform is sum of the product of characteristic function of Laplace operator and time domain [29].

The Fourier transform can also be expressed as the integral of the time domain signal with the eigenfunction of the Laplace operator. For the Fourier transform of the graph we need to find the Laplace operator of the graph and its eigenfunctions. After research, scholars found that the Laplace matrix and its eigenvector can be used as a substitute for the two in Fourier transform.

Based on the formula of the Fourier transform, we transform the characteristic function into eigenvector, and transform integral into sum, the formula of Fourier

transform on the graph with N vertex is like:

$$F(\lambda_l) = \widehat{f}(\lambda_l) = \sum_{i=1}^{N} f(i) * u_l(i), \tag{2.10}$$

where λ_l is the lth eigenvalue of Laplace operator on graph, u_l is its lth eigenvector. It could write into matrix way: $\widehat{f} = U^T f$, where U is Laplace Orthogonal Matrix. Laplace Matrix $L = D - A$, where A is the adjacency matrix of graph, and D is the degree matrix of graph. So Laplace Matrix is real symmetric matrix, its orthogonal similarity diagonalization is $L = Udiag[\lambda_1, ...\lambda_n]U^T$. Based on the convolutional theorem and the formula of Fourier transform on graph, we can get an equation like that:

$$(f * h)_G = F^{-1}[F\{\} * F\{h\}] = F^{-1}[U^T f * \widehat{h}], \tag{2.11}$$

Let the convolutional of the f signal and h signal transform into Fourier domain, then do product, and then do inverse transform operation, where the product of the vector f and the vector of $\widehat{h}$ will equals to the multiplication with $\widehat{h}$ in diagonal matrix:

$$(f * h)_G = F^{-1}[U^T f\widehat{h}] = F^{-1}[diag[\widehat{h}_1, ..., \widehat{h}_n]U^T f], \tag{2.12}$$

Based on the inverse transform calculate form in graph, we can transform the last formula into an equation express by matrix U.

$$(f * h)_G = Udiag[\widehat{h}_1, ...\widehat{h}_n]U^T f, \tag{2.13}$$

We can set $diag[\widehat{h}_1, ...\widehat{h}_n] = diag[\theta_1, ...\theta_n] = g_\theta$ [30].

Base on the last two formulas, we can get:

$$y = \sigma(Ug_\theta U^T x) = \sigma(Udiag[\theta_1, ...\theta_n]U^T x), \tag{2.14}$$

Following on the first generation of GCN, let g_θ transform the eigenvalue function of

the Laplace matrix L into $g_\theta(\Lambda)$:

$$y = \sigma(U g_\theta U^T x) = \sigma(U g_\theta(\Lambda) U^T x), \tag{2.15}$$

However, taking the eigenvalue of Laplace matrix as the convolution kernel has some drawbacks: it does not have local connectivity and has too high time complexity. Therefor, the k-order polynomial is introduced:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k \Lambda^k, \tag{2.16}$$

Where θ_k is the coefficient of polynomial, because of the k-order locality in filter, the time complexity reduce to $O(K)$.

Substitute the last formula into the first generation of GCN, we can get:

$$y = \sigma(U g_\theta(\Lambda) U^T x) = \sigma(U \sum_{k=0}^{K-1} \theta_k \Lambda^k U x) = \sigma(U \sum_{k=0}^{K-1} \theta_k L^k x), \tag{2.17}$$

Because L_k is not a sparse matrix when K is very large, it has a high time complexity, while using Chebyshev polynomial expansion on L_k can get a result similar to itself.

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \ , \\ T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \end{cases} \tag{2.18}$$

For the reason of the input of Chebyshev polynomial is in the section of [-1,1], therefore the Laplace matrix is positive semi-definite matrix, so all of the eigenvalues are positive or zero. Divide all of the eigenvalues with the maximum value to normalize the eigenvalue into the section of [0,1], then transform the values into the section of [-1,1] , so we transform λ into $\widehat{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$; λ_{max} is the maximum eigenvalue in Laplace matrix L [31]. Substitute the last formula into $g_\theta(\Lambda)$:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda}), \tag{2.19}$$

2.3. The Graph Convolutional Network(GCN) used for clustering

Substitute Chebyshev polynomial into the Convolutional transformation:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})x, \tag{2.20}$$

where $\widetilde{L} = \frac{2}{\lambda_{max}} L - I_N$.

GCN is superimposed by multi-layer convolution layer through the above formula, and each layer is superimposed point by point. In terms of time complexity, let's say K is equal to 2, in other words, we have a second order approximation of the Laplace operator. We can do the convolution operation on the network and it doesn't add much computation. The nonlinearity of the model can be improved by the number of layers.

The eigenvalue section of the normalized Laplace matrix is [0, 2] , set $\lambda_{max} \approx 2$, substitute $K = 2$ into the Chebyshev polynomial in the second generation GCN:

$$g_\theta * x \approx \theta_0 x + \theta_1(L - I_N) = \theta_0 x - \theta_1 D^{-1/2} A D^{-1/2}, \tag{2.21}$$

In the process of GCN parameter training, it is necessary to normalize parameters to avoid overfitting, set $\theta = \theta_0' = -\theta_1'$, we can get.

$$g_\theta * x \approx \theta(I_N + D^{-1/2} A D^{-1/2})x, \tag{2.22}$$

The eigenvalue of $I_N + \widetilde{D}^{-1/2} A \widetilde{D}^{-1/2}$ is in range of [0, 2]. In order to avoid gradient explosion, it needs to be normalized again, transform $I_N + \widetilde{D}^{-1/2} A \widetilde{D}^{-1/2}$ into $\widetilde{D}^{-1/2} A \widetilde{D}^{-1/2}$. Finally we can get, Z = $\widetilde{D}^{-1/2} A \widetilde{D}^{-1/2}$.

Then, we could find that the essence of GCN is a neural network. Here is its rule of propagation:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2} H^{(l)} W^{(l)}), \tag{2.23}$$

where $\widetilde{A} = A + I_N$, A is the adjacency matrix of the graph, I_N is identity matrix of graph; $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, $\widetilde{D}$ is the degree matrix of the adjacency matrix; $W^{(l)}$ is the weight matrix in the l^{th} layer in the neural network; $H^{(l)}$ is the activate matrix in

the l^{th} layer in the neural network, when $l = 0$, $H^{(l)} = X$, X is the matrix input; $\sigma()$ is the activation function. In our work, we adopt the ReLU function as the activation function [32]. We choose ReLU as the activation function because it can relieve vanishing gradients problems and reduce the negative influence of over-fitting problem on GCN.

2.4 The LSTM model for prediction

Long - short - term memory unit (LSTM) is a kind of time recursive network, which can achieve good results in many applications [33]. In this research, we choose to use the LSTM method to predict the possibility of future stock trend changes according to the pattern sequence.

In the field of deep learning, the problem of "long-term dependency" is common, and RNN is not sensitive to relatively remote information. The reason for long-term dependence is that after many stages of calculation of the nodes of the neural network, the characteristics of the previously long time slice have been covered. Under the influence of this mechanism, RNN loses the ability to learn information connected so far as time slices of data are increased [34].

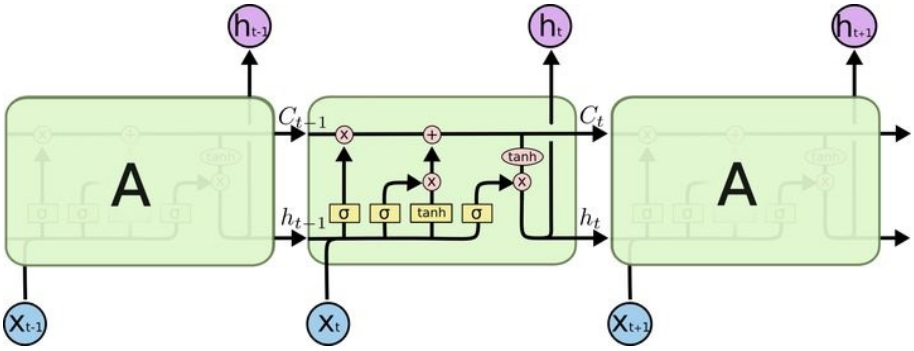

Figure 2.4 – Chain structure diagram of LSTM

As can be seen from figure 2.4, LSTM is composed of a series of LSTM units in a chain form, which we call cells in this paper. The use of LSTM is motivated to address the long-term dependency problem we mentioned above. LSTM can solve the long-term dependence problem because it introduces a gate mechanism to control the

flow and loss of intercellular characteristics. The key to LSTM is the update of cell status, and the information about the state of the cell is transmitted directly along the cell's sequence, with only a small number of linear interactions. It is easy for information to get around and stay the same. LSTM has the ability to remove or add information to the cellular state through elaborate structures called gates. A door is a way of letting information selectively through. They contain a Sigmoid neural network layer and a bitwise multiplication operation. The Sigmoid layer outputs a number between 0 and 1, describing how much of each part can pass through. To update the cell state in figure 2.4, use the equation:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t, \tag{2.24}$$

Here, the f_t is called the amnesia gate, indicating which features of C_{t-1} are used to calculate the C_t. f_t is a vector whose elements are in the range [0,1]. Usually we use sigmoid as the activation function, and the sigmoid output is a value in the range [0, 1], but when we look at a trained LSTM, we will find that most gate values are very close to 0 or 1, and the rest are very few and far between. The expression of the forgetting gate is:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f), \tag{2.25}$$

where, x_t is the current input vector, h_{t-1} is the current hidden layer vector, and h_{t-1} contains the output of all LSTM cells. The parameter b_f, and W_f are offset and weight respectively. In equation 2.24, $\widetilde{C}_t$ represents the cell status update value, the input data x_t and the hidden node h_{t-1} are obtained through a neural network layer. The activation function of the cell state update value usually uses the tanh function. It is called the input gate. Like f_t, it is also a vector with elements between [0,1], which is also calculated by x_t and h_{t-1} through the Sigmoid activation function. The parameter i_t is used to control the feature of $\widetilde{C}_t$ to update C_t expression the of it and expression are like:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i), \tag{2.26}$$

$$\widetilde{C}_t = tanh(W_C * [h_{t-1,x_t} + b_C]), \tag{2.27}$$

In equations 2.26 and 2.27, where, x_t is the current input vector, h_{t-1} is the current hidden layer vector, and h_{t-1} contains the output of all LSTM cells. B_i and W_i are bias and weight, respectively. Finally, in order to compute the predicted value $\widehat{y}_t$ and generate the complete input for the next time slice, we need to compute the output HT of the hidden node.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{2.28}$$

$$h_t = o_t * tanh(C_t), \tag{2.29}$$

h_t is obtained by output gate o_t and cell state C_t, where o_t is calculated in the same way as f_t and i_t.

2.5 S-Dbw validation index for measuring the effect of cluster

Evaluating the quality of the clustering results is an important issue. Clustering is an unsupervised classification procedure. There often lacks an objective criterion for evaluating the clustering results; they are assessed using a cluster validity index. The cluster validity index also plays an important role in determining the number of clusters. It is expected that the optimal value of the cluster validity index should be obtained at the true number of clusters. A general approach for determining the number of clusters is to select the optimal value of a certain cluster validity index.

A good clustering method can produce high-quality clusters with high similarity within clusters and low similarity between clusters. Generally speaking, there are two criteria for evaluating clustering quality, internal quality evaluation index and external evaluation index. The internal evaluation index usually evaluates the clustering quality by calculating the overall similarity and the average similarity within the cluster. The external evaluation index usually evaluates the clustering quality by calculating the overall similarity and the similarity between clusters. By considering both types of evaluation indexes, one obtains S-Dbw.

In fact, S-Dbw validation is considered to be the best validation index of unsupervised learning by many authors [35, 36]. S-Dbw is a density-based indicator that evaluates the effectiveness of clustering by comparing the compactness within clusters with the density between clusters [37].

S-Dbw consists of two parts:

$$S\_Dbw = Scat + Dens\_bw, \tag{2.30}$$

where $Scat$ represents the density within the cluster and $Dens - bw$ represents the density between clusters, which are defined:

$$Scat = \frac{1}{c}\sum_{i=1}^{c}\frac{\|\sigma(C_i)\|}{\|\sigma(D)\|}, \tag{2.31}$$

$$Dens\_bw = \frac{1}{c(c-1)}\sum_{i=1}^{c}(\sum_{j=1,i\neq j}^{c}\frac{density(u_{ij})}{max\left\{density(v_i), density(v_j)\right\}}), \tag{2.32}$$

In equation 2.31, D represents the whole data set, C represents the number of clusters, C_i represents the ith cluster, $\sigma(C_i)$ represents the variance of the ith cluster's sample. It's a vector of the variance of each dimension of the sample. $\|\sigma(C_i)\|$ is the L2 norm of $\sigma(C_i)$ vector. That's the Euclidean distance from the origin. $\sigma(D)$ is the data set D, that's the variance vector of all the samples. $\|\sigma(D)\|$ is the L2 norm of $\sigma(D)$ vector.

$$density(u) = \sum_{l=1}^{n_{i,j}} f(x_l, u), \tag{2.33}$$

where function f is to judge whether the distance between the sample point x_i and the center point is less than the average standard deviation of all clusters stdev. The equation is:

$$f(x_l, u) = \begin{cases} 0 \ if \ d(x_l, u) > stdev \\ 1 \ otherwise \end{cases}, \tag{2.34}$$

The Stdev equation is:

$$stdev = \frac{1}{c}\sqrt{\sum_{i=1}^{c} \|\sigma(v_i)\|}, \tag{2.35}$$

where *Stdev* represents the average deviation of each cluster of a dataset. This evaluation index also takes into account the compactness, which is the measurement of the sample distance in the cluster. And the degree of separation, which is a measure of the sample distance between clusters.

The clustering whose index reaches the minimum must be the optimal clustering, and the clustering result is independent of the algorithm. The smaller means a better clustering effect.

Chapter 3

Time Series Clustering and Prediction

In this chapter, we will present our method for clustering time series data in morphological. It is important in financial market to understand and predict the stock trend. Our aim is to mine the natural patterns in these serial data to provide the basis information for predicting future stock movements. We build a sub time series compare method based on the graph convolution model, mining the stock trend pattern by the unsupervised clustering model. We will introduce data preprocessing, sub-sequence similarity model based on graph convolution, sub-sequence clustering, clustering experiment and result evaluation, etc in this research. Figure 3.1 presents the framework of our project, GCN learning (i.e., adjacency matrix building), GCN clustering learning, and LSTM forecast.

3.1 Data Preprocessing

In order to remove unwanted information from the original data, strengthen useful information and reduce the complexity of the model, data preprocessing is required before data clustering. This includes noise reduction and data standardization. The denoising of data can help reduce the complexity and the over-fitting of the model, and minimize errors during the experiments. Data standardization helps scale the

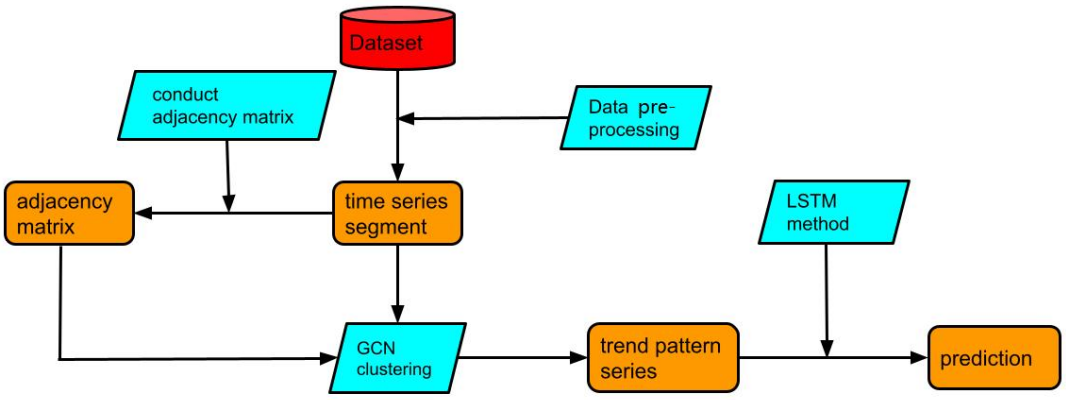

Figure 3.1 – Overview of the proposed method

data according to a certain proportion to make it fall into a specific area for a comprehensive analysis.

3.1.1 Noise Reduction

Since the original data of each stock has a lot of noise, which may have an important negative impact on the clustering process, it is necessary to reduce the noise in the data. Generally, noise can be divided into three categories: abnormal data, data that are not well interpretable, and data that do not follow the record form. Data with significant amount of noise will slow down the speed of our neural networks, reduce the interpretability of the training results, and produce the phenomenon of over-fitting, which will eventually lead to the low accuracy of the trained model.

There are usually several ideas for dealing with the noise issues, [38]. One is to detect outliers and treat them as noise; Another one is to set a threshold in advance, when a data point is found outside the threshold, treat it as noise and remove it from the training set. This latter method is mainly aimed at data points with abnormal presence status, which is easy to implement, but it has high requirements for the setting of threshold. Excessively wide thresholds make it difficult to find anomalies, while excessively narrow thresholds lead to exclusion of non-noise data

from the training set.

Feature based noise reduction consists in first extracting features of noise, then filtering the data, and finally, repackaging the filtered data and extracted features. This approach mainly deals with data that is not well interpretable. It aims to reduce the entropy of the data; moreover, the correlation between data points is reduced to make the algorithm more conducive to noise reduction in the time domain. The Fourier transform model and the discrete wavelet transform model are typical of this type of noise reduction methods. Given that few outliers appear in stock data, it is a higher proportion of noise that is relatively unexplained. Therefore, the feature based noise reduction will be used in this research. According to the previous context(Chapter 2), Discrete Fourier transform (DFT) and Discrete Wavelet Transform (DWT) are the two most commonly used noise reduction methods. They are related, but they are also very different. Compare with the DWT, the DFT saves global information but is available to deal with the local information, so DFT is not suitable for mining local information in time series, therefore the DWT is adopted as the noise reduction method.

Figure 3.2 shows the comparison of BAC stock data before and after noise reducing. The red line represents the line before noise reducing, and the black line represents the line after noise reducing. When DWT is used as the noise reduction method, the part of noise is obviously alleviated, and good results are achieved in both low frequency signal part and high frequency signal part. We can see that the noise reducing ability of wavelet transform in time series is very good. It can be seen that the noise reduced data (black line) more clearly reflects the trend of the stock.

3.1.2 Data Standardization

To avoiding a larger value will have a greater impact on clustering results. Before clustering, we need to standardize the data. In stock data, the maximum and minimum values of each time series are different, and the difference between the maximum and minimum values of each series is different. If it is not processed, the direct operation will bring a large error, affecting the clustering effect. Many morphological similar sequences will not be able to come together because of differences in horizon-

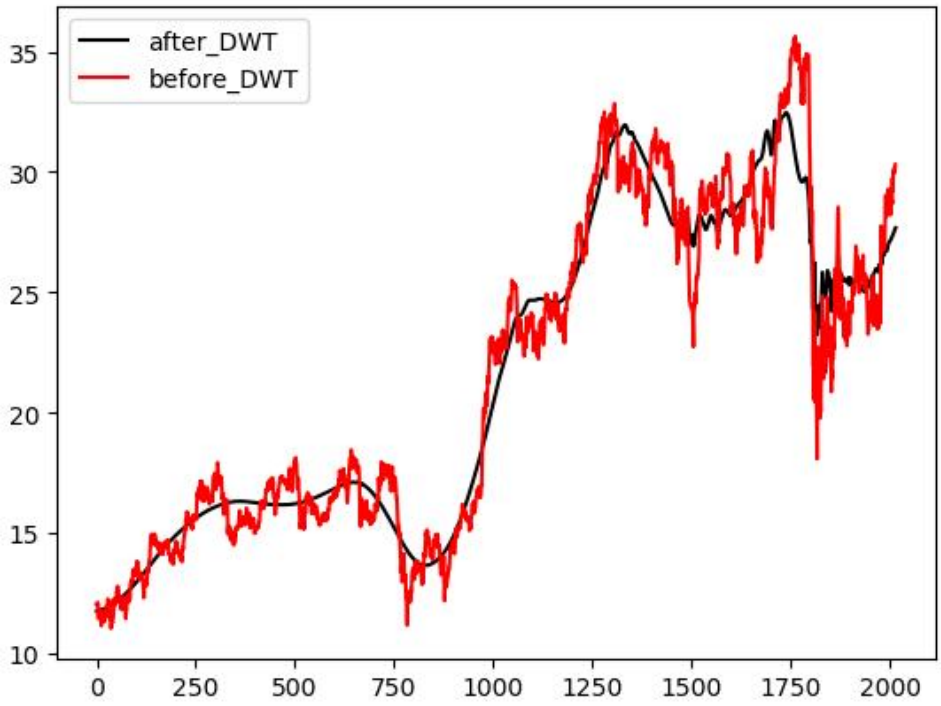

Figure 3.2 – BAC stock data comparison before and after noise reduction

tal position and increase. In this study, we further preprocessed the data using a standardized method. Convert all sequences to a sequence with mean 0 and standard deviation 1. The effect of scale differences between features is eliminated in order to treat each dimensional feature equally. Equation 3.1 is the process of standardization used in this research:

$$x^* = \frac{x - u}{\sigma}, \tag{3.1}$$

where u is average value of the time series, σ is the standard deviation value of the time series.

3.2 Measuring the Similarity of Time Series

In order to cluster time series, the key problem is how to define similarity. In this section we are going to discuss how to calculate the similarity between series and build the similarity matrix. Conventionally, researchers prefer measuring the similarity on temporal or spatial similarity. DTW is a common way to measure similarity on temporal; Euclidean distance, cosine similarity, Pearson correlation coefficient etc. are some frequent used methods to measure the similarity in spatial. In these choices, it is important to consider the characteristics of the time series data, both to identify the similarities in the shape, and to be able to reflect the correlation between the two sequences as much as possible.

Of the many ways to measure distance, the easiest is to use the Minkowski distance, i.e. L_n norm, defined as follows:

$$d_{Ln}(x, y) = (\sum_{i=1}^{M}(x_i - y_i)^n)^{\frac{1}{n}}, \tag{3.2}$$

where M is the length of vector, also called the lockstep measurement, that is, the measurement is compared according to the position of the elements, and when $n = 2$, that's what we call the Euclidean distance. For sequences of the same length, calculate the distance between each pair of points and sum them up. The smaller the distance, the higher the similarity. Although Euclidean distance is easy to understand and to realize, it has the following disadvantages: (1) it cannot distinguish the shape similarity; (2) it cannot reflect the similarity of dynamic variation amplitude of trend; (3) the calculation based on point distance cannot reflect the difference of different analysis frequencies.

The equation of cosine similarity is written as [39]:

$$cos < a, b >= \frac{a \cdot b}{|a| \cdot |b|}, \tag{3.3}$$

which a and b are all vectors. In the measurement of time series similarity, the sequence is regarded as a vector of N dimensions, and the similarity between two time series is measured by calculating the cosine of the included angle between two vectors.

The measurement method is easy to calculate and implement, and it focuses on comparing the directional differences between two vectors. However, it is insensitive to the two vectors whose absolute values differ greatly but their directions are similar. In addition, when measuring two time series, cosine similarity may produce a result of smaller similarity for two time series with similar shape but with phase difference, which is not applicable to time series data.

Under the Pearson correlation coefficient, assume there are two vectors X and Y, then the two vectors have the following coefficients according to Pearson's formula [40]:

$$r_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sigma_X \sigma_Y}, \tag{3.4}$$

where $E()$ is the mathematical expectation, σ is the standard deviation, Pearson provides a more complex way of estimating the similarity as compared to Euclidean distance. Its correlation coefficient is estimated by fitting two sets of data to a line, which is actually the slope of the line. Its slope interval is between [-1,1], and its absolute value reflects the similarity between the two data. The larger the slope is, the greater the similarity is. When the similarity is 1, the line is a diagonal line. Pearson correlation coefficient is actually an indicator to measure the degree of linear correlation between two variables, but its value cannot fully reflect the real relationship between two variables. In some cases, nonlinear correlation may also lead to large linear correlation coefficient.

As presented in Chapter 2, DTW is an algorithm to measure the distance between time series, which is evolved from the idea of dynamic programming. It locally stretches and compresses the sequence data linearly to find the best match between two numeric sequence segments, and also calculates the distance between these sequence segments.

Suppose there are two time series x, y, length m and n respectively, a m by n matrix D, $f(x_i, y_j)$ representing the distance between the ith node of x and the jth node of y. $D_{i,j}$ is defined as follows:

$$D_{i,j} = \begin{cases} f(x_i, y_j) + min\left\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\right\} \ if\ i \neq 0\ and\ j \neq 0 \\ 0\ if\ i = 0\ and\ j = 0\,, \\ +\infty\ otherwise \end{cases} \tag{3.5}$$

After calculation, $D_{m,n}$ is the distance of two series. Even if there is a time shift between one signal and the other, DTW still allows the system to compare two signals and search for similarities [41].

In this work, the purpose of employing DTW is not only to obtain the distance between the two time series, it is also to find the corresponding relationship between the two time series, so that the two series can be better compared. Although the direct implementation of DTW takes more time than the calculation of Euclidean distance or Pearson coefficient, the dynamic programming technique used in DTW makes its calculation almost linear in terms of time series length.

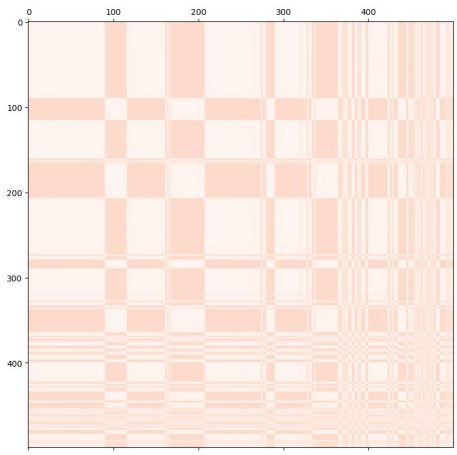

Figure 3.3 – Similarity matrix obtained by DTW method reduction

In order to verify the validity of DTW, we use the daily closing price data of BAC stock from 2013 to 2020, and regard 10 days as a window width to generate 500 sequence segments (sequence data points) in total, and calculate the DTW distance

matrix between all 500 sequence data points, as shows in Figure 3.3. In this matrix, the larger the distance is, the dummier the corresponding element of the matrix is.

3.3 Building the Adjacency Matrix for Dealing With Unbalanced Clusters in GCN

Traditional time series clustering methods need to make some assumptions about data distribution, and some of them are unrealistic [42]. For example, K-means needs to assume that different clusters have roughly the same density. Linkage-based clustering methods (i.e., GCN) can achieve higher accuracy without making assumptions about data distribution. Due to the complexity of stock market, it is more favorable to use GCN method since it can capture features of stock market by studying data which has topological structure.

Traditional GCN establishes weight matrix as follows. Assume that the graph $G = (V, E, W)$, where V is vertex set and $|V| = N$, E is the set of edge, W is the weight matrix. If there is edge $E(i, j)$ connecting node i and node j, then there is a non zero weight $W_{i,j} \neq 0$, otherwise $W_{i,j} = 0$. If graph G is disconnected and has M connected components ($M > 1$), we can divide the signal on G into M pieces (subgraphs) corresponding to M connected components, and process the separated signals on each subgraph independently [43]. The weights of the connecting edges of vertices i and j are defined by the threshold Gaussian kernel weight function:

$$w_{i,j} = \begin{cases} exp(-\frac{d_{i,j}^2}{\sigma^2}) & i \neq j \ and \ exp(-\frac{d_{i,j}^2}{\sigma^2}) \geq \varepsilon \\ 0 & otherwise \end{cases}, \tag{3.6}$$

In Equation 3.6, $d_{i,j}$ can represent the physical distance between vertex i and j, and can also represent the Euclidean distance between the two eigenvectors describing i and j, σ and ε are the thresholds controlling the size of $W_{i,j}$. This method is especially common in graph-based semi-supervised learning methods. When the distance between two data points is small enough, it means there is a connection between them,

and the weight of the connected edge is $W_{i,j}$. This method is easy to implement and it provides a representative matrix to represent the relationship between data points for data sets that do not have adjacency matrix.

Traditional Gaussian kernel function graph could encounter difficulties in the presence of unbalanced clusters. If the weighted graph formed by the primitive Gaussian kernel function graph construction algorithm [44] is directly used to establish the connection matrix, it will lead to too many links between sequence segments. Consequently, the final clustering results tend to form large clusters, while smaller clusters will be eroded. This is the unbalanced cluster, or "Matthew effect" that will happen in the clustering process.

As an example, suppose that A, A' and B express segments from three sequences (data points). A has a larger number of segments are with a close distance (meaning that A belongs to a larger cluster), segment A' is one them. B has less segments are close to itself (means that B should belong to a small cluster), but A' is also close to B in distance. A' might belong to the same cluster with segment A or segment B. If A is far to B, A and B should not belongs to the same cluster. but because of segment A', A and B might clustering into the one cluster. The other segments which belongs to the same cluster with B will share the same cluster with A. If we ignore this kind of phenomenon, we might only have one cluster as the result.

We improve the traditional Gaussian kernel composition method by using DTW. Here the distance matrix calculated by pairwise distance between time series segments is set as D, where $D_{i,j}$ is the DTW distance between the current segments of time series i and j. Our method employs two threshold parameters whose values are set via model selection. The threshold α is used as a cap on $D_{i,j}$ beyond which no link is allowed between the segments of time series i and j. We also employ an upper limit γ for the maximal number of links to a vertex. Let $N_i(\gamma)$ represent the γ nearest vertices (neighbors) of vertex i. The adjacency matrix $A = (A_{i,j})$ of the graph is defined as follows:

$$A_{ij} = \begin{cases} 1 & D_{ij} \leq \alpha, j \in N_i(\gamma) \\ 0 & otherwise \end{cases}. \tag{3.7}$$

The value of α and γ in our method is application-dependent. To determine their

value, we use the S-Dbw index to measure the validity of choices of values for α and γ on the clustering results. S-Dbw is an index that can measure the clustering validity of a clustering algorithm. S-Dbw can fully reflect the degree of separation and the compactness of the clusters. The smaller the S-Dbw index value, the better the clustering results. We should therefore choose the combination of parameter values with the smallest S-Dbw. The corresponding adjacency matrix established for 1500 from a part of our data set is shown in Figure 3.4.

In this figure, time series segments with joined edges are colored, and two segments without joined edges are not colored. The diagonal lines of the matrix are also colored for easy observation. Therefore, it can be observed that the adjacency matrix is a sparse matrix. The computation time will be reduced accordingly, the computation is more efficient. In stock data, segments with closer position in matrix are easier to be grouped into a cluster with the segments themselves. If the segments in closer positions in adjacency matrix can have a greater impact, the clustering effect will become better and the interpretability of clustering will be further improved. Through local observation, it can be found that the elements of adjacency matrix are more likely to gather at the diagonal position, which indicates that the similarity of segments with similar positions is greater, and it is easier to establish edges and form connections. The similar points in each segment can have a greater impact on its clustering process. It has a favorable effect on the clustering of stock time series.

3.4 Clustering Based on GCN

According to the previous content(see section 3.1), we will get a sequence data that can represent the trend of stock price after normalization, denoising and slicing. We hope to get the probability of the cluster to which each sequence belongs through the algorithm of graph accumulation cluster.

The essence of GCN is several layers of neural networks. By iteration, the neural networks can be trained to implement an adaptive system for analyzing time series. In this thesis, the GCN neural network used is formulated as below:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)}), \tag{3.8}$$

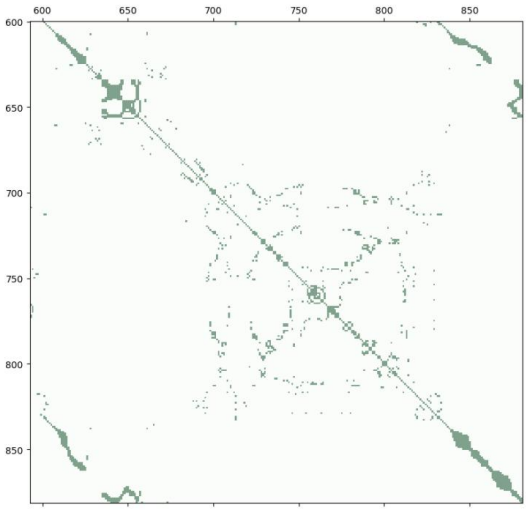

Figure 3.4 – Local of the adjacency matrix

where $H^{(l)}$ is the lth layer output value, when l equals to 0, the input is the preprocessed data for the graph convolution network; $\widetilde{A} = A + I$, where A is the adjacency matrix, I is identity matrix; $\widetilde{D}$ is the degree matrix of $\widetilde{A}$, the equation is showed as $\widetilde{D}_{i,j} = \sum_j \widetilde{A}_{i,j}$; σ is the nonlinear activation function, in this research, we use softmax function as the activation function; $W^{(l)}$ is the lth layer parameters of graph convolution network.

In this research, an iterative method is adopted to optimize and cluster the sequence segments. We use information entropy as the objective function because the output results are the probabilities of each category. The information entropy function formula is as follows:

$$H(x) = -\sum\sum X_{ij} log X_{ij}, \tag{3.9}$$

where X is the output of the GCN. According to the equation 3.9, we can found that while the confidence of of clustering become higher(the probability concentrate on one category in most of segments), the object function value become lower. And compare with the cross entropy function, the information function doesn't need label data. Therefore, the information entropy function will be used as the objective function of

the clustering algorithm.

In addition, Adam(ADAM)'s method will be used to optimize the GCN algorithm. Optimization algorithm of stochastic objective function based on low - order moment adaptive estimation is presented [45]. Adam combines the advantages of ADAGRAD and RMSProp, and after bias correction, each iteration learning rate has a certain range, which makes the parameters relatively stable. That decrease the epoch of study, and make the study process more effective.

In this research, the ADAM optimization algorithm was used to carry out 100 iterations on the cross-entropy objective function, as shown in Figure 3.5.With the increase of iteration times, the cross-entropy objective function gradually decreases, which means that the maximum probability of sequence segment ownership increases gradually, and the difference between them and other clusters will be bigger.

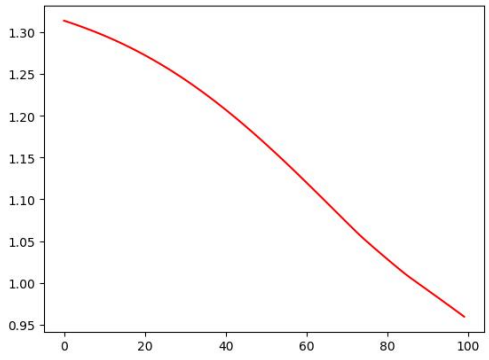

Figure 3.5 – Variation of the value of the objective function with the number of iterations

3.5 Prediction Based on LSTM

The ultimate goal of financial time series analysis is to provide reference for decision makers by forecasting the future values, short-term or long-term. As described in Chapter 2, LSTM is suitable for processing and predicting important events with relatively long intervals and delays in time series, such as learning to translate languages, controlling chatbots, predicting diseases, click-through rates, and stocks. Among the

advantages of LSTM algorithm, we have its high accuracy, strong parallel and distributed processing ability, distributed storage and learning ability. Moreover LSTM is robust against noise and is highly effective to learn complex nonlinear relations.

For prediction of time series, selection of time step has always been a challenge and needs to be addressed. How long will the future information relates the previous data is unknown to us. For the structure of the cyclic neural network, the time step determines the number of learning features of the network model. Theoretically, the more features are learned, the better the prediction accuracy of the model will be. However, in practice, irrelevant information in history would also stick into the training, which will affect the final prediction result. In this research, we take the daily stock closing price as the main research object, so the data amount is small and the setting of experimental parameters is not sensitive. Therefore, this thesis selects a single experimental parameter which has obvious significance for practical application, and does not adjust these parameters with others to optimize the experimental effect [46].

Our strategy is to train and forecast each stock separately, that is, train each part of each stock sequence separately, and then forecast another part of that stock. The morphological sequence of each stock is transformed into a matrix of size $N * K$ (where N is the number of training samples and K is the size of a training sample), in which the first $K-1$ column of each row is the training data and the last data is the label data. Assuming that the category list of each time series segments is X=$[x_1, x_2, x_3, \ldots, x_N]$, the following is a training example:

(x_{n-k}..., x_{n-2}, x_{n-1})→x_n

In the above formula, (x_{n-k}..., x_{n-3}, x_{n-2}, x_{n-1}) is a training sample, x_n is its training label. And we adopt the sliding window method to help the training process, where k is the size of window in the training example. This is shown in Figure 3.6. In our experiments, we adopted $k = 4$.

In order to implement the above idea, we transform each stock into a unique heat code, and then adjust the training set and label set. After that, LSTM algorithm is used to train the data, and finally the new data is predicted by the trained system.

The state data is converted to a unique hot code by using the N-bit status register to encode N states, each of them has its own separate register bits, and only one of

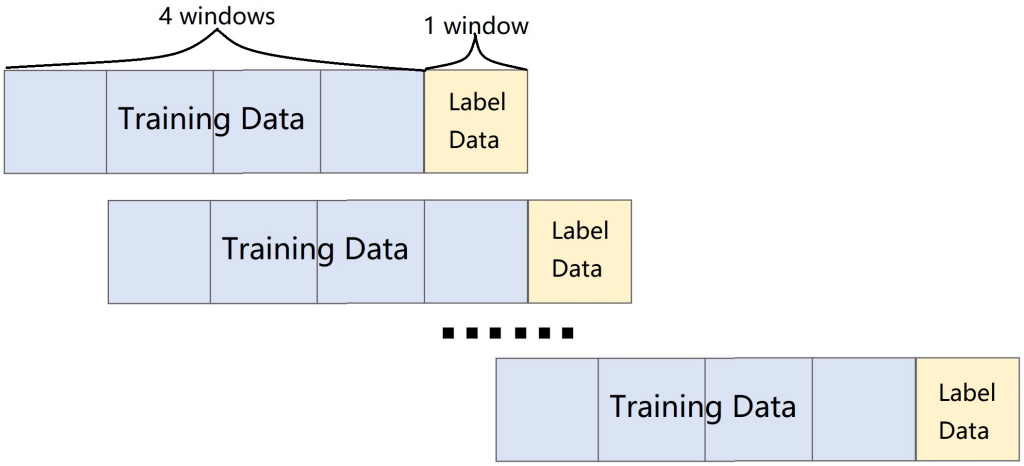

Figure 3.6 – Schematic diagram of training data and label data

them is valid at any given time. The features formed between the data are mutually exclusive, and only one is activated at a specific time, as a consequence the data becomes sparse. This can solve the problem that the classifier is hard to deal with attribute data, and also play the role of feature expansion to a certain extent. Because our classification results tend to get the probability of belonging to a certain category, this becomes very convenient when calculating the loss function or accuracy. The prediction results of the classification show how the state of the time series will change in the foreseeable future. Specifically, the probability that the next sequence will be assigned to each state.

In LSTM algorithm, units parameter represents how many hidden neurons are contained in the cell layer of LSTM. Too many neurons will lead to overfitting of the algorithm, and too few neurons will lead to underfitting of the LSTM algorithm. Based on the experience of previous research, we set the units parameter as 20. As the accuracy of the test set increases first and then decreases with the increase of the number of iterations, too many iterations will only result in a waste of computing resources. Therefore, based on experience of previous studies, we set the number of iterations directly to 200. Another parameter of LSTM is branch size, which means the number of small batch samples for each iteration. Here, we choose to set the size

of this parameter to 35.

In the program of LSTM, we use the softmax function as the activation function, and the equation of the softmax function is as follows [47]:

$$P(y=i) = \frac{exp(\sum_d w_{id} x_d)}{\sum_j exp(\sum_d w_{id} x_d)}, \tag{3.10}$$

The influence of its characteristics on the probability is multiplicative. Because its derivation part is easy to express, it is suitable for application in the neural network back propagation. Therefore, Softmax activation function will improve the efficiency of the optimization process. In the prediction part, cross entropy is selected as the loss function. In the neural network, if the error between the predicted value and the actual value is larger, then in the process of back propagation optimization, the adjustment range of various parameters will be larger, so that the optimization will converge faster. If the error between the predicted value and the actual value is small, the adjustment range of various parameters will be smaller, so as to reduce the shock. Using cross entropy as the objective function, the larger the error is, the larger the gradient of the parameter will be, and the convergence can be rapid. At the same time, Adam algorithm is also adopted in the prediction part to update the gradient. The variation of cross entropy loss with the number of iterations is shown in Figure 3.7.

It can be observed in Figure 3.7 that the cross entropy loss decreases rapidly in the first 25 iterations, and then slows down, and there is a small amplitude of vibration, which indicates that the Adam algorithm can make the loss function convergent, and the entropy optimization effect of the algorithm in LSTM is good.

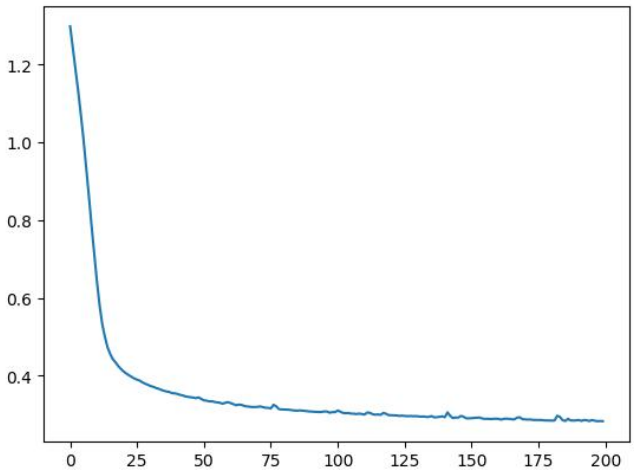

Figure 3.7 – Variation of cross entropy loss with the number of iterations

Chapter 4

Experimental Analysis

In this chapter, we present the experimental results of our method. Our experiments include optimization of values of of system parameters and the optimal number of clusters. The results show that our method is more effective compared with the other algorithms.

4.1 Dataset Description

In order to verify the validity of our proposed stock trend pattern clustering method, we selected 150 stocks from the Yahoo Finance website which fall into five categories (time period 2017-2020 for selected stocks): bank stocks, biotech stocks, medical stocks, oil and gas stocks and semiconductor stocks. In the financial research domain, it is customary to process stock data using the closing price. The factors that can affect stock trends are very complex, including exogenous factors such as the global financial crisis, hence, we only considered stock trends under a relatively stable external environment.

The trend clustering information can be found by directly observing the trend chart of stocks. As shown in Figure 4.1, stocks are in a rising state most of the time, with occasional drops and turns. Therefore, it can be roughly inferred that most of the clustering results should be upward data. The clustering results should be

upward data accounting for the majority, downward data accounting for a relatively small part, and turning data accounting for a small part. Observing the visual data can help us understand better the data.

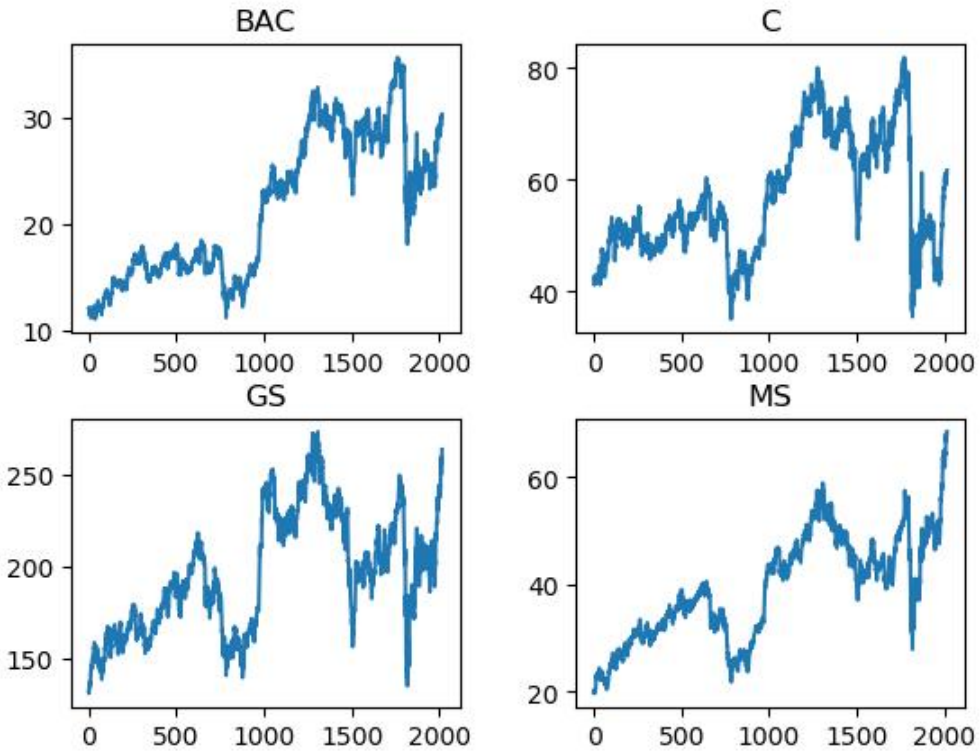

Figure 4.1 – Some example of stock price variance, where the x axis means the date, y axis the price

Table 4.1 – The S-Dbw index values by different clustering numbers

number of cluster	2	3	4	5	6
S-Dbw index	2.7636	1.9523	**1.3017**	3.6308	4.1158

4.2 Stock Trend Clustering Analysis and Evaluation

4.2.1 Determine the number of clustering

After noise reduction, we use window slide strategy to cut the original stock data into segments. We set the window size as 10, and the step as 5, because we expect each series segments could represent the stock information in two trading weeks, and one trading week new information in the next segment. As mentioned in previous section, we use the S-Dbw index to search the optimal number of clusters. We did experiments with various numbers of clusters, from 2 to 6, and presented the results in Table 4.1. According to the variation in values of S-Dbw index, we determine that the optimal number of clusters is 4, i.e. the number allowing to achieve the lowest S-Dbw (1.3017).

4.2.2 Parameters selection

In order to find the appropriate parameters α and n required in the component graph, we conducted experiments to find the appropriate parameter combination by comparing the changes of S-Dbw values under different parameter groups. The smaller the S-Dbw value means the better the clustering effect.

In Table 4.2, the first row represents the choice of parameter n in section 3.3, the first column represents the choice of α parameter in section 3.3. There are some Nan values, which represent that under this parameter group, GCN will only cluster all sequence segments into one category, which means that this group of parameters has no effect on the clustering process. Through experiments, we can observe that when the parameters α =4 and n=25, the clustering effect is the best.

Table 4.2 – S-Dbw index values obtained by GCN clustering in each parameters

	10	15	20	25	30	35	40
1	Nan	24.1307	17.3991	13.9363	11.4663	15.8021	18.3413
2	Nan	9.4387	13.0911	5.4283	17.391	15.6645	19.6029
3	Nan	34.7537	8.9168	2.3159	11.2432	19.4762	11.0116
4	Nan	18.4148	4.8935	**1.2356**	18.4307	10.1324	7.3623
5	36.7231	2.3201	2.0023	2.0337	1.6556	22.4086	49.0954
6	19.5755	5.9595	38.8593	6.0084	2.8195	16.947	11.3662
7	Nan	Nan	3.8513	16.6582	5.0727	6.2566	8.054

Table 4.3 – S-Dbw index values obtained by clustering algorithms

algorithm	GCN	K-mean	DBSCAN	K-shape
Value of S-Dbw	**1.2356**	11.4227	7.3704	1.9971

4.2.3 Comparison with other algorithms

S-Dbw evaluation index was used to measure the clustering results of the data to evaluate the clustering effect of GCN. K-mean algorithm, DBSCAN algorithm, and K-shape algorithm were used to compare with GCN clustering algorithm. K-mean computation is sensitive to the distance between clusters, DBSCAN is sensitive to the density of data distribution, and K-Shape is sensitive to the shape of time series. Therefore, these three algorithms are used to participate in the comparison of clustering effect with GCN algorithm.

As shown in Table 4.3, GCN algorithm clearly outperforms K-mean algorithm and DBSCAN algorithm. This is largely due to the creation of appropriate adjacency matrix by using DTW for computing similarity between time series. DTW allows to compare more effective the shape of time series than the Euclidian distance and the density measures. It is necessary to notice that K-Shape also considers whether the shapes of time series are similar in the process of clustering. However, the problem of size cluster balance is not taken into account, so the performance of GCN in the clustering process is better than that of K-shape algorithm.

4.2.4 Visual exhibition of clustering results

In order to show the trend property of obtained clusters, we visualize the average vector within each cluster. The average vector of a cluster can be seen as the prototype of the cluster. Because the stock series data has been standardized, all of these time series are sequences with a mean of 0 and a variance of 1, so the mean vector is also a vector with a mean of 0 and a variance of 1. This is helpful to compare the properties between clusters. The trend property of these average vectors can basically represent the meanings represented by the clustering categories.

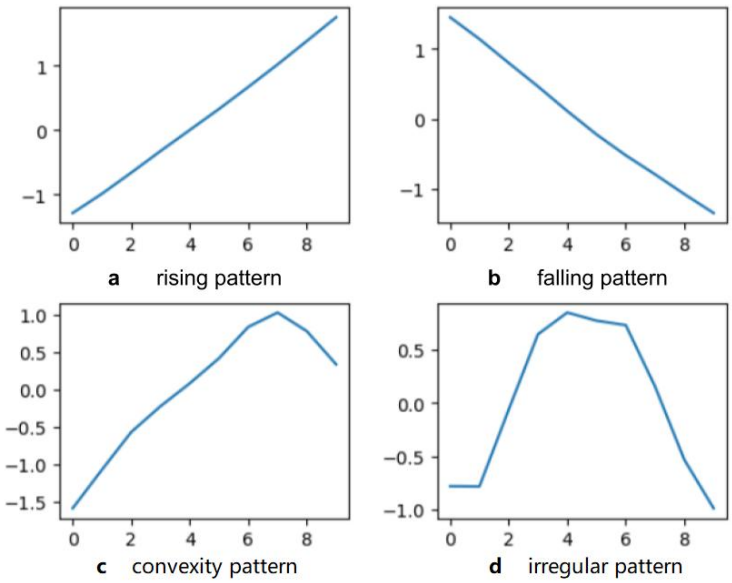

Figure 4.2 – Average value of data for various stocks

Rank the four clusters according to the amount of data contained in each cluster from the largest to the smallest. Figure 4.2 shows the trend of stock closing prices of each cluster in 10 days, where the horizontal axis represents the time (1-10 days) and the vertical axis represents the average normalized closing prices in this cluster every day. In Figure 4.2, curve of subfigure A is the rising pattern, indicating that the rising pattern is dominant within the observation. The curve of subfigure B is a falling pattern, representing the average vector of falling clusters. The curve of subfigure C rises then falls, representing the average vector of the cluster that rises

then falls. The curve of subgraph D seems to rise first and then fall, but in fact there is no change, because compared with the change degree of the other three categories, the ordinate range of the fourth category is smaller, and the data size of this category is also very small. Therefore, it can be inferred that this category represents the average vector of irregular categories.

In order to verify the conjecture above, a sample is selected for observation from each of the following four clustering results. Figure 4.3 to Figure 4.6 is one of the original series data sample from each clusters. From the above argumentation and observation, it can be concluded that the clustering result of this project is basically in line with the reality. The following example presents the correspondence between the raw stock series segments and the four clusters. Obviously, Figure 4.3 corresponding a rising pattern, Figure 4.4 corresponding a falling pattern, Figure 4.5 corresponding a pattern of convex, and Figure 4.6 corresponding a pattern of irregular categories. From the above argumentation and observation, it can be concluded that the clustering part of this project is basically in line with the reality.

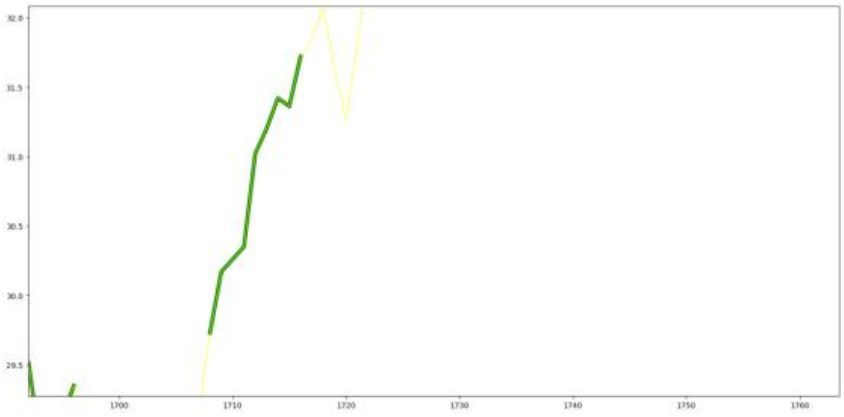

Figure 4.3 – A sample of the rising cluster

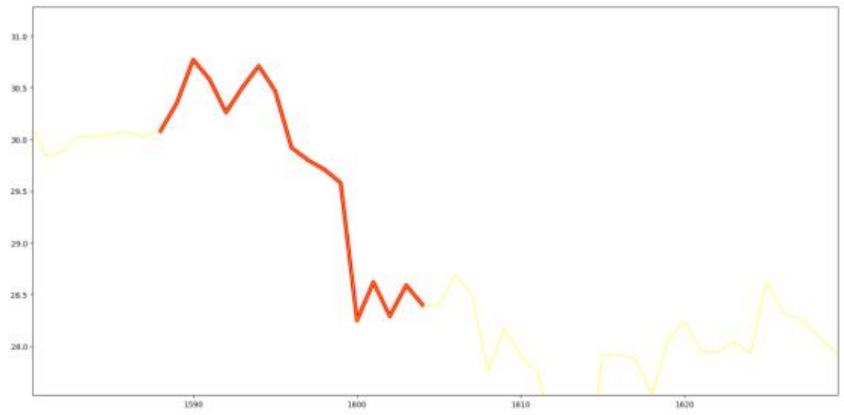

Figure 4.4 – A sample of the falling cluster

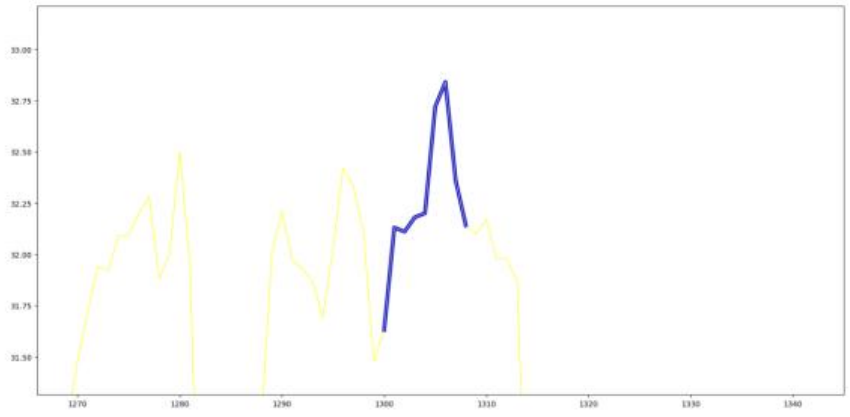

Figure 4.5 – A sample of the convex cluster

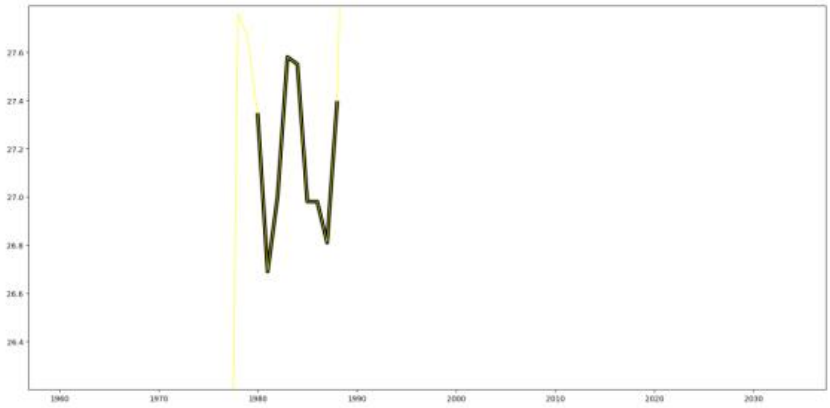

Figure 4.6 – A sample of the irregular cluster

4.2. Stock Trend Clustering Analysis and Evaluation

From the overall change of real stock price, the stock trend properties mined by graph convolution clustering model is in line with the actual situation. We use the raw sequences of bank stocks from this research to exhibit the clustering results of our proposed method in Figure 4.7. Directly observed from Figure 4.7, during this period, the stock price generally rose, more of which showed an upward trend, a few of which fell, and others showed an upward and downward trend or uncertainty. The raw sequences segments data are labeled by cluster categories, and are processed by the t-distributed stochastic neighbor embedding (t-SNE) [48] dimension reduction method to transform the n-dimensional data into 2-dimensional data and displayed in the figure. A red dot represents a time series segment belonging to a rising pattern; a blue dot, a time series segment belonging to a falling pattern; a black dot, one belonging to a convexity pattern; and a yellow dot, a segment belonging to an irregular pattern. It can be seen from the figure that the stock is in a rising state most of the time, with some falling and turning. Therefore, it can be roughly inferred that most of the clustering results are rising pattern, a relatively small part is falling pattern, a smaller part is convex pattern and the rest are the irregular pattern. The results adapt to the directly observation.

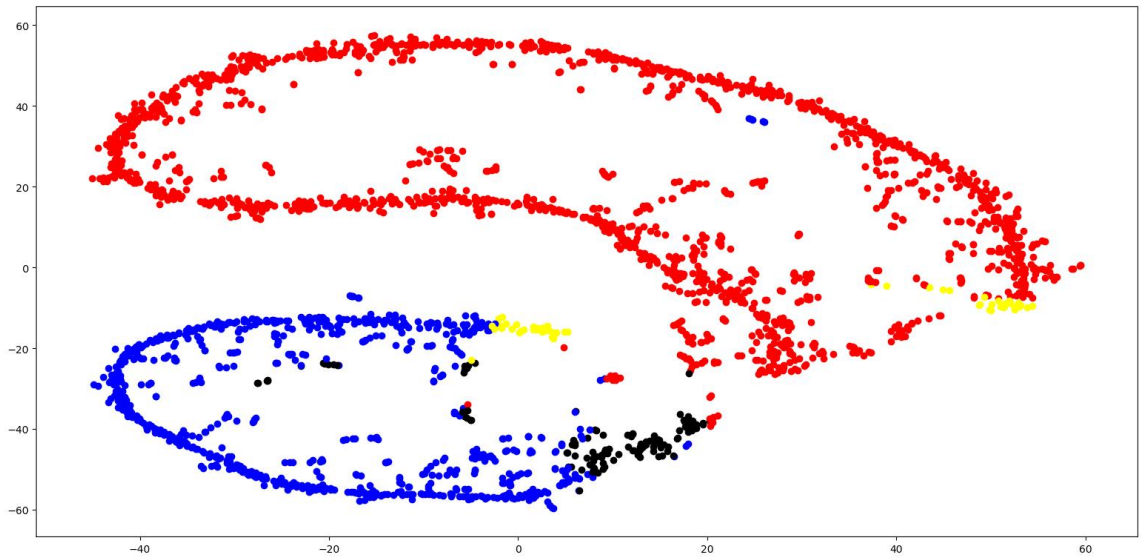

Figure 4.7 – The clustering labels on t-SNE processed time series data

4.3 Experiments and analysis of prediction

Before prediction, we will label the pattern we get from the clustering, they are rising pattern, falling pattern, convex pattern, irregular pattern. Then we will get a pattern series for predicting the future trend of the stock.

In this research, accuracy rate indexes will be observed to measure the clustering effect. Because the prediction result given by the LSTM system is the probability of the cluster to which each data point belongs. The cluster with the highest probability is selected as the prediction result, and the accuracy equation is as follows:

$$accurancy = \frac{c_{max}}{n}, \tag{4.1}$$

Where c_{max} is the number of the largest probability cluster same with its real cluster, n is the number of predicting set.

In this experiment, we use 10 hidden layers, 10 neurons in each layers, 100 epochs in LSTM algorithm in the method we proposed. To observe the performance of our proposed method, we compared our method with some classic methods and recent appear method, including LSTM, CNN-TA, SVM, MLP. Where the LSTM method in our compared experiment is reference from [49], and the CNN-TA method is proposed by [10]. Except the CNN-TA method, the rest of the method for compare all share the same strategy of stock series labeling. In these methods, is labeled as the series data which has a higher price at the end of series than the begin of series as a rising pattern. And we set the other series as the falling pattern. In CNN-TA method, the labeling strategy is following the [10] paper.

We used Bank, Biotech, Medical, Oil and Gas, and Semiconductor stocks from the Yahoo Finance as data for training models. We use the slide window strategy to cut the pattern series into segments for training and testing model. We set the windows size to 5, and the step to 1. And set the last pattern as the label number in each segments, because we can build relevance between the trend information in a future week and its historical month. We choose the data from the front 80 percent of period as the training data and the rest as testing data. We trained the models and observed the accuracy of the rising and falling trend patterns. Model training was

Table 4.4 – The accuracy of each method prediction in kind of stock

stocks type	stock-single					stock-multi			
	GCN-LSTM	LSTM	CNN-TA	SVM	MLP	GCN-LSTM	LSTM	SVM	MLP
banks	**0.7483**	0.7012	0.5469	0.6766	0.6127	**0.7478**	0.6091	0.5691	0.6433
biotech	**0.8048**	0.6693	0.5592	0.6153	0.5765	**0.6621**	0.5983	0.5771	0.5756
medical	**0.7310**	0.7133	0.5631	0.6767	0.6171	0.6901	0.6387	0.6228	**0.7094**
Oil and Gas	**0.6809**	0.6601	0.5448	0.5795	0.5695	**0.6493**	0.5675	0.5733	0.5383
semiconductor	0.6819	**0.7523**	0.5549	0.6844	0.6356	0.7709	**0.7817**	0.6916	0.7258

performed based on either a single stock or on multiple stocks. For the single stock way, we attempt to use the historical data of the stock to predict its future target. On contrast, for the multiple-stock way, besides the target stock, we also use some other stocks' history to forecast the target stock's future price. The left part of Table 4.4 shows the single-stock forecast accuracy, and the right part of Table 4.4 shows the multistock forecast accuracy. It can be seen that the proposed GCN-LSTM method outperforms others both on the single-stock and multi-stock training sets. It is worth noting that GCN-LSTM did better despite the fact that it is harder for a multi-trend detection method to increase its accuracy than for a binary trend detection method. The other methods only need to detect two types of trend, whereas GCN-LSTM needs to detect four.

From Table 4.4, we can find out the proposed method GCN-LSTM can preform more effect than other method. What ever in single stock training set or multi stock training set. Although it is harder for GCN-LSTM to detect these two type of pattern than other method. Because it's harder for a multi trend detection method to increase its accuracy than a binary trend detection method. The other method just need to detect two type of trend, and in GCN-LSTM method, we need to detect 4 different kinds of method.

4.4 Financial evaluation

To further demonstrate the usefulness of the proposed method, we perform financial evaluation. In this evaluation, we simulated stock trading based on some common stock trading strategies while taking advantage of trend predictions by different methods. The initial capital for financial evaluation is 50000 dollar.

Table 4.5 – The profit rate (%) of stocks in each method

	GCN-LSTM	LSTM	MLP	SVM	SMA	NA
banks	**0.72**	-7.13	-7.87	-14.59	-6.14	-17.21
biotech	**27.64**	18.21	21.42	18.88	13.04	17.71
medical	**28.95**	17.59	12.85	18.51	12.11	17.94
Oil and Gas	**6.85**	-3.59	4.96	-24.97	5.38	-33.81
semiconductor	**28.37**	19.77	21.86	16.09	17.68	19.95

We adopt a method presented in the work [50] to simulate the process of stock trading. Noticing that we predict the four types of pre-generated trend patterns, i.e. the rising pattern, the falling pattern, the convexity pattern, and the irregular pattern. If the stock trend is predicted as a rising pattern, the stock will be bought at the closing price of that day with all of the current available capital. If the stock trend is predicted as a falling pattern, the stock will be sold out at the closing price of that day. If the stock trend is predicted as a convexity pattern, half of the stock will be sold at the closing price of that day. If the stock trend is predicted as an irregular pattern, no action will be taken at that day.

Moreover, we make use of two commonly used stock trading strategies as a benchmark to compare with the proposed method. Similar to the paper [50], we chose the Simple Moving Average signal (SMA) strategy as one of the comparison methods. In this momentum strategy, it is hypothesized that use the immediate past period predict the future. To verify the effectiveness of our method, we also added no action (NA) strategy. Just buy and hold the stocks until the experiment end. This passive strategy provides an idea of overall changes of the profit rate over the testing period. We calculate the profit margin of all of the stocks in 52 weeks under each strategy. For illustration purpose, we calculate the average profit margin over each category of stocks, and the results are shown in Table 4.5.

From Table 4.5, we can find out that the GCN-LSTM method obviously outperforms the other methods in terms of profit rate. Due to page limit, we show here only one profit situation of a stock in Figure 4.8. We choose the stock which is coded "ING", and observe the profit situation from Nov 2019 to Nov 2020 by each strat-

egy. Further investigation into the investment actions reveals that the GCN–LSTM method allows adopting a more conservative investment strategy comparing to other methods due to our accurately stock trend prediction. This helps the investors avoid the investment risks and explains why the proposed method is more effective than the others in achieving stable and consistent returns.

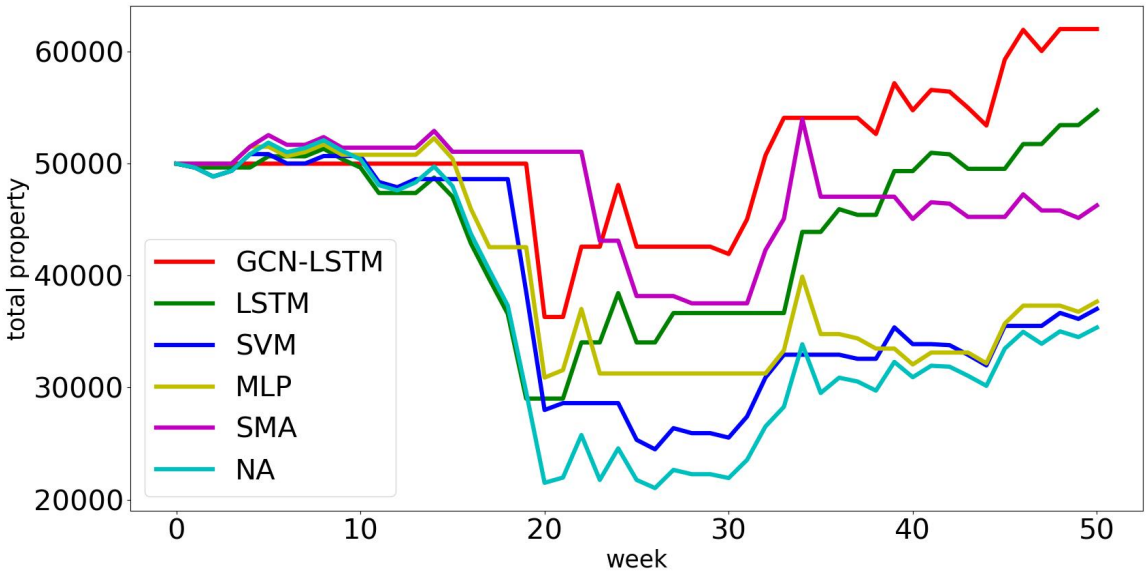

Figure 4.8 – Profits of stock code “ING” from Nov 2019 to Nov 2020

Conclusion

This thesis proposes a stock trend prediction method based on a combination of graph convolutional neural network and short and long-term memory neural network (GCN-LSTM). Our aim is to predict the stock trend in the short term by learning the past trend of the stock. Our approach is as follows : time series are segmented according to time windows to create sub time series or segments of time series. These segments are preprocessed by wavelet transform for noise reduction and Z-score standardization method ; DTW method is then used to construct the similarity matrix between series segments, and the correlation graph model is established according to the similarity matrix. Then, graph convolution is used to perform the clustering of series segments, and the existing trend patterns (types) in stock series are mined. The next step is to label the corresponding pattern (type) of the original series, and form a new pattern type coding ; finally, the deep learning LSTM model is used to train the trend pattern coding sequence to obtain the prediction model, which is used to predict the future stock trend.

In order to verify the effectiveness of the proposed model, we use the historical data of 5 bank stocks, identify their patterns by clustering the stock subsequences using GCN, and then predict the future trend of stocks by using LSTM method to predict the pattern sequences identified. In order to test the validity of clustering, S-DBW clustering evaluation index was used. In order to test the accuracy of prediction, accuracy rate, recall rate and F1-score were used to test the prediction effect of the prediction part. The above studies have achieved good results. The main contributions of this thesis :

1. Discovery of trend patterns in stock sequence segments through an improved clustering method. The stock trend inside a period of time can be divided into a few

forms such as rising, falling, uncertainty commonly. Our graph convolution neural network clustering algorithm has been designed to effectively discover "up", "down", "rise-fall" and "uncertain" patterns from the real stock data.

2. Effectiveness in dealing with the problem of class unbalance in clustering. We identify cluster imbalance problem in stock data clustering and improve the effectiveness of graph convolution neural network clustering by proposing a new method to define adjacency matrix. By using S-DBW index and model selection, we were able to discover the optimal number of clusters to detect, and consequently outperform widely used mehtods such as k-mean and K-Shape in time series clustering. The experimental results show that the new method is better for the data clustering of unbalanced classes.

3. As for stock prediction, our proposed method has potential to help decision-makers see the possibility of profit and avoid risks. In the prediction of each stock, we can observe that the prediction accuracy of the stock pattern sequences can reach over 70 percent on average. Not only do we look at the accuracy of stock sequences, but we also look at the accuracy and recovery rates of specific categories. That will help the decision-makers make the correct decision in short-term or in long-term decision.

This study may have a broad prospect in both future application and research. Here, we use data in days, but the stock market often changes in minutes or even seconds, so this method can be extended to the future minute prediction. In addition, in this study, we predict stock trends by studying individual stock data. In real applications, it is normal for multiple stocks to influence each other. Therefore, in the future, we should try to predict the trend of stocks by considering multiple time series at the same time.

With the development of graph neural networks, the theory and application of other graph neural networks are gradually improved. We can also consider using other graph neural networks to study stock trend prediction, such as Recurrent Graph Neural Network (RecGNNs), Gated Graph Sequence Neural Networks(GG-NN) and so on.

Bibliography

[1] Hirotugu Akaike. Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, 21(1) :243–247, 1969.

[2] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5) :465–471, 1978.

[3] William F Trench. Explicit weighting coefficients for predicting arma time series from the finite past. *Journal of computational and applied mathematics*, 34(2) :251–262, 1991.

[4] Janos Abonyi, Balazs Feil, Sandor Nemeth, and Peter Arva. Modified gath–geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets and Systems*, 149 :39–56, 2005.

[5] Ming-Chi Lee. Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8) :10896–10904, 2009.

[6] Ivana Marković, Miloš Stojanović, Jelena Stanković, and Milena Stanković. Stock market trend prediction using ahp and weighted kernel ls-svm. *Soft Computing*, 21(18) :5387–5398, 2017.

[7] Mustafa Onur Özorhan, İsmail Hakkı Toroslu, and Onur Tolga Şehitoğlu. Short-term trend prediction in financial time series data. *Knowledge and Information Systems*, 61(1) :397–429, 2019.

[8] James Leonard and Mark A Kramer. Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering*, 14(3) :337–341, 1990.

[9] Luca Di Persio and Oleksandr Honchar. Artificial neural networks architectures for stock price prediction : Comparisons and applications. *International journal of circuits, systems and signal processing*, 10(2016) :403–413, 2016.

[10] Omer Berat Sezer and Ahmet Murat Ozbayoglu. Algorithmic financial trading with deep convolutional neural networks : Time series to image conversion approach. *Applied Soft Computing*, 70 :525–538, 2018.

[11] Omer Berat Sezer and Ahmet Murat Ozbayoglu. Financial trading model with stock bar chart image time series with deep convolutional neural networks. *arXiv preprint arXiv :1903.04610*, 2019.

[12] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50 :159–175, 2003.

[13] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert systems with applications*, 36(2) :3839–3844, 2009.

[14] Pei-Chann Chang et al. A novel model by evolving partially connected neural network for stock price trend forecasting. *Expert Systems with Applications*, 39(1) :611–620, 2012.

[15] Cem Kocak. Arma (p, q) type high order fuzzy time series forecast method based on fuzzy logic relations. *Applied Soft Computing*, 58 :92–103, 2017.

[16] Mu-Yen Chen and Bo-Tsuen Chen. A hybrid fuzzy time series model based on granular computing for stock price forecasting. *Information Sciences*, 294 :227–241, 2015.

[17] Hui Qu and Yu Zhang. A new kernel of support vector regression for forecasting high-frequency stock returns. *Mathematical Problems in Engineering*, 2016, 2016.

[18] Manas Ranjan Senapati, Sumanjit Das, and Sarojananda Mishra. A novel model for stock price prediction using hybrid neural network. *Journal of The Institution of Engineers (India) : Series B*, 99(6) :555–563, 2018.

[19] Dharmaraja Selvamuthu, Vineet Kumar, and Abhishek Mishra. Indian stock market prediction using artificial neural networks on tick data. *Financial Innovation*, 5(1) :1–12, 2019.

[20] Muhammad Hafid Riza Alvy Syafi'i, Eka Prasetyono, Muhammad Khanif Khafidli, Dimas Okky Anggriawan, and Anang Tjahjono. Real time series dc arc fault detection based on fast fourier transform. In *2018 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, pages 25–30. IEEE, 2018.

[21] Ashwani Kumar Yadav, R Roy, Archek Parveen Kumar, Ch Sandesh Kumar, and Shailendra Kr Dhakad. De-noising of ultrasound image using discrete wavelet transform by symlet wavelet and filters. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1204–1208. IEEE, 2015.

[22] Guohui Li, Qianru Guan, and Hong Yang. Noise reduction method of underwater acoustic signals based on ceemdan, effort-to-compress complexity, refined composite multiscale dispersion entropy and wavelet threshold denoising. *Entropy*, 21(1) :11, 2019.

[23] Munshi Yadav and M Afshar Alam. Dynamic time warping (dtw) algorithm in speech : a review. *International Journal of Research in Electronics and Computer Engineering*, 6(1) :524–528, 2018.

[24] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. Generalizing dtw to the multi-dimensional case requires an adaptive approach. *Data mining and knowledge discovery*, 31(1) :1–31, 2017.

[25] Jingyi Shen, Weiping Huang, Dongyang Zhu, and Jun Liang. A novel similarity measure model for multivariate time series based on lmnn and dtw. *Neural Processing Letters*, 45(3) :925–937, 2017.

[26] Manabu Okawa. Template matching using time-series averaging and dtw with dependent warping for online signature verification. *IEEE Access*, 7 :81010–81019, 2019.

[27] Xiaogang Ruan and Chongyang Tian. Dynamic gesture recognition based on improved dtw algorithm. In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2134–2138. IEEE, 2015.

[28] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv :1404.2188*, 2014.

[29] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs : Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3) :83–98, 2013.

[30] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR*, volume 2014, 2014.

[31] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29 :3844–3852, 2016.

[32] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[33] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350. PMLR, 2015.

[34] Hongdou Yao, Xuejie Zhang, Xiaobing Zhou, and Shengyan Liu. Parallel structure deep neural network using cnn and rnn with an attention mechanism for breast cancer histology image classification. *Cancers*, 11(12) :1901, 2019.

[35] Yueyu Dong, Fei Dai, Xiaolong Xu, Mingming Qin, and Zhenping Qiang. Empirical research on cluster analysis of spectral information of hyperspectral remote sensing data. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 761–766. IEEE, 2020.

[36] Tong Liu and Zheng Wang. schicembed : Bin-specific embeddings of single-cell hi-c data using graph auto-encoders. *Genes*, 13(6) :1048, 2022.

[37] Csaba Legany, Sandor Juhasz, and Attila Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS international conference on artificial intelligence, knowledge engineering and data bases*, pages 388–393. World

Scientific and Engineering Academy and Society (WSEAS) Stevens Point ..., 2006.

[38] Jelmer M Wolterink, Tim Leiner, Max A Viergever, and Ivana Išgum. Generative adversarial networks for noise reduction in low-dose ct. *IEEE transactions on medical imaging*, 36(12) :2536–2545, 2017.

[39] Atefeh Heydari, Mohammadali Tavakoli, and Naomie Salim. Detection of fake opinions using time series. *Expert Systems with Applications*, 58 :83–92, 2016.

[40] Rongheng Lin, Budan Wu, and Yun Su. An adaptive weighted pearson similarity measurement method for load curve clustering. *Energies*, 11(9) :2466, 2018.

[41] Antonio Parziale, Moises Diaz, Miguel A Ferrer, and Angelo Marcelli. Sm-dtw : Stability modulated dynamic time warping for signature verification. *Pattern Recognition Letters*, 121 :113–122, 2019.

[42] Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. Linkage based face clustering via graph convolution network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1117–1125, 2019.

[43] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks : A deep learning framework for traffic forecasting. *arXiv preprint arXiv :1709.04875*, 2017.

[44] Kanghao Shao, Zhongnan Zhang, Song He, and Xiaochen Bo. Dtigccn : Prediction of drug-target interactions based on gcn and cnn. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 337–342. IEEE, 2020.

[45] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[46] Tolga Ergen and Suleyman Serdar Kozat. Efficient online learning algorithms based on lstm neural networks. *IEEE transactions on neural networks and learning systems*, 29(8) :3772–3783, 2017.

[47] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv :1611.01144*, 2016.

[48] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40 :100378, 2021.

[49] Siyu Yao, Linkai Luo, and Hong Peng. High-frequency stock trend forecast using lstm model. In *2018 13th International Conference on Computer Science & Education (ICCSE)*, pages 1–4. IEEE, 2018.

[50] Wei Chen, Manrui Jiang, Wei-Guo Zhang, and Zhensong Chen. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences*, 556 :67–94, 2021.